CowChips4Charity

Project Plan

sdmay19-28

Client: Ken Johnson

Advisor: Lotfi Ben Othmane

Connor Rust - Backend Lead

Jack Boike - Backend / Scribe

Ben Meeder - Frontend Lead

Kenneth Ho - Frontend / UX / Meeting Scribe

Daniel Lev - Business Lead (PM)

Alex Lev - Administrative / UI

Team email: sdmay19-28@iastate.edu

Team website: http://sdmay19-28.sd.ece.iastate.edu/

Final Version: 04/29/19

1. Executive Summary

1.1 Acknowledgement

We would like to thank our client, Ken Johnson, for the opportunity to work with him and the Boo Radley Foundation for this project. Ken has provided significant assistance in the forms of financial aid and project context throughout the duration of this project. We would also like to thank Professor Lotfi Ben-Othmane for volunteering his time to be the faculty advisor.

1.2 Problem Statement

The Boo Radley Foundation aids in the research of diseases found in both humans and their animal companions. In order to raise funds and awareness, the organization hosts interactive events at college football games called CowChips4Charity (cowchips4charity). The CowChips4Charity event is a modern version of cow chip bingo. Currently, the CowChips4Charity event is operated manually and is executed without leveraging any technologies. We believe the development of a web application interface for this event will increase the number of participants and resulting donations that the Boo Radley Foundation will receive.

We will develop a cross-platform web application for the CowChips4Charity event to be used during collegiate sporting events and possibly professional sporting events in the future. The web application will provide an efficient way for sporting event attendees to participate in the cow chip bingo game. Users will be able to sign in through their profiles credentials. Once the user is signed in, they will then be able to select the sporting event they are attending and the team they are supporting. The user will be able to acquire square(s) for the cow chip bingo game through a donation via credit card. The selection of square(s) can be done well in advance of the game. Users who select the winning square for their sporting event will be notified via email and will then receive a prize in the mail.

1.3 Operating Environment

The operating environment of our application will be from a laptop or mobile device. The web application is hosted on off-site servers, and thus there are no conditions or hazards from the outside world that we need to take into account. Users will be able to access the application during or prior to the sporting event. Due to a large number of anticipated users and limited cellular coverage during sporting events, the web application needs to be readily available and user-friendly.

2. Requirements

2.1 Functional Requirements

Our team's functional requirements are listed below, the way our team approached our functional requirements is by breaking the application into three separate entities(Front-end, Admin-Panel, and Back-end). The first major component of the application is what our team referred to as the Front-End - through which the User will access the web application and play the game. The other component is what we called the Admin Panel - this component lets the admin control the in-game environment and allows our client to utilize the information from the Front-End. Throughout our agile development process, we worked on developing these functional requirements with other client and our faculty advisor.

Front-End:

1) Login to Account
2) Register an Account
3) Play the game
   a) Select the Organization
   b) Select the Tiles
   c) Donate based off of selected tiles
4) Donate without having an account
5) View Account page
   a) Change the information of the user
   b) View previously selected tiles
6) View about page

Admin Panel:

1. View Crud for Users
   - 1.1. View Records
   - 1.2. Page through records
   - 1.3. Delete Records
   - 1.4. Search records
     - 1.4.1. Clear search
   - 1.5. Add new records
     - 1.5.1. Validation for creating new records
   - 1.6. Export to Excel
   - 1.7. Edit Records
     - 1.7.1. Validation for editing Records

2) View Crud for Games

- 1.8. View Records
- 1.9. Page through records
- 1.10. Search records
  - 1.10.1. Clear search
- 1.11. Add new records
  - 1.11.1. Validation for creating new records
- 1.12. Export to Excel
- 1.13. Create a default game board
- 1.14. Create random game board
- 1.15. Select Organizations
- 1.16. Select winning Tile
- 1.17. Delete Records
- 1.18. Edit Records
  - 1.18.1. Validation for editing Records

3)

View Crud for Organizations

- 1.19. View Records
- 1.20. Page through records
- 1.21. Search records
  - 1.21.1. Clear search
- 1.22. Add new records
  - 1.22.1. Validation for creating new records
- 1.23. Export to Excel
- 1.24. Delete Records
- 1.25. Edit Records
  - 1.25.1. Validation for editing Records

4)

View Crud for Donation

- 1.26. View Records
- 1.27. Page through records
- 1.28. Search records
  - 1.28.1. Clear search
- 1.29. Add new records
  - 1.29.1. Validation for creating new records
- 1.30. Export to Excel

5)

View Crud for Admins

- 1.31. View Records
- 1.32. Page through records
- 1.33. Search records
  - 1.33.1. Clear search
- 1.34. Add new records
  - 1.34.1. Validation for creating new records
- 1.35. Export to Excel
- 1.36. Delete Records
- 1.37. Edit Records
  - 1.37.1. Validation for editing Records

6) Logging in to Admin Panel

   i) Validation for logging in

2.2 Non-Functional Requirements

Our team has developed our Non-Functional requirements from researching industry best practices, client needs, and faculty advising. The Non-Functional requirements were heavily involved in the testing and production of the application. The Non-Functional requirements are listed below and encompass both the "Front-End" and "Admin Panel" which is discussed above.

1) Only the Admin shall be able to select the winning squares (security)
2) The application shall 100% runtime during football games (reliability)
3) The application shall be able to support scalability for multiple football game users (performance)
4) The user shall be able to complete the transaction in one page of the web app (usability)
5) The entire credit card transaction information will be encrypted (security)
6) At least 1,000 users will be able to use the web app as it is designed to work (reliability)
7) The web app shall be able to calculate the entire cost of the donation once the user is "checking-out" within 3 seconds (performance)
8) The winning user shall receive a notification within 12 seconds from when the admin selects the winning square (performance)
9) The web application will be compliant with all policies and regulations set forth by the Boo Radley Foundation (operational)

2.3 Interface Requirements

One main focus of our application as requested by our client was to develop an exceptional UI/UX for our application. Therefore, throughout development, our team worked closely with our client to make sure that the user interface requirements were met. These mainly included such tasks as mocking out our application before starting development. We heavily relied on our client's graphic design team and leveraged their images and color themes to incorporate into our application. In regards to UX, we discussed user business flows with our client and incorporated our functional requirements into an optimized business flow. The reasoning for this is because our client wants an application that is easy to use and also looks professional. Our client will be using this application for fundraising and therefore, the UI/UX is very important to our client and the users for utilization of the system.

3. System Design and Development

3.1 Design Plan

Our team has developed requirements and design documents that has allowed us to visualize how our application will look and perform. Prior to approval, we shared these documents with many different parties including our professor, colleagues, and our faculty advisor. Through many rounds of critiquing and feedback analysis, our team was able to develop an exceptional design. All of the requirements and design documents have been approved by our client and have given us a template for how the application will perform.

Our team created wireframes for the screens that the user and admin will see when they are using the application. The wireframes were approved by our client and our team moved into implementing the actual product. We utilized an agile approach to development in which we continuously worked and showed our client the results following our two-week sprints. Doing this type of agile development gave us time to alter the system if anything was wrong and or misunderstood from our perspective.

Our project has a very defined set of requirements and scope, we also have thorough documentation and wireframes that aided us in ensuring that our team met all of the client's requirements. The gathering process for these requirements took an extensive amount of time. We were able to gather these requirements by interviewing and prototyping in order to receive client feedback. Our team was able to assist in providing our client with information in regards to the best practices of web application development to ensure he understood what options best suited his business requirements. Through having open and exceptional communication with our client, we successfully obtained a list of all of his requirements; both functional and non-functional.

We believe our agile approach to design and development enabled us to complete our project within the specified time constraints. We developed our design ahead of time during team meetings through various mocking technologies(draw.io) and whiteboarding, then we sent this to our client and explained the design. The reasoning behind our design is because there was a technical gap between our team and our client. Furthermore, our client was also very particular about us focusing on UI. This technique of development has been successful for us.

From developing the design documents and thinking about design we have learned a vast array of new things. We have concluded it is best to not only use one design approach or development practice. It is very important to fully understand the requirements of the project and also understand the client. Throughout our project, we were able to tailor our development and approaches to our client needs. Our client is very hands-on and wants to know the progress; therefore we chose to do agile and report progress after sprints.

3.2 Design Objectives, System Constraints, Design Trade-off

Design Objective:

Our team's objective for the design was to make something that was simple to use for our users and also simple for developers to understand and extend. By using best practices for design and having exceptional documentation we believe that our client who isn't as technical as the rest of the team is able to understand our thought process and how our implementation worked..

We heavily concentrated on making the design agile which for us meant we would design before we started coding and also do design throughout programming iterations. Throughout the whole process of making the application after every iteration, we made sure to look at our domain model and also our general software architecture of the project. We had made several big design changes through our project by using this agile approach. Along with this, we believe that we were able to achieve completing the project for this reason. We optimized our design early and often and weren't afraid to make necessary changes to the overall design of our application. While doing these changes we always made sure to create separate documents. This benefited us in regards to regression testing, and if the design worked in theory, but in application ended up failing.

System Constraints:

While developing the design we recognized that the system would have constraints and we need to address them, the non-functional requirements are where our team had to address the system constraints. Our system will see large peak times during sporting events. The system will have to handle these peak times; therefore during design, we chose to use MongoDB because of Mongo's ability to scale horizontally. We utilized services that would automatically handle our reliability constraints by doing things such as load balancing. The system was designed based off of the constraints of the functional and non-functional requirements, please see the requirements above.

Design Trade-Off:

When considering the design we could have used a different type of architecture. For example, we could have had a microservice architecture based off of the Java Spring framework. We believe that any other architecture would be over complicating the scope and requirements of our project. During server selection, we considered three different hosting options: Heroku, Amazon Web Service, and MongoDB(mlab). The team and our client chose to move forward with MongoDB and Heroku due to their competitive pricing options and our team's experience with the technology. This process is used because we have used it in other projects learning from our prerequisites from other courses. We have also taken into consideration the best industry practices, which we applied when making the design of this project. The requirements and screens are separated so we can develop them through separate sprints. After talking with our client and other resources (i.e. professor and advisor) we decided these design documents and process would best fit our project. We extensively focused on all of the main functionalities because that is our client's main concern. Our client isn't as concerned about a lot of the nonfunctional requirements but wants the application to be simple and to have all of the main components working reliably.

One thing we really wanted to focus is being able to utilize outside API's the reason for this is because it is best practice when making software in industry. Utilizing APIs saved us time in development and testing..Furthermore, we will be transitioning this project when we are finished the API's allow the project to be maintained with fewer resources. For example, AWS lets us reach out to users via email. AWS sends and processes the emails, removing the security burden from our team. The same strategy is used for our Stripe API; which we are using for credit and debit card processing, alternatively the client would be held liable if we didn't use a third party credit card processing service.

The benefit of our design is that it is simplistic; we didn't want to over complicate the design or the implementation. We have made our documentation thorough and simple because our team doesn't know the technical knowledge of who the project is being transitioned to. On the contrary, if we did have a more complicated design and made all of our code in-house instead of using API's, we might have had more control of the functionalities. Alternatively, it would be far more challenging to update it in the future. Our team believes we made the right choice because our own code might've worked better for the final prototype but in the long term it will be outdated very fast. Our team has learned to consider the long term sustainability of the application in regards to design. We have thought about not only the code but also the scalability and the reliability of the application; which are very important and not taught in class sufficiently. We have also been conscious of the budget when thinking of design, we were aware we have a tight budget and the tools we have to use to create the application have to be very efficient in terms of dollars to utilization.

To summarize our team has learned to consider a vast array of criteria that we previously didn't consider when developing applications. Along with communicating with a client in regards to changing the scope and requirements of the project. We have also learned to consider a budget while developing an application. In particular, while utilizing software tools (Saas, PaaS).

3.3 Architectural Diagram, Design Block Diagram -- Modules, Interfaces

Process details:

- Our design consists of two actors (regular user and admin)
- Our design is broken up into different events where each event/game has different entities
- To pay and donate we leverage a credit card payment service
- All usage information and events are broken down for each specific game
- Regular users are able to view their overall history and their donations for each specific game

Design Diagrams:

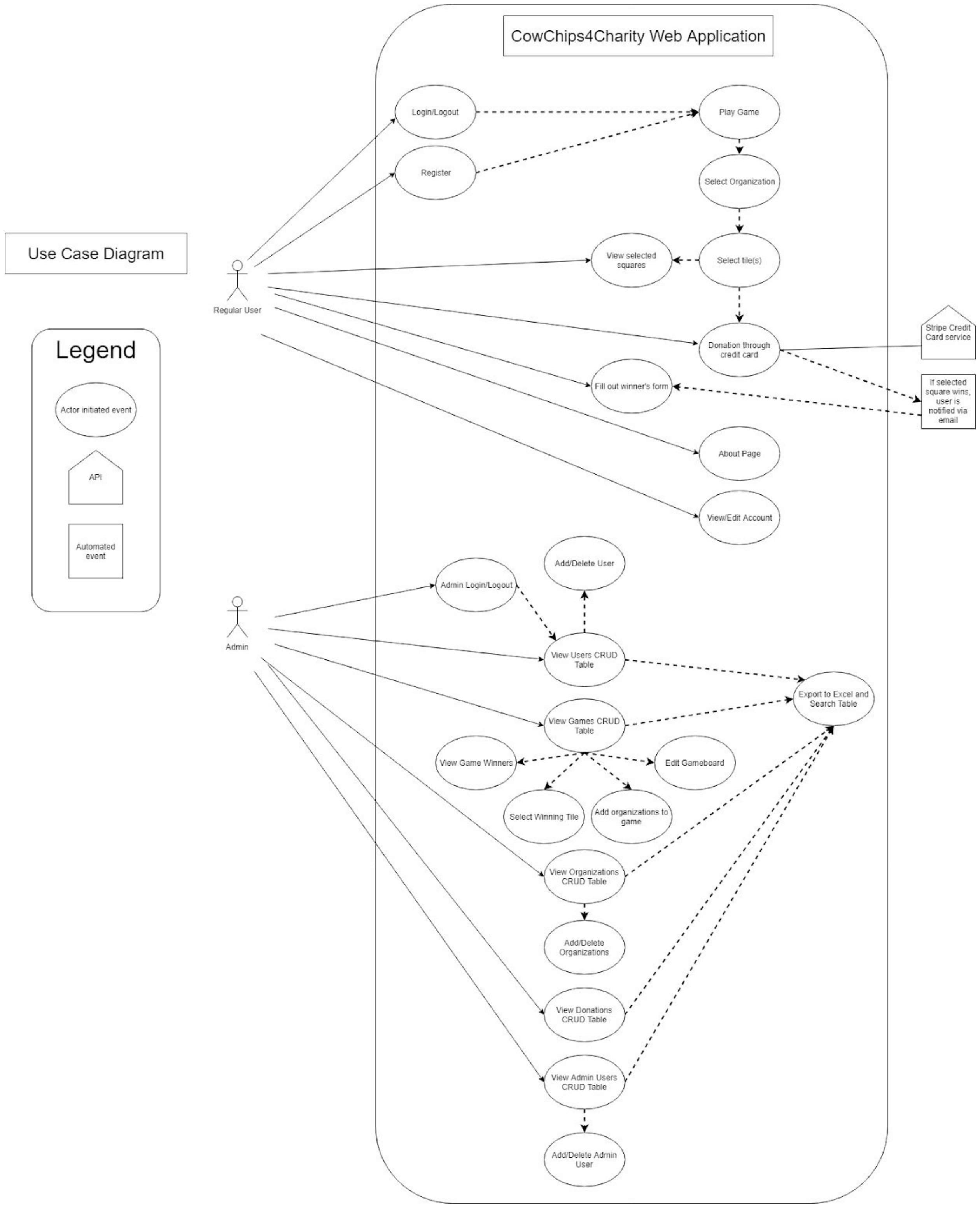


Figure 1: Use Case Diagram

We decided to use a Use Case Diagram to show how our users would interact with the application. In the diagram (Figure 1) it can be seen that there are two types of users that will have separate interactions. The regular users are people who will actually be playing the game, while admin users are the employees of the non-profit actually administering the game. Those users will initiate actions which will either trigger further actions (dotted arrows) or will trigger an automated action (house like pentagon shape) that initiates an external action such as email.

The different components of the application are the adjacent system and users that are apart of the application. We have many different components of the application; the first being the User component, which has all of the functionalities listed in the Use Case diagram. Another main component is the Admin, which has different functionalities and also access to the admin panel which gives information regarding the users, organizations, donations, and events.

Use Case	**Actor(s)**	**Input and Output**
Login/Logout	User/Admin	The User will enter their login credentials. Upon successful completion, they will be redirected to the home screen. The user will also be able to logout of their account
Register	User	The User will be able to register for an account by filling out the desired information of the system. Upon success, the user will be redirected to the home screen.
Select Organization	User/Admin	The input will be the Organization they want to select and donate for. The output will be the confirmation of the event selected.
Donation through Credit Card	User	The User will be able to donate from playing the game and selecting tiles. Each tile has a value X and the number of tiles selected

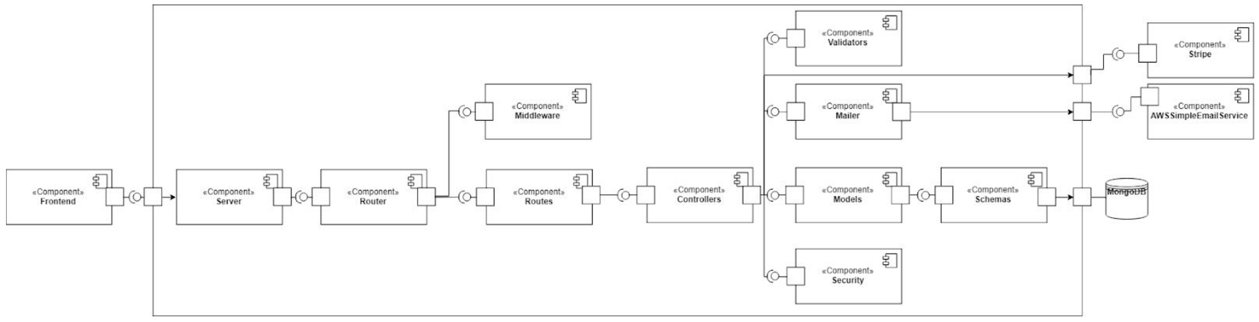
		Y. The amount the user will donate if playing the game is Z= X*Y. A User can also Donate N dollars without playing the game at any time.
Fill out Winning form	User	The User will fill out a winning form if they have won and they need to fill out the information that is missing e.x. User doesn't have an address in their account information.
View about page	User	The User will be able to navigate and view the about page. This explains the application and the Boo Radley foundation
View/Edit Account	User	The User will be able to view their account information. The user will also be able to edit their information in this page i.e. address, name, etc..
Use Case	**Actor(s)**	**Input and Output**
Select Tiles	User	The user will be able to select squares on the bingo board they want to choose and be able to donate to. The output will be that the squares they select can be viewed in their selected squares section.
Admin Login	Admin	The admin will be able to login using their credentials. The output will be a redirect to the admin main page.
Send out Email Blasts	Admin	The admin will be able to send email blasts to a select demographic of users. This can be by event or by team or all of the signed up members. The output of this will be an email is sent out to the respective users.
Select winning square	Admin	The admin will be able to select

		the winning square to the event. The output is going to be users who won receiving an email to fill out the information so they can receive their prizes.
Add/Delete User	Admin	The admin will be able to add a regular user to the system. The admin can also delete or add Admin Users
Export to Excel/Search	Admin	The admin will be able to export a CRUD table to excel and also search through a CRUD table
View CRUD Table	Admin	The Admin will be able to view the different CRUD tables: Users, Organizations, Donations, etc...
Add Records to CRUD Table	Admin	The Admin will be able to add records to any of the CRUD tables.
Edit Game Board	Admin	The Admin will be able to create and edit the game board. The admin can also randomize the tiles on the gameboard by selecting that option.

Table 2: Use Case Diagram Table - Adjacent System / Components

Adjacent System/Components	**Description**
Credit card payment service	We will have an API connected that is a secure source, so a user will be able to enter their credentials and pay for their selected squares.
AWS	The AWS service will send emails to users that the admin wishes to send emails to. This will sometimes include another link where the user can enter their info for their prizes.
User	The user will be able to do all of the main functionalities of the application that are documented in the requirements section of the Project Plan.
Admin	The Admin will be able to do all of the main functionalities of the application that are documented in the requirements section of the Project Plan.

Component Diagram:



The above diagram shows the architecture used on the backend server. As you can see, a chain of responsibility is created through the server, the router, the routes, the controllers, and the models, all leading down to the database interface. In using this pattern, we have separated business logic (contained in the Controllers) from the database logic (accessed by the Models and used in the Schemas). This will allow for easier database upgrades, and database changes if need be. The same has been done for security, mailing, and validating data.

3.4 Description of Modules, Constraints, and Interface

Frontend: Frontend represents the client component, written in VueJS. The frontend will make HTTPS requests to the backend's CRUD API.

Server: The server represents the main entry point that is in control of the system. The server component is in charge of opening and closing the listening ports, opening communication to the database, and system initialization.

Router: The router is in charge of setting up all of the routes. The router can be thought of as the main route, which essentially glues all of the route components together.

Routes: The routes components specify which routes are reachable by the client, and they pass the request parameters to the appropriate controllers for request handling.

Middleware: The middleware components are used for parsing request information, and performing preliminary action before the request is passed on to the controllers. This is used to reduce code repetition for things such as cookie parsing and authentication.

Controllers: The controllers are what handle the requests from the client. Controllers perform any business logic necessary and communicate to the database via the Model components. Controllers are also in charge of sending the response to the client after the request has been handled.

Validators: Validators use the Joi library to validate data sent by the client. Validators will ensure that the data sent by the client follows all syntax requirements and data type requirements to ensure data is not malformed before being processed and/or saved in the database.

Mailer: The Mailer module is used to interact with AWS, but to decouple AWS from the rest of the application. The Mailer module supports email template operations (get, update, delete, and create) and performs the bulk email sending to the winners of the games.

Models: The Models communicate with the database through the Schema components and provide an interface for accessing the database without knowledge of the Mongoose API. This component is mainly used for the abstraction of the Mongoose API and it allows us to embed some logic along with the database communication.

Security: The Security module provides the security operations necessary for our project. Some examples are password hashing and comparison, permissions checking, and cookie creation and authentication using the JWT (JSON Web Token) library.

Schemas: The Schemas are Mongoose created objects, which define document structure and provide an interface for interacting with documents on the MongoDB server.

Stripe: Stripe is an API which provides the debit/credit card processing functionality for our application. Stripe includes the security and validation required for our web app.

AWSSimpleEmailService: The AWSSimpleEmailService component refers to the AWS SES API. SES is used for sending emails using AWS as an email service. SES allows us to send bulk templated emails to a massive number of users for a low price per email.

MongoDB: MongoDB is the database we are using. MongoDB is interfaced with through the Schemas, which utilize the Mongoose library.

4. Implementation

4.1 Implementation Diagram, Technologies, Software Used

Software Implementation Plan:

Our team started by retrieving and refining the requirements obtained from our client. After all of the requirements were gathered, we focused on brainstorming ideas and designs for the product. Then, we set up our databases, repositories, and tools necessary for development. After creating diagrams and documentation, including a wireframe prototype, we began development on the product.

Our development cycle included development, automated testing, and code reviews. After completing a development task, our Project Manager would assign a new task.

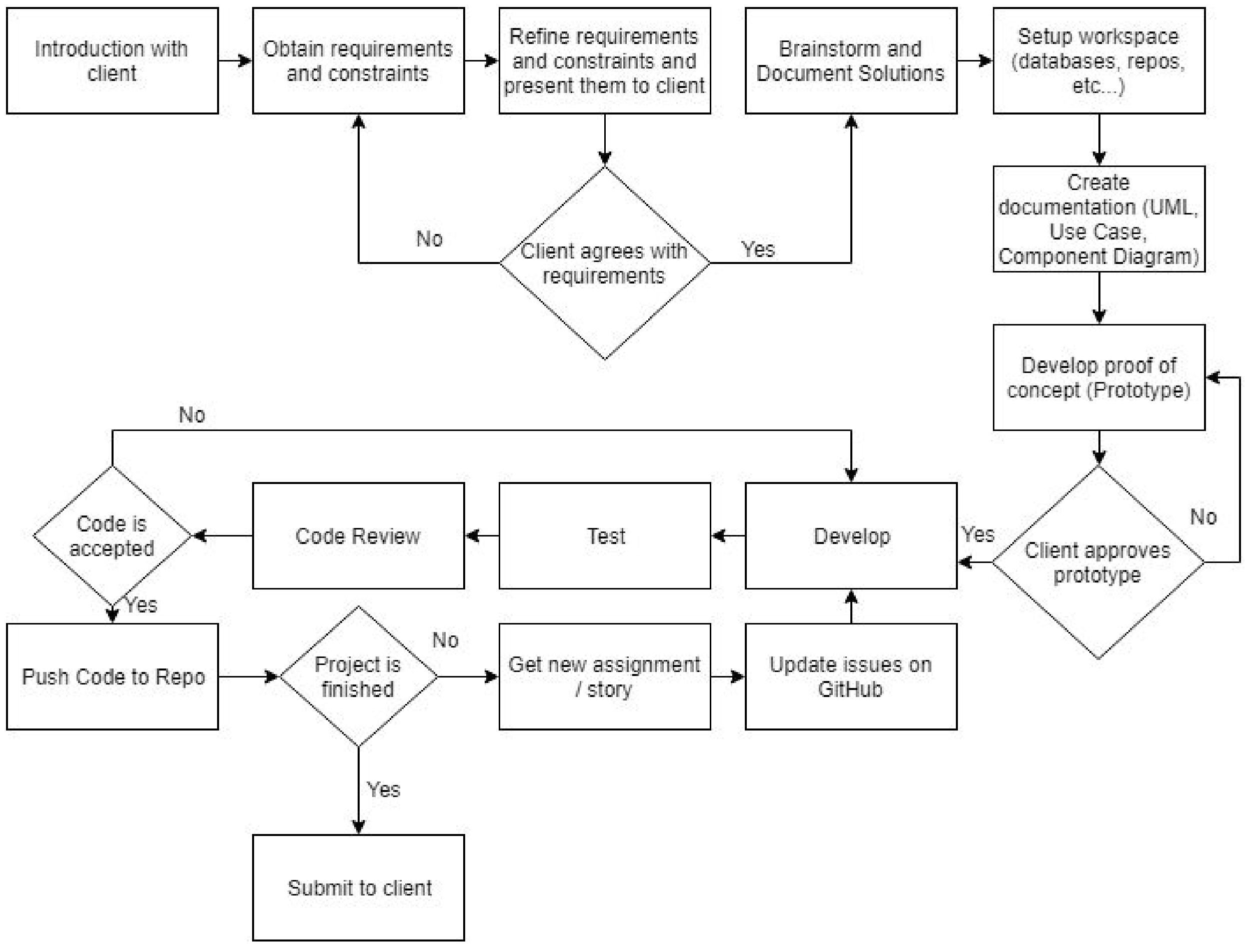

Introduction with client
Obtain requirements and constraints
Refine requirements and constraints and present them to client
Brainstorm and Document Solutions
Setup workspace (databases, repos, etc...)
Client agrees with requirements
No
Yes
Create documentation (UML, Use Case, Component Diagram)
Develop proof of concept (Prototype)
Client approves prototype
No
Yes
Develop
Test
Code Review
Code is accepted
No
Yes
Push Code to Repo
Project is finished
No
Yes
Get new assignment / story
Update issues on GitHub
Submit to client

Figure 3: Project Process Diagram

Table 3: Project Process Dictionary (Continued on next page)

Step	**Description**
Introduction with client	We started with a simple introduction with the client and gathered the main ideas for the product that the client wished to have developed
Obtain requirements and constraints	During this phase, we recorded the requirements for the product based on the information we had gathered during the initial meeting with the client (the empathize phase).
Refine requirements and restraints and present them to clients	This phase is used to ensure that the client agrees with the requirements we have developed for our product. After summarizing our requirements through the refinement of our requirements and constraints (the definition phase), we will present the requirements to the client and if the client approves them, then we will move forward, if not we will go back to the *Obtain requirements and constraints step.*

Brainstorm and Document Solutions	This is similar to the ideating phase in Design Thinking. During this step, we are brainstorming different ways to successfully develop our product in order to cover all requirements from the client. During this phase, we decided what hosting service we will use, how we will divide the components of our project, what technologies we will use, and what the overall system will look like. We also made sure to document our ideas in order to obtain a list of possible ideas and to collectively decide which path to take.
Set up workspace	During this phase, we will do the initial setup for our project before we begin development. We will do this to ensure that our setup works and that we will not face any issues later on during development or the design of the proof of concept.
Create Design Documentation	During this phase, we will create an assortment of diagrams in order to describe the brainstormed ideas and to create a plan of execution and design for the components of our project. These documents will be useful throughout the development phase for planning purposes and after during the baton handoff between our group and our client.
Proof of Concept	This phase will be used as a test of our brainstormed idea. We will present a prototype (or proof of concept) to the client in order to receive feedback about his thoughts. This proof of concept will be a rough-edged design of what our end product should look like. We will most likely not put as much emphasis on this step as our development cycle is iterative and we will present the main features as they are developed to our client for feedback. If the client approves our prototype, we will continue. Otherwise, we will repeat this step until our client believes we have successfully designed an application that will meet his needs.
Develop	During this phase of our process, we will develop the different components that we have identified in order to create our product. This step is part of a cycle, meaning development will be performed iteratively in order to ensure that what we develop is what the client wants. Also, this is to ensure that bugs are found and mitigated.
Test	During this phase, we will test the developed components to ensure that the components work as per our requirements and to ensure the usability of the component.
Code Review	During this phase, we will review the tested code to ensure that the code is well kept and bug-free. This step is important, even though we already have tested the new code. Testing will be used to visually check the functionality of the developed code, but that will not catch all bugs that could be hiding under the surface and in edge cases. If the code is accepted, the process will continue. Otherwise, we will go back to the *Develop step*.
Push Code	Next, the newly accepted code will be pushed to the main repository, where it will become part of the finished product. After this step, if the project is finished, the *Submit to client* step will be executed. Otherwise, the *Get new Assignment / Story* step will be executed.

Get new Assignment / Story	After completing a story, a developer will be assigned a new story/feature to implement. The assignment will be made by our Project Manager, Daniel Lev.
Update issues on GitHub	We will then update the issues on GitHub to show that the old issue was handled and that the developer will begin completing another issue from the issues list.
Submit to Client	During this step, we will present our final working product to the client and hand off all of our code and work in order for the client to continue the development and maintenance of the system.

4.2 Rationale for Technology/Software Choices

Stripe
Online payment processing for internet businesses. Stripe is a suite of payment APIs that powers commerce for online businesses of all sizes and includes fraud protection. We used Stripe to process all of the credit/debit card donations from our users.

AWS
Amazon Web Services offers reliable, scalable, and inexpensive cloud computing services. We used AWS to notify the winners of cow patty bingo games by using AWS's Simple Email Service to send bulk templated emails.

GoDaddy
GoDaddy registers and sells domain names for websites. We use GoDaddy to secure and maintain our website domain for the CowChipsForCharity game.

Mlab
Now part of the MongoDB family, powering over 1 million deployments on AWS, Azure, and Google. We use Mlab as the database for our application storing all of our information.

Heroku
Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. This is used to run our application in the cloud, making it scalable.

VueJS
Our team utilized VueJs to implement the front-end component of our project. We chose this software because it is easy to use and it also makes testing and scaling easier for ourselves and future teams who will build on top of our project. Vue has Vuetify which makes UI easy and reliable for both the developers and Users.

NodeJS
We utilized NodeJS to develop our backend. We decided that the npm packages and our previous Node experience will give us leverage while developing

MongoDB:
We used MongoDB as our database due to its horizontal scalability and ease of use with NodeJS applications.

CoreUI
We utilized CoreUI to help create our admin panel. The templates helped for the UI of our admin panel, furthermore we chose this application because our developers are more familiar with programming rather than design.

Jest - Javascript Automated Test Runner. We will use this to run a set of javascript unit and integration tests. The main advantage of Jest is its powerful watcher that allows for queries based on filename, test name, and suite name. Our team plans to take full advantage of this test runner and fully implement Test Driven Development within our team. Jest will also be run during the continuous integration process in order to verify that new features work correctly and that there are no regressions. Another main advantage of using jest to write our unit tests is the way that unit tests act as "self-documenting" code for future maintainers of the project.

Cypress - Cypress is an end to end browser automation test runner that will run tests in a Google Chrome or Headless Electron browser. This allows the team to create tests that hit every portion of the application simulating real user input and verifying that features are working as expected. We will use this within our continuous integration server to confirm that no breaking changes occur during changes, as well as prove that any new feature that requires user input performs correctly. Cypress is accompanied by tools that allow the developers to watch a video of any test runs and show their errors allowing for easy debugging when builds fail due to integration testing failures.

4.3 Standards and Best Practices

1. ABET practices based off of professors inquiries
2. IEEE Agile Software Development Practices
3. Ethical Programming
4. IEEE Testing Plan (Std 829-1998)

5. Testing, Validation, and Evaluation

5.1 Software Test Plan

Software Test Plan (STP)

This document is an annotated outline for a Software Test Plan, adapted from the IEEE Standard for Software Test Documentation (Std 829-1998).

Iowa State University of Science and Technology

Cow Chips 4 Charity

Software Quality Assurance Plan

Version: (Final) **Date: (04/19/2019)**

Document History and Distribution

1. Revision History

Revision #	Revision Date	Description of Change	Author
1	02/15/19	Started test plan document	Kenneth Ho
2	2/25/19	Incorporated IST best practices	Alex Lev
3	3/15/19	Updated CD/CI industry standards	Ben Meeder
4	4/01/19	Created and IST plan of action	Connor & Jack
5	4/18/19	Record IST and UAT	Daniel

2. Distribution

Recipient Name	Recipient Organization	Distribution Method
Kenneth Ho	CowChips4Chairty	Admin Panel testing
Connor Rust	CowChips4Charity	Back end Testing
Daniel Lev	CowChips4Chairty	Front end Testing
Alex Lev	CowChips4Chairty	Front end Testing
Jack Boike	CowChips4Chairty	Admin Panel Testing
Ben Meeder	CowChips4Chairty	CD/CI testing and maintenance

TABLE OF CONTENTS

5.1.1. INTRODUCTION

(NOTE 1: THE SOFTWARE TEST PLAN GUIDELINES WERE DERIVED AND DEVELOPED FROM IEEE STANDARD FOR SOFTWARE TEST DOCUMENTATION (829-1998)).

1.1 Objectives

The plan of our group is to continuously test throughout the development of our project. Throughout the project, we have utilized continuous development and continuous integration into our environment. A precondition of any of our work reaching our production environment is that our code must pass through our CD/CI environment (Travis CI). Furthermore we unit tested all of our code. The last check was, we made sure someone code reviewed a colleagues code before it was merged into our production environment. The last part of our testing objectives was to run a successful Integration System Testing (IST) - this consisted of us going through all possible business flows of our application and making sure all of our requirements and their exceptions are met. Along with IST we then facilitated UAT to make sure we could have multiple users, and that our application was able to be utilized by people who didn't work on building it.

1.2 Testing Strategy

The test plan for the organization is during development to utilize CD/CI and peer code reviews. After that, we plan on doing IST by running test scripts through all of our business scenarios. These test scripts will include the business flow of our application and all of the possible requirements. Then these documents will be replicated and edited for UAT. The UAT will be in two parts: part 1- is users testing the front-end by following the documents of IST, part 2- will be our client which is the admin of the application testing the admin panel. Below, you can see all of our functional requirements that are both on the Front-end and Admin Panel of our application.

Front-End:

1) Login to Account
2) Register an Account
3) Play the game
    a) Select the Organization
    b) Select the Tiles
    c) Donate based off of selected tiles
4) Donate without having an account
5) View Account page
    a) Change the information of the user
    b) View previously selected tiles

6) View about page

Admin Panel:

1) View Crud for Users
    a) View Records
    b) Page through records
    c) Delete Records
    d) Search records
        i) Clear search
    e) Add new records
        i) Validation for creating new records
    f) Export to Excel
    g) Edit Records
        i) Validation for editing Records

2) View Crud for Games
    h) View Records
    i) Page through records
    j) Search records
        i) Clear search
    k) Add new records
        i) Validation for creating new records
    l) Export to Excel
    m) Create a default game board
    n) Create random game board
    o) Select Organizations
    p) Select winning Tile
    q) Delete Records
    r) Edit Records
        i) Validation for editing Records

3)

View Crud for Organizations
    s) View Records
    t) Page through records
    u) Search records
        i) Clear search
    v) Add new records

        i) Validation for creating new records
    w) Export to Excel
    x) Delete Records
    y) Edit Records
        i) Validation for editing Records

4)
View Crud for Donation
    z) View Records
    aa) Page through records
    bb) Search records
        i) Clear search
    cc) Add new records
        i) Validation for creating new records
    dd) Export to Excel

5)
View Crud for Admins
    ee) View Records
    ff) Page through records
    gg) Search records
        i) Clear search
    hh) Add new records
        i) Validation for creating new records
    ii) Export to Excel
    jj) Delete Records
    kk) Edit Records
        i) Validation for editing Records

6) Logging in to Admin Panel
    i) Validation for logging in

1.3 Scope

Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very 'dependent' activity. As a result, test planning is a continuing activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing. During development as previously mentioned our teams plan is to utilize our CD/CI and colleagues alongside with unit testing our code. As we move through agile development through progressions of our project we will then evolve to IST and finally UAT.

1.4 Reference Material

- *Project Plan*
- *Design Document*
- *IEEE best practices*
- *Advisors advice*
- *Requirements documentation*
- *CD/CI tools*

1.5 Definitions and Acronyms

NA

5.1.2. TEST ITEMS

- *Requirements specification - Can be found in the project plan*
- *Design specification - Can be found in the project plan*
- *Users guide - Separate document in the repository*
- *Operations guide - Separate document in repository titled "Operations guide"*
- *Verification and validation plans. - IST and UAT success scenarios can be found in the project plan*

2.1 Program Modules

- *Create Unit Tests for your component*
- *Make sure your component passes the CD/CI*
- *Assign a peer reviewer to your issue*

2.2 Job Control Procedures

- *Test in development environment*
- *Test IST and UAT in the production environment*

2.3 User Procedures

- *Peer reviews of all documentation*
- *Client and Advisor approval of documentation*

2.4 Operator Procedures

- *Conduct Testing in the development environment*
- *Follow operations manual*
- *Admin has admin documentation*

5.1.3. FEATURES TO BE TESTED

The features that will be tested are all of the features in the requirements specification and scope documentation. This can be found in the project plan, all of these requirements are parallel with our GitHub repository and they're documented on there as well. Our tests all will have documentation alongside with them that will let anyone who takes the project over can see our methodologies and testing strategies.

5.1.4. FEATURES NOT TO BE TESTED

All of our requirements will be tested. There are some nonfunctional requirements that won't be able to be tested due to scheduling i.e. we envisioned having a test event that we could test our application, but the client notified us that the event got pushed to the fall.

5.1.5. APPROACH

5.1 Component Testing

Our team concluded component testing by utilizing unit testing alongside with CD/CI. We broke our application into sectors and once a sector of our application was done our team would have a meeting and move through IST together.

5.2 Integration Testing

- Integration Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test Cases/Scripts
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test
  - Perform

5.3 Conversion Testing

Our system is brand new and we have not converted any of our data or existing systems; therefore we will not have any conversion testing.

5.4 Job Stream Testing

All of our integration testing and user acceptance testing will be on the production environment to make sure that the production build works correctly.

5.5 Interface Testing

We have a UI/UX lead that has created a color scheme and business flow of our application. This was done with communication between our team and our client. During UAT our client gives us feedback for our Interface and we often prototype our interface before we implement.

5.6 Security Testing

During our testing periods, we make sure that all of our requirements exceptions are met and that the system is able to handle all of these exceptions. We also use API's for handling any of the users' confidential data, we made sure to pick API's which are the industry leaders and followed up that they follow current security protocols.

5.7 Recovery Testing

Our data is being stored in the Admin panel which is stored on a separate database than our front-end therefore if either of them fails our application will still be able to operate accordingly.

5.8 Performance Testing

By utilizing Heroku which is a POS this has handled all of our performance testing. This platform guarantees horizontal scaling along with the availability of our application. If needed to buy more space our client can do so at any given time, if needed.

5.9 Regression Testing

Our CD/CI environment is what we mainly rely on for regression testing. That is why the full Travis build must pass before any new components get integrated into our production build. By ensuring our previous components use cases pass, we can guarantee we haven't anything by creating new features.

5.10 Acceptance Testing

The acceptance testing of our system is determined through two phases: IST and UAT. During both of these phases, we will run through our business flow by utilizing test stubs.

5.11 Beta Testing

Our client will conduct beta testing after we hand over our application during an event. We will conduct our beta testing during UAT by simulating all of the business cases of our application.

5.1.6. ENVIRONMENTAL REQUIREMENTS

6.1 Hardware

Our application is not hardware dependent. We use Heroku as a platform as a service to guarantee that our application will have uptime and the necessary performance capabilities.

6.2 Software

The software requirements description can all be found in the project plan under software requirements.

6.3 Security

We rely on our API's to handle this along with all of the outside services that we use. By conducting research and selecting the correct outside tools we believe we have covered our security aspect.

6.4 Tools

As previously mentioned the main tools we utilized in testing is our CD/CI environment.

6.5 Publications

We have relied on support from our faculty advisor along with the professor. Furthermore, we have also been following IEEE specifications.

6.6 Risks and Assumptions

The biggest risk our project phases in terms of testing is that we don't have an actual way to beta test in a live environment. The reason for this is because the purpose of our application is mainly to be used at sporting events, and the event we were supposed to test at got pushed to the fall.

5.1.7. CHANGE MANAGEMENT PROCEDURES

If there is a change in requirements or scope this must be first approved by the project manager and also another member of the team. Then this will get escalated to the client with a written proposal as to why we are making the change. All of our changes and the reasoning behind them are recorded.

5.1.8. PLAN APPROVALS

The approval of the testing plan is as follows :

-Benjamin Meeder

-Daniel Lev

-Alex Lev

-Kenneth Ho

-Jack Boike

-Connor Rust

Final Requirements - Test Scripts - IST & UAT

This Document is used for IST and UAT - when you receive this document navigate to www.cowchips4charity.com to test the Front-end of the application. To test the Admin Panel of the application navigate to www.admin.cowchips4charity.com obtain credentials from the project manager (Daniel Lev - 651-328-7052). The purpose of this document is to be able to iterate through all of our functional and interface requirements. The left-hand column of the test scripts describes the functional and or interface requirement. The Right-hand column will tell you the steps to do for testing this requirement and will also give you an expected result. If the requirement was able to be completed please highlight the whole row of the requirement green. Otherwise, highlight the row red and give an explanation of what is incorrect.

Front-End:

1) Login to Account

Precondition: The user must not be logged in	
1.)User Navigates to Login	The User will select the "Login" tile on the home screen of our page
2.)User Logs into account	The User enters their credentials into the "Email" and "password" section
2a.) User enters incorrect information	The User enters incorrect credentials and the system shall notify them that they have entered incorrect credentials
3.) User will be redirected to the home screen	Upon entering correct information the user will be redirected to the home screen of the application

2) Register an Account

Precondition: The user must not be logged in	
1.)User Navigates to Register	The User will select the "Register" tile on

	the home screen of our page
2.) User Registers	The User will enter their name, email, password, and re-enter to verify their password.
2a.)User enters incorrect information	The user enters invalid information- test for invalid email (no @), passwords mismatching. The system shall notify them of their errors
2b.) User enters an account that's registered	The user enters the account information that already exists. The system notifies the user that the account exists
3) User Registers their account	The user has successfully completed registration and will be redirected to the home screen
4) User receives email from AWS	The User will receive an email to the account they have signed up with. The user will then click the link to verify their account. Once the link is clicked they will be redirected to a congratulations page.

3) Play the game
   a) Select the Organization
   b) Select the Tiles
   c) Donate based off of selected tiles

Precondition: The user is logged into the system	
1.)User Navigates to Play tab	The link should direct the user to the select organization page
2) User selects an organization	The user will select an organization, for example, the organization "Iowa State University". Then the user will be redirected to the tile selection page
3) User selects Tiles	The User will see 36 tiles on the page be displayed and will be able to toggle and untoggle. The price of the tile will be

	displayed on the screen and once toggled the price will go up 1x, if another tile is selected it'll go up 2x, etc...User clicks the button next when done
4)The User will enter their credit card information	The User will enter their credit card information example "4242 4242 4242 4242" and will click the "Pay" button
5) The User will be taken to the thank you page	The User will be taken to the thank you page and then will be allowed to select the "take me home" button

4) Donate without having an account

1.)User Navigates to Donate tab	The User will select the "Donate" tile on the home screen of our page

2)The User will enter their credit card information	The User will enter their credit card information example "4242 4242 4242 4242" and will click the "Pay" button
3) The User will be taken to the thank you page	The User will be taken to the thank you page and then will be allowed to select the "take me home" button

5) View Account page

a) Change the information of the user
b) View previously selected tiles

Precondition: The user is logged into the system	
1.)User Navigates to the Account tab	The User will select the "Account" tile on the home screen of our page
2) User Enters their new account information	The User will enter their updated information for the fields: Name, Email, Address, confirming their

	address, and their zip. Then the user will hit "submit"
3)The User will select "View my Tiles"	The User will see all of our past tiles that they have and the date that they have purchased the tiles appear

6) View about page

1.)User Navigates to About tab	The User will select the "About" tile on the home screen of our page
2.) User views about page	The User will see the about page and all of the information on there

Admin Panel:

1) View Crud for Users
    a) View Records
    b) Page through records
    c) Delete Records
    d) Search records
        i) Clear search
    e) Add new records
        i) Validation for creating new records
    f) Export to Excel
    g) Edit Records
        i) Validation for editing Records

Precondition: The user is logged into the system	
The User will navigate to the "Users" tab	The User will see the Users crud table appear
The User will be able to delete a record	The User will go to the actions column and click the trash can icon. Once this action is completed the User will not see that record on the screen anymore

The User will be able to search records	The User will go to a search bar and select a field they want to search for. Once this is selected they will enter their search criteria and hit the search button. Their query of records that match the criteria will appear.
The User will be able to clear records	On the search bar row, the user will click Reset search and their query including their criteria will disappear and bring them back to their original crud table
The User will be able to add a new record	The user will click the "Add new" record and fill out all of the criteria. Once the user clicks save the new record will be in displayed in the crud table by doing a search function (See above)
The User will be able to edit a record	The User will select a record from the CRUD table and under the actions column click the edit icon which is shown as a pen symbol on a piece of paper. The User will edit the respective criteria and then hit save. This new information will be updated for the record
The User shall be able to export the records to excel	The User will click "export to excel" once they have clicked this the downloading of the excel document shall start

2) View Crud for Games

- h) View Records
- i) Page through records
- j) Search records
  - i) Clear search
- k) Add new records
  - i) Validation for creating new records
- l) Export to Excel

m) Create a default game board
n) Create random game board
o) Select Organizations
p) Select winning Tile
q) Delete Records
r) Edit Records
    i) Validation for editing Records

Precondition: The user is logged into the system	
The User will navigate to the "Games" tab	The User will see the Users crud table appear
The User will be able to delete a record	The User will go to the actions column and click the trash can icon. Once this action is completed the User will not see that record on the screen anymore
The User will be able to search records	The User will go to a search bar and select a field they want to search for. Once this is selected they will enter their search criteria and hit the search button. Their query of records that match the criteria will appear.
The User will be able to clear records	On the search bar row, the user will click Reset search and their query including their criteria will disappear and bring them back to their original crud table
The User will be able to add a new record	The user will click the "Add new" record and fill out all of the criteria. Once the user clicks save the new record will be in displayed in the crud table by doing a search function (See above)
The User will be able to edit a record	The User will select a record from the crud table and under the actions column click the edit icon which is shown as a pen symbol on a piece of

	paper. The User will edit the respective criteria and then hit save. This new information will be updated for the record

The User shall be able to export the records to excel	The User will click "export to excel" once they have clicked this the downloading of the excel document shall start
The User shall be able to create a default game board	The User will click edit game board and select their tiles
The User shall be able to create a random game board	The User will click edit game board and select randomize game board. Once this happens the tiles will get randomized. The user will hit submit and then be redirected to the crud table.
The user will be able to select organization(s)	The User will click "Edit organization List" under the actions column and then be redirected to a screen. From there the user will select their organizations(s) and hit submit. The user will then be redirected to the crud table
The User will be able to select a winning tile	The User will click "Select winning tile" under the actions column and then be redirected to a screen. From there the user enters the tile they want to be selected as the winning tile and hit submit. Then a confirmation popup will appear where the user will hit confirm and then be redirected to the crud table

3)

View Crud for Organizations

s) View Records
t) Page through records
u) Search records
    i) Clear search

v) Add new records
    i) Validation for creating new records
w) Export to Excel
x) Delete Records
y) Edit Records
    i) Validation for editing Records

Precondition: The user is logged into the system	
The Organizations will navigate to the "Organizations" tab	The User will see the Organizations crud table appear
The User will be able to delete a record	The User will go to the actions column and click the trash can icon. Once this action is completed the User will not see that record on the screen anymore
The User will be able to search records	The User will go to a search bar and select a field they want to search for. Once this is selected they will enter their search criteria and hit the search button. Their query of records that match the criteria will appear.
The User will be able to clear records	On the search bar row, the user will click Reset search and their query including their criteria will disappear and bring them back to their original crud table
The User will be able to add a new record	The user will click the "Add new" record and fill out all of the criteria. Once the user clicks save the new record will be in displayed in the crud table by doing a search function (See above)
The User will be able to edit a record	The User will select a record from the crud table and under the actions column click the edit icon which is shown as a pen symbol on a piece of paper. The User will edit the respective criteria and then hit save. This new

	information will be updated for the record
The User shall be able to export the records to excel	The User will click "export to excel" once they have clicked this the downloading of the excel document shall start

4)

View Crud for Donation

z) View Records

aa) Page through records

bb) Search records

    i) Clear search

cc) Add new records

    i) Validation for creating new records

dd) Export to Excel

Precondition: The user is logged into the system	
The User will navigate to the "Donation" tab	The User will see the Donations crud table appear
The User will be able to search records	The User will go to a search bar and select a field they want to search for. Once this is selected they will enter their search criteria and hit the search button. Their query of records that match the criteria will appear.
The User will be able to clear records	On the search bar row the user will click Reset search and their query including their criteria will disappear and bring them back to their original crud table
The User shall be able to export the records to excel	The User will click "export to excel" once they have clicked this the downloading of the excel document shall start

5)

View Crud for Admins

ee) View Records

ff) Page through records

gg) Search records

    i) Clear search

hh) Add new records

    i) Validation for creating new records

ii) Export to Excel

jj) Delete Records

kk) Edit Records

    i) Validation for editing Records

Precondition: The user is logged into the system	
The User will navigate to the "Admins" tab	The user will see the Admins crud table appear
The User will be able to delete a record	The User will go to the actions column and click the trash can icon. Once this action is completed the User will not see that record on the screen anymore
The User will be able to search records	The User will go to a search bar and select a field they want to search for. Once this is selected they will enter their search criteria and hit the search button. Their query of records that match the criteria will appear.
The User will be able to clear records	On the search bar row, the user will click Reset search and their query including their criteria will disappear and bring them back to their original crud table
The User will be able to add a new record	The user will click the "Add new" record and fill out all of the criteria. Once the user clicks save the new record will be in displayed in the crud table by doing a search function (See above)

The User will be able to edit a record	The User will select a record from the crud table and under the actions column click the edit icon which is shown as a pen symbol on a piece of paper. The User will edit the respective criteria and then hit save. This new information will be updated for the record
The User shall be able to export the records to excel	The User will click “export to excel” once they have clicked this the downloading of the excel document shall start

6) Logging in to Admin Panel

i) Validation for logging in

The User Will login to the admin panel	The User will type in their credentials and then be redirected to the admin panel

6. Project and Risk Management

6.1 Task Decomposition & Roles and Responsibilities

The objective of this project is to develop a cross-platform web application for the CowChips4Charity event during collegiate and professional sporting events. The web application will provide a way for attendees of a sporting event to interface with the Cow Chip Bingo game sponsored by CowChips4Charity. This application will consist of a user being able to login to their account. Users will also be able to access the app as a guest. The application will let users pick what event they're attending and fill out all of their personal information (phone number, address, team, credit card). This application will let users provide credit card information to bet on a set of squares on the bingo board through a donation to the client's Boo Radley Foundation. Once the Cow Chip Bingo activity is finished, the admin will be able to select the winning square and users will be notified via email.

Frontend Tasks

1. Create axios base instance for connection to the backend
    a. Create an axios base instance that automatically sets headers and URL
    b. Add base axios instance
2. Login Page
    a. Create a login page for users to login
    b. Send HTTP request to the backend and authorize the request
3. Home Page
    a. Create the Home Page and add an additional standalone donation button
    b. Link all of the pages to the home page via buttons
    c. Ensure that some features are hidden and only able to be viewed after a user logins
4. Donation Page
    a. Create a donation page for users to be able to donate money
    b. Integrate stripe API for credit card processing
    c. Ensure Stripe integration is secure and encrypted
    d. We need the login page submit button to send the data to stripe and then receive the callback.
5. Link donation page to home page
    a. When the user clicks on the donation button on the home page, they will be rerouted to the donation page.
    b. Ensure that all of the user's credentials are saved via Stripe

6. Tile selection page
    a. When the user goes to the tile selection page, then the page should get the layout of the tiles for the specific game they have selected, display the tiles, and allow the user to select the tiles, once the user has selected the tiles they should be able to continue on to the donation page.
    b. Integrate Tiles select page into stepper so all selected tiles by the user are saved
7. Game Selection page
    a. When a user wants to play from the homepage they will be redirected to the game selection page, on this page the application should get the list of currently active games, display that list, and allow them to select which game they would like to play, the selection should be stored in the Vuex store. Then the user will be redirected to the tile selection page.
    b. Ensure that only present or future games are able to be selected
8. My tiles page
    a. A user should be able to see the previous tiles that they selected
    b. When a user clicks the "my tiles" button on the homepage they should be taken to the "my tiles" page. The page will show a breakdown of their previous tiles listed by game ordered in chronological order. If the game is already over then the winning tile number should be shown next to the game, with any winning tiles that the user selected highlighted in some way. If a user just made a donation without actually playing the game, then that should be shown differently in some way.
    c. Integrate the backend so that the user can see the past donation amounts
9. User Settings Page
    a. User should be able to update their account information
    b. When the user clicks the account button on the homepage, then their account info should fill in and be editable.
    c. The backend will receive the changes and update the user data
10. Registration Page
    a. Create the registration page in parallel to the requirements documentation

    b. The users account will be created and they will automatically be logged in

11. Create a layout for all of the pages to be displayed and unified
    a. Define a default layout that will hold all of the views inside of it
    b. Ensure all pages are displayed and can be navigated to
12. Organization select
    a. As a player of cowchips4charity I need to select which "organization" I am supporting so that Ken can properly allocate funds.
    b. Retrieve organizations that have active games. From the backend route
    c. Display all of the organizations in a list

Backend Tasks

1. Create backed framework
    a. Create and test SRC DIR
2. Setup unit test with watch
    a. Using JEST
    b. Chai HTTP
3. Setup Travis CI
    a. Travis CI will be used for all testing
    b. Needs to pass Travis CI for the issue to be closed
4. Define the game routes
    a. Allow admins to create new games
    b. All games have a start and end time
    c. Create a route that allows the frontend to get the currently active games. The format should be a JSON array of games objects.
5. Integrate PM2 to serve the application
    a. For a production environment where the application is getting hit many times, it is important that we serve the application via PM2. In order to set this up, we need to set up the Heroku environment for PM2 and make sure that we can successfully retrieve the logs.

6. Define game model for tiles
    a. In the games collection, each game has their bingo tiles laid out in a random way that will be entered by the user on the admin panel
    b. The tile each need to be saved as an array to see which ones were selected and which one is the winning tile
7. Define donation schema
    a. Create a donation schema for the user's donation array to be stored
    b. Store donations separately and make it relational
8. Create CRUD routes for games
    a. Create CRUD routes for games so getALL, getOne, update, and delete
    b. Create a user route for a user to get a game record by ID
9. Create organization functionalities
    a. Create organization routes, controller, model, test and validation
10. Automated test to ensure the cleanup
    a. Setup automated tests to see what works and what fails
11. Donations READ and query Routes
    a. We need a READ route in order to get all donations and get donations by ID.
    b. Will return only the #th page from the results of the query. We need to define the number of items per page
12. Pagination
    a. We want to be able to select "pages" of items from the database.
    b. Should make a page method that will find objects by pages, or if too hard just use pagination anywhere we select objects from the DB.
13. Create Data models and update schemas
    a. Board must be square according to the data model
    b. Ensure only one donation per tile
    c. Override the end time of the game once the admin selects the winning tile
14. Amazon SES integration
    a. Create AWS account and integrate AWS SDK

    b. Create HTML emails for AWS to use

15. Route for retrieving organizations for active games
    a. Create a route to only show games that are currently active and the game is not yet over
16. Backend response overhaul
    a. Update the backend to fetch the appropriate object when an object id is nested in another object and replace the id with the object itself before sending back the message.
17. Create populate optional flag
    a. Create a URL parameter that we can attach to GET requests that will replace the ObjectID
18. Create a winning user route
    a. Make route that will find the winner for each game based on the selected winning tile

Admin Panel Tasks

1. Create sidebar
    a. Configure sidebar with the general aspects of the panel
2. User CRUD table implementation
    a. On the sidebar, there should be an item called "users"
    b. That should take the admin to a users CRUD table
    c. User information should be displayed in a b-table
    d. Each user should have an edit action button that opens up a page that is a form allowing the admin to edit the user's information
3. Hash routing and implementation history
    a. Switch the application from using hash routing to be using history routing
    b. Currently several bugs with using hash routing
4. Create excel export for tables
    a. The excel export button should be able to create a downloadable excel spreadsheet with the current table data with any filters or sorting applied.

5. Login
    a. Integrate login for the sidebars and ensure are routes are protected
6. Implement admin permissions
    a. Have permissions limit sidebar items
    b. Restrict specific routes
7. Dashboard define teams
    a. Pick which data to display
    b. Create graphs for visual representation for the user activity
8. Generalize CRUD tables for reuse
    a. Make the users crud general so that multiple CRUD screens can use it and just pass in the necessary data as props and slots
    b. Slots should be the update form and create form fields
9. Make usage of HTTPS
    a. When a user goes to the production version of the application we want them to ALWAYS be using https
10. Login error handling
    a. Create error pages for login
11. Games organization select
    a. As an admin of the admin panel, I need to be able to add organizations to a game's organizations array.
    b. Add an action button on the CRUD table
    c. Get a list of all of the organizations from the backend
12. Game board creation
    a. As a user of the admin panel, I want to be able to edit the game board to configure where the tiles will display on the board. The possible numbers are 1-36 and there should be no duplicates.
13. Donations CRUD table
    a. Create a crud table to show the past donation data of the various games
14. Organizations CRUD table

        a. Create a crud table to show all of the active and past organizations for games

15. Winning tile select
    a. Once the winning tile is selected the game is over
    b. Ensure that the user that has the winning tile is recorded
16. Searching
    a. Create a drop-down box for criteria for searching
    b. Ensure that all of the crud tables are able to be searched
    c. The search criterion needs to be validated

Roles and responsibilities

Connor Rust - Backend Lead

Jack Boike - Admin Panel / Scribe

Ben Meeder - Frontend Lead

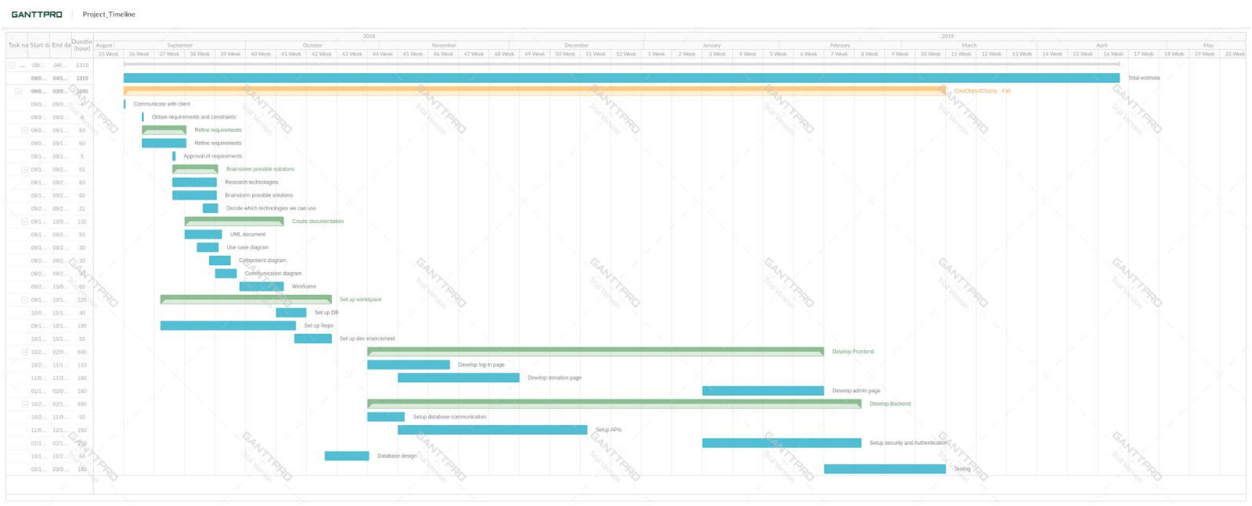
Kenneth Ho - Admin Panel / Meeting Scribe

Daniel Lev - Frontend / Business Lead (PM)

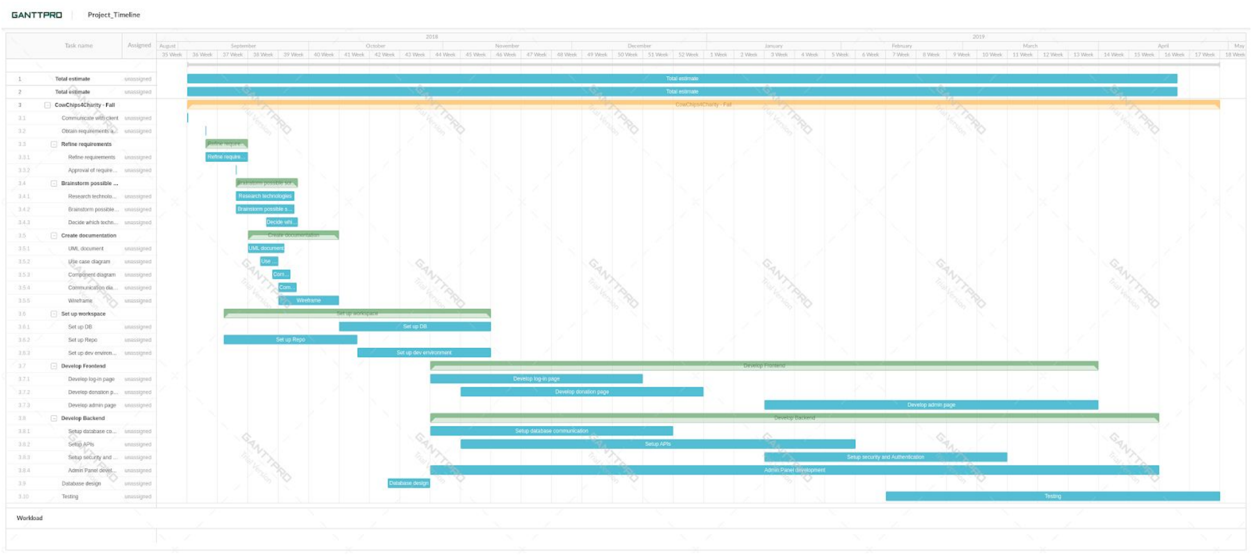
Alex Lev - Administrative / UI

6.2 Project Schedule - Gantt Chart (proposed vs. actual)

Proposed



Actual



GANTTPRO | Project_Timeline

	Task name	Assigned
1	**Total estimate**	unassigned
2	**Total estimate**	unassigned
3	**CowChips4Charity - Fall**	
3.1	Communicate with client	unassigned
3.2	Obtain requirements a...	unassigned
3.3	**Refine requirements**	
3.3.1	Refine requirements	unassigned
3.3.2	Approval of require...	unassigned
3.4	**Brainstorm possible ...**	
3.4.1	Research technolo...	unassigned
3.4.2	Brainstorm possible...	unassigned
3.4.3	Decide which techn...	unassigned
3.5	**Create documentation**	
3.5.1	UML document	unassigned
3.5.2	Use case diagram	unassigned
3.5.3	Component diagram	unassigned
3.5.4	Communication dia...	unassigned
3.5.5	Wireframe	unassigned
3.6	**Set up workspace**	
3.6.1	Set up DB	unassigned
3.6.2	Set up Repo	unassigned
3.6.3	Set up dev environ...	unassigned
3.7	**Develop Frontend**	
3.7.1	Develop log-in page	unassigned
3.7.2	Develop donation p...	unassigned
3.7.3	Develop admin page	unassigned
3.8	**Develop Backend**	
3.8.1	Setup database co...	unassigned
3.8.2	Setup APIs	unassigned
3.8.3	Setup security and ...	unassigned
3.8.4	Admin Panel devel...	unassigned
3.9	Database design	unassigned
3.10	Testing	unassigned

Workload

6.3 Risks and Mitigation: Potential (anticipated) vs. Actual (happened) and how they were mitigated

Potential Risks and mitigation

Some risks that the team has identified regarding our project process and development plan include:

- Project transfer after our team is rolled off of the engagement in May 2019
- Privacy laws that may prevent user data storage in some states
- Service reliability caused by network lag during sporting events
- Server host pricing as web application expands and matures
- Learning curve when utilizing new APIs
- Instructing admin users on how to execute admin capabilities

Constraints Considerations

One of the constraints that we will have is our budget. Our client wants us to make this project scalable because it will be taken over once we are done and he will try expanding the outreach to the users. We have to be cautious about our budget because our client's foundation is paying for this project.

Another non-technical requirement is the reliability of the product during the main uses of it, which will be during football games. The constraint for this is that wireless networks are often worse in the general vicinity of college football games. Therefore, our server reliability will be integral to the success of the project.

The last main non-technical requirement will be the usability of our page. Our client wants to have the quickest way for users to be able to donate with as little confusion as possible.

The standard protocols we followed while developing our project followed an agile protocol. We have multiple members of our team who are agile certified therefore we used their expertise to maintain the agile process. The agile process is taught in school in many of our classes, therefore, they meet IEEE and ABET criteria.

Actual Risks and mitigation

Actual Risk: Project transfer after our team is rolled off of the engagement in May 2019

Mitigation of respective actual risk: Have created thorough documentation outlining all components of the application, credentials needed, operational manual, transition statement of work, memorandum of understanding, and contact information. We have also been in talks with the professor to get another senior design team to take over the project in the Fall. Lastly, we have trained our client on how to operate our application and admin panel and gave him full permission to the entirety of the project.

Actual Risk: Learning curve when utilizing new APIs

Mitigation of respective actual risk: When learning new APIs our team ensure we all did our proper due diligence and training. We utilized various online training modules and platforms to become familiar with the new APIs we use. We also helped each other out to master the API technologies.

Actual Risk: Instructing admin users on how to execute admin capabilities

Mitigation of respective actual risk: We lead various training sessions and created thorough documentation to ensure that the admins know how all the features work. Finally, we gave the admin all of the credentials and rights before we transitioned off to see if they had any questions or needed additional training.

Actual Risk: Not being able to notify the correct winner

Mitigation of respective actual risk: We were first using Twilio to notify users if they won and collect their prize preferences and shipping information to ship the prizes. This was not working well since users can change their phone number and during the event cellular service does not always work well. Instead, we decided to use AWS's Simple Email Service to send emails to the winners of each game. AWS would send an email to the winner and collect their information and store it in the cloud so the admin can then send out a prize to them.

6.4 Lessons learned

Throughout the duration of our project, there were many lessons our team learned dealing with new technologies, leadership, and project management. This project and course prepared us to become productive members of industry upon graduation.

First off we learned how to manage a project that took the entire year to plan, develop, test, and transition. Due to the sheer size of the project, there were many lessons learned regarding that. The first takeaway when dealing with such a large project is that it is crucial to stay on task and updated. As a group, we knew it was necessary to create a structure and created several different ways to stay organized by utilizing various technologies. Although we tried to stay organized, there were many times that our GitHub was not up to date or some members would only update certain aspects of the project log that not everyone could see. We learned that is is necessary to update all of the logs and tasks and ensure everyone knows the progress of each task so all work could be expedited.

There were also many technical lessons that we learned throughout our project. We learned that it is not always the best idea to select technologies that are the most cutting edge. This is because many of these newer technologies do not have a large number of training resources and have a high learning curve. Many of the technologies we chose were extremely hard to learn and did not come with many training resources. Thus, we have realized that choosing technologies to utilize is integral to project success.

Lastly, our team learned a lot about leadership and teamwork. During this project, there were many times that unexpected situations came up and certain members had to step up and help. We learned that even though you're not assigned to a certain task if the team needs your help it is crucial to step up. We also learned how to communicate well as a team and leveraged different communication channels to ensure that everyone was up to date.

7. Conclusions

7.1 Closing remarks for the project

To conclude our project we have accomplished all of our client's requirements from when we first inherited the project. Following the implementation and design of the project, our team was able to take on more responsibility and found many more requirements. The requirements that we added were focused on the admin panel and having our client utilize the application and all of the data they receive from the application to their maximum potential.

Originally the client requested we make the application just for the user and have the client control it with a separate login as an admin. Our team decided (with the client's approval) that to optimize the client's needs we would create a separate application known as the admin panel. The goal of the admin panel was for our client to see all of the backend data of the application along with being able to control the "Front-end" of the system. The client was very pleased we created this easy to use Admin Panel with features like exporting all of the user data and information from the database into excel sheets. The client will be able to utilize this admin panel to keep improving their foundation and maximize the revenue.

Throughout this project, our team developed a close relationship with our client who is from the East Coast. Our client made the drive to Ames, Iowa numerous times to meet with us and see our progress. When the client met with us, he was always ecstatic to see our work and progress. The team was generally enthused to work on this project throughout the year.

We are happy we were able to develop a fully working system and even add on features that the client didn't originally request. We have learned a lot about development and project management throughout this process, along with client relations. Going forward our team is excited to see our application being used in real life events, and we cannot wait to see the successes of the Boo-Radley Foundation.

7.2 Future Work

During our closing out meeting with the client, we discussed the future of the application and his organization from a software perspective. The client realizes that the Cow Chip Bingo event is very expensive and that it has to occur during sporting events, which is a constraint. Therefore, moving forward the client thinks that another group can build onto this application by inserting components into our application which would allow the Cow Chip Bingo event to be simulated. This would mean that a physical cow and bingo board would no longer be required and the winning squares would be chosen by a random algorithm. We agreed with our client that this component would advance the application and also save our client a vast amount of money.

Along with adding the virtual Cow Chip Bingo component, we believe that the admin panel can be further extended. The data that the client has in the Admin Panel isn't being visualized currently within the logistics dashboard. To further expand the Admin Panel, a team could transform all of the data in the CRUD Tables to visual graphs that the client could see and track. Visual graphs would make it easier for the client to analyze the data in comparison to exporting the data into excel sheets and using Excel to analyze the information. We believe there is more work to be done, but our team also strongly believes we have done an exceptional job to build a well functioning application that will provide the client with all of his needs.

8. References

Our team has referred to past work and literature to help develop the design document and project plan. The extensive version of the past work and literature we referenced is in the project plan document. We were able to utilize resources from other classes that involve web development. Along with that, we made sure to utilize our client's websites and past work to figure out how he wanted the layout of the web application to be. All of the API's and other components of the application that are integral were found from research. To summarize we have been able to reference our past work and our client's past work to design our project.

To understand our project statement, our team participated in several meetings with our client to make sure we understand the statement of work and create the proper goals and initiatives. Our client provided us with several resources to develop an understanding of the CowChips4 Charity (http://cowchips4charity.com/) initiative and the Boo Radley Foundation (http://booradleyfoundation.org/). We also took reference from other nonprofits to understand how they create outreach and increase sponsorship. These references come from The Make-A-Wish Foundation (http://wish.org) and from the Special Olympics event of Polar bear plunge (http://soiowa.org).