

Butler University Fall 2016

EPICS Ronald McDonald House Service Project

Chris McDonald, Austin Bange, Alex Sottile, Jordan Sloan

12-12-2016

Table of Contents

Table of Contents	1
Abstract	2
Chapter 1: Introduction.....	2
Chapter 2: Requirements Specifications	3
Chapters 3 and 5: Architecture and Implementation	3
Chapter 4: Design	6
Chapter 6: Quality Assurance and Testing	6
Chapter 7: Project Organization and Management	7
Chapter 8: Future Work	9
Appendix.....	10
References	10
Weekly Status reports.....	10
Source Code	22

Abstract

The Ronald McDonald House Charities of Central Indiana is a non-profit organization dedicated to serving as a home-away-from-home for families whose children are currently staying at the Riley Hospital for Children or other children's hospitals in Indianapolis. It takes many gracious volunteers to make the Ronald McDonald House thrive. Volunteers can help in many ways: providing a meal for the House residents or doing an in-house service project. The House has no formal way of scheduling in-house service projects; they must correspond with the volunteers through a chain of emails. This process is inefficient and can lead to confusion. The EPICS team has been tasked with building a web app to improve this process. The web app will have an interactive calendar that volunteers will be able to use and schedule their service events with. The web app will also have a summary page of the many types of in-house service projects volunteers can engage in as well as look aesthetically appealing and be easy to use. The team has successfully built the web app; however, there is some future work that can be done. For example, the web app could automatically send an email to the House when a group signs up for a service event or have a logon system.

Chapter 1: Introduction

The development team is led by Chris McDonald, a senior Computer Science major who has plenty of web design experience. Austin Bange, a senior Interactive Media major, designed the look of the web app as well as served as the liaison between the team and the client. Alex Sottile, another senior, served as the team's webmaster and documentation lead. Rounding out the team is junior Computer Science major Jordan Sloan, who, having never previously worked with JavaScript, learned the language and built some of the web app's JavaScript controllers.

The client, the Ronald McDonald House Charities of Central Indiana, is a non-profit organization that houses families from afar whose children are inpatients at the local children's hospitals. They are located in downtown Indianapolis in two separate locations. The House has many volunteers that coordinate the day-to-day operations of the House, but even more volunteers help the House by providing a meal for residents or doing an in-house service project.

The House interacts with many volunteers on a daily basis. They use another Ronald McDonald's website Take-a-Meal to schedule volunteers providing meals for families. However, scheduling for the in-house service projects are done through a string of emails. The volunteers must email Mary Friend, the volunteer coordinator, and she must email back the available dates, as well as all the information relevant to their service project. This is inefficient and has caused confusion for both the volunteers and the coordinators of the House. In addition, the current volunteering website is information-heavy and it is hard for end users to find pertinent information regarding their particular service project. The EPICS group has been tasked with building a web app that schedules these in-house service events. The web app should have a calendar so it is easy for end users to see available dates as well as be easy to schedule a service project. This web app will not only speed the scheduling process greatly, but will provide an intuitive, easy-to-use website for volunteers. It will reduce the amount of time

spent trying to schedule an event, and give more time to volunteers to help the House and make a difference.

Chapter 2: Requirements Specifications

The top priority of the web app is to have a functioning calendar to allow the House and volunteers to better schedule events. Users should be able to click on a date and the web app will schedule their volunteering for that date. The calendar should only allow users to schedule events for future dates (the House may desire a buffer of time after the current date in which users cannot schedule events, for example, events must be scheduled five days in advance). When a date is clicked, a form will pop up, allowing users to enter in their volunteering information, for example, number of volunteers, type of service project, and primary contact email. The calendar will be a dynamic webpage so users will see events appear on the calendar as they happen and will not have to refresh the page.

In addition, the web app should contain relevant information as to the service opportunities. It will have a list of the service projects with a short one sentence description of each project. Users will be able to click on a project type to see more details about the project as well as what supplies they will need to provide. This will reduce the amount of overall information on the web page but still contain all the information volunteers need to complete their service.

The calendar and the list of service projects are the most important primary functions of the web app. A secondary function is to have the web app send the Ronald McDonald House an email notification whenever a group signs up for or cancels a service project. The House also suggested to not allow cancels online after a day or two before the event as to require users to cancel over the phone with such short notice. Eventually the web app should require sign ups to be confirmed by the House before they are official. The calendar should also have a waitlist/queue for multiple signups on the same day, so when one group cancels there is another group waiting to fill the void. Lastly, a login/signup system can be implemented into the web app, so the House has all the volunteer information in an easy to access database (See Chapter 8: Future Work for more details). These implementations, while important, are secondary to having a functioning web app and calendar that is intuitive and easy to navigate through and use.

Chapters 3 and 5: Architecture and Implementation

Technology Stack:

- **Front-End:**
 - o HTML5
 - Markup language used for the static web pages.
 - o CSS3

- Styling tool used for the front-end application.
- o Javascript
 - Base programming language for the front-end application.
- o JQuery
 - Javascript framework used as a dependency for other tools such as Bootstrap and AngularJS.
- o Bootstrap
 - CSS library used to improve the graphics of the front-end application.
- o AngularJS
 - Javascript framework used to control the routing and data access for the front-end application.
- o Grunt
 - Popular front-end server tool used to serve the static html files i.e. the front-end application.
- o Bower
 - This is used to install the dependencies for the front-end tools such as AngularJS, Bootstrap, JQuery, etc.
- o NPM
 - Node package manager. This is used to install the dependencies for Grunt.
- **Back-End:**
 - o Node.js
 - Programming language used to write the API.
 - o Express

- Node.js library used as the server for the API.
- Mongoose
 - ORM for mapping objects to the database. This allows the user to create a model for the objects being sent/received by the API.
- NPM
 - Node package manager. This is used to install the dependencies for the API.

How-To For Developers:

- First, you have to download the repository with all of the code in it. Everything is in one repository. You can either download the .zip file with the code or receive access to the repository on GitHub.
- The first thing you need to install is Node.js with NPM. We will use NPM to install everything else. After installing Node.js, use this command to install the database (you may need to use sudo): `npm install mongodb`.
- Now the database is installed and you can use: `'sudo mongod'` to start running the database. Once it is running, you can use `'mongo'` to access the command prompt for mongo to see databases, collections and query the database, search and delete.
- Now if you go to the directory for the repository `'volunteer-service'`, you can begin installing the dependencies.
- You can run: `'npm install'` to install all of the npm packages. Again, you may need to use sudo (`sudo npm install`).
- Then, you need to run `'bower install'` in order to run all of the bower packages.
- Now the front-end application has everything it needs to run, so now you can use the command `'grunt serve'` and it should begin running on port 8000.
- Now, to get the back-end running, you need to go into the nodejs directory and then run `'sudo npm install'` again there.

- o After running `npm install` in the nodejs directory, you can go into the server directory and use the command `'node www'` in order to run the server.
- o Now the server for the API should be running on port 3000.

Chapter 4: Design

The user interface is meant to be simplistic and intuitive. We wanted to keep the application minimalistic as not to confuse or mislead any users but get right to the point. There is not a lot of fluff with our service because most everything other than the application itself can be found on the original website. We went with a flat and modern look using red as our primary color and a dark gray (almost black) as our secondary. With that, for the most part we kept from using too many round edges to make the appeal sharp. The navigation is visible at all times to help navigate through pages at any moment.



Chapter 6: Quality Assurance and Testing

Our main method of testing, although it was under a tight time constraint, was a survey by the client on the prototype version of the application. Once a fundamentally running application was developed, it was sent to the client for evaluation and observation. Any comments made by the client were taken into consideration for the next version of the project. A document could be developed in order to better survey the client in specific areas of the application. Chris McDonald, the team lead, was the main problem solver for the project. When

an issue would develop, it would then be reported to Chris for troubleshooting. We made sure that the application developed up until this point exhibited the essential main points brought up in the first meeting with the client, including the ability for volunteers to make volunteer appointments directly through the application, as well as the ability to edit appointment details and/or cancel.

Chapter 7: Project Organization and Management

Our organizational structure consisted of four members all holding a unique role. We had a team lead, client liaison, developer, and documentarian. Each has its own responsibilities aside from working on the application itself.

Team Lead: (Chris McDonald)

- Give direction to the project
- Assign tasks to other team members
- Manage the Github repository and monitor consolidations to master
- Meet with professor about project weekly
- Make decisions about which software and framework to use
- Write API
- Design and create database
- Host server
- Implement functionality of calendar

Client Liaison: (Austin Bange)

- Convey updates to client
- Schedule meetings with client
- Design the user interface
- Customize calendar

Documentarian: (Alex Sottile)

- Create and upload Weekly Status Reports (WSR)
- Manage project documentation on EPICS website
- Create materials for final presentation
- Create contact page

Developer: (Jordan Sloan)

- Develop html for the home page
- Document final product and its functions
- Contribute to backend development
- Hack the application

Work was divided among members based on competencies to the extent that members most likely worked on the project in areas that they were most comfortable. Upon arriving to class time each day, the team lead would provide the direction of progress of each member for the day.

The process of our project management consisted of meeting with the client at the beginning of the project to understand its expectations. We then continued to take some creative freedom in meeting those expectations. We met as a group twice a week, trying to make as much progress together as we could, some work was completed outside meeting times. Periodically we sent updates to the client with our progress until we had a working demo for them to try.

We used a number of different technologies to assist our production:

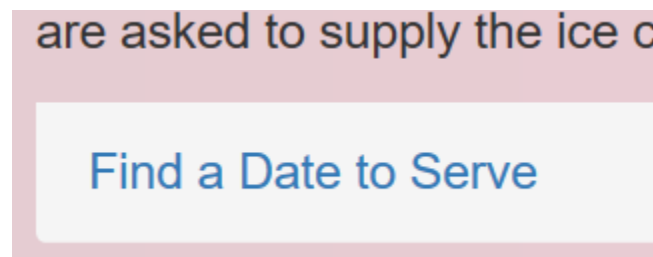
- JetBrains' WebStorm IDE
- AngularJS
- NodeJs
- GitHub

Below is a Manual to How to Use the Application

The first step to using the application would be to enter a browser, and enter the IP address in to visit the application: <http://34.193.243.89:8000/>

From this page, the user can learn more about each way they could volunteer. There are links for each opportunity to learn more, where some of the links revert the user back to the already existing RMH website.

Assuming the user wants to do a volunteer service project, in the “In-House Service Projects” section of the page, they would click the link:



This will lead the user to the calendar page, where they can schedule an appointment.

How to Schedule an Appointment

In order to schedule an appointment, the user clicks on the space of the calendar corresponding with the day they would like to serve. When doing so, a form will pop-up requesting appointment details. The user enters the details, and clicks Submit. This will create a visual event on the calendar, as well as send confirmation emails to both the RMH and the user.

Contacting the Ronald McDonald House

By clicking the “Contact Us” link in the left hand column of the page, the user will be led to a page that has contact information for the RMH.

See Appendix for WSR's.

Chapter 8: Future Work

Further implementations that need to be tackled include:

- Volunteer ability to create an account w/ user credentials
 - Ability to create an appointment and a form to fill out with appointment details
 - Ability to view already scheduled appointments and appt. details
 - Ability to edit appointment details and/or cancel within a certain time frame
- Blacklist for repeat “no-show” offenders for appointments
- A waitlist and/or queue for days with already scheduled volunteers
- Automatic email confirmation sent to client and volunteer when an appointment is made
- Ability to schedule multiple appointments in a single day (number of appointments at the client’s discretion)
- Further improved visual aesthetics

Appendix

References

No outside code was used except for an open-source Angular calendar which was used as a template for the web app calendar. It can be found at this URL: <http://angular-ui.github.io/ui-calendar/>.

Weekly Status reports

SUBJECT: Status report for week 9/12-9/16

I. RED FLAGS: none

II. ISSUES: We need to meet with the client to discuss the project in more detail before we begin work. Austin is setting up a meeting with the client.

IV. ACCOMPLISHMENTS (dates):

9/7

- Formed team
- Elected team leader and assigned roles
- Contacted client about setting up a meeting

V. ACTION ITEMS FOR FOLLOWING WEEK (9/12-9/16):

- Organize client meeting (Austin)
- Outline project (Chris)
- Create/update project web page (Alex)

WEEKLY STATUS REPORT (WSR)

9/19

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 9/19-9/23

I. RED FLAGS: none

II. ISSUES: None. The team has a project outline however we are waiting to hear more from the client before proceeding.

III. ACCOMPLISHMENTS (dates):

9/15

- Scheduled meeting with Client (9/28)

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- Meet with Client – 9/28 @ 2:30 (Team)
- Form a project plan after details are finalized (Chris)
- Maintain website & document meeting (Alex)

WEEKLY STATUS REPORT (WSR)

9/26/16

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 9/26-9/31

I. RED FLAGS: none

II. ISSUES: must meet with the client before finalizing project outline

III. ACCOMPLISHMENTS (dates):

9/28

- Meeting with Client

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- Meet with Client on 9/28 (Team)
- Form a project outline (Chris) by 10/3

WEEKLY STATUS REPORT (WSR)

10/3

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 10/3-10/8

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

9/28

- Met with Client
- Discussed their goals and needs

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 10/3 – Form a project outline (Chris)
- 10/3 – Upload meeting notes and other documents (Alex)
- 10/3 – Begin building the prototype (Team)

WEEKLY STATUS REPORT (WSR)

10/10/2016

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 10/10 – 10/14

I. RED FLAGS: none

II. ISSUES: none

III. ACCOMPLISHMENTS (dates):

10/3

- Sketched a template for what each page of the web app should look like

10/5

- Developed a project plan
- Downloaded WebStorm, an IDE for web development
- Began building architecture for project (overall layout so team members can add functionality and design later)

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- Download WebStorm (Team members who do not have it yet) – 10/10
- Finish building project architecture (Chris) -10/10
- Begin building Functionality (Team)

WEEKLY STATUS REPORT (WSR)

10/17

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 10/17 – 10/21

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

10/10

- Built project architecture
- Created GitHub repository
- Created HTML templates for front end of prototype

10/12

- Built project architecture
- Created GitHub repository
- Created HTML templates for prototype

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- Build Javascript functionality into HTML files.
- 10/24 – Create a functioning prototype to send the RMH

10/24

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 10/24-10/28

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

10/17

- Found an open-source JavaScript calendar to use as a template
- Implemented the static calendar into WebStorm

10/19

- Began front and back end server implementations

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 10/24 – Finish implementing front and back end servers(Chris)
- 10/26 – Implement the JS calendar base layout (Team)

WEEKLY STATUS REPORT (WSR)

10/31

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 10/31-11/4

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

10/24

- Downloaded and ran NGINX, a server for the website to run on localhost.
- Chris began working on implementing the calendar controller
- Alex began working on the input form for the volunteer information
- Austin continued working on the content and front-end look of the website
- Jordan began working on styling the front end

10/26

- Finished input form HTML layout
- Finished “home page” for our website
- Continued building calendar

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 11/2 – Finish building calendar
- 11/2 – Begin linking webpages together, begin working on and testing calendar functionality.

WEEKLY STATUS REPORT (WSR)

11/7

TO: Team, Professor Linos

FROM: Alex Sottile

SUBJECT: Status report for week 11/7-11/11

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

10/31

- Made visual improvements to home page as well as volunteer input forms
- Implemented calendar's code; though it is not fully functional yet

11/2

- Fine-tuned calendar; is functional now
- Modified calendar to better suit needs
- Began connecting HTML files into the main site

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 11/11 – Have a functioning calendar

WEEKLY STATUS REPORT (WSR)

11/14

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 11/14-11/18

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

11/7

- Created a new Github repository to store the finished product
- Calendar is fully functional; however it does not have the required functionality (for example, it lets us pick dates in the past, etc.)
- Input form is merged with calendar

11/9

- Tested calendar; the calendar works perfectly but we still need to tweak some functionality.

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 11/14 – Implement desired functionality into calendar. Have a working product ready to show to client before December.

WEEKLY STATUS REPORT (WSR)

11/28

TO: Team, Professor Linos

FROM: Alex Sottile

SUBJECT: Status report for week 11/28-12/2

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

11/21 to 11/27 was Thanksgiving break

11/28

- Added click functionality to calendar

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 11/30 – (Austin): Email client about final presentation information
- 11/30 – (Austin & Alex) – build a contact page for our web app
- 12/3 – (All): Show calendar to client for beta black-box testing

WEEKLY STATUS REPORT (WSR)

12/5

TO: Team, Professor Linos
FROM: Alex Sottile
SUBJECT: Status report for week 12/5-12/9

I. RED FLAGS: none

II. ISSUES: none at the moment

III. ACCOMPLISHMENTS (dates):

11/28

- Made a “Contact Us” page on the web app
- Improved calendar to have desired functionality

11/30

- Finished most of the web app styling
- Began working on the final report

IV. ACTION ITEMS FOR FOLLOWING WEEK (dates):

- 12/9 – Create the final presentation slides
- 12/9 – Finish the final report
- 12/5 – Finish front end
- 12/7 – Finish back end and make various tweaks to create the final product
- 12/9 – Finish web app

Source Code

Code can be found zipped with this report or on the EPICS website.