CowChips4Charity

Design Document

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/

Revised: 12/2/18 - V2.0

Table of Contents

List of Figures

List of Tables

List of Symbols

None at this time.

List of Definitions

CowChips4Charity: A charitable game initiative that is based on bingo fundamentals of the Boo Radley Foundation to increase donations and outreach.

1. Introduction

1.1 Acknowledgement

Our team 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. Our team would also like to thank Professor Lotfi Ben Othmane for volunteering his time to be the faculty advisor.

1.2 Problem and Project 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. 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. Our team believe the development of a web application interface for this event will increase the amount of participants and resulting donations that the Boo Radley Foundation will receive.

Our team 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 via Facebook, Google, or as a guest. 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 text message and will then receive a prize in the mail.

1.3 Operational 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 the large amount of anticipated users and limited cellular coverage during sporting events, the web application needs to be quick and easy to use.

1.4 Intended Users and Uses

The team has identified two types of intended users for this application: regular users and admin users. The regular user is an individual who is accessing the application with the purpose of participating in the CowChips4Charity game. The regular user will be able to sign up for a regular account, select bingo square(s), and donate money to the Boo Radley Foundation. This user will need a simple and quick user interface that can be used from a computer or mobile device. Regular users will be able to access the application prior to and during the sporting event.

The admin user is an individual who works for the Boo Radley Foundation and is running the CowChips4Charity event. This user have access to information on all of the CowChips4Charity participants. The admin user will have the ability to send out texts blasts and select the winning square for each occurence of the CowChips4Charity event. We are anticipating that the admin user will not be someone with a software development background. Given this constraint, the application will need to be easy to use and maintain.

1.5 Assumptions and Limitations

Assumptions

- Users will have a cell phone with internet access
    - To receive text message and fill in the address form when the user's square wins
- Users will have a credit card
    - In order to complete square(s) donation
- Users will not be citizens of the European Union or residents of California
    - Due to the GDPR and CCPA privacy laws just passed
- Client will provide logo design and theme for application
- Admin users will not have deep software development knowledge
- Winning square selection and live stream will be handled by external party

Limitations

- The application cannot be used in California
    - Due to the CCPA privacy laws
- The application cannot be used by citizens of the European Union
    - Due to the GDPR privacy laws

1.6 Expected End Product and Deliverables

The deliverables of our project will be documentation and a cross platform web application. Documentation will be used by future developers and admins of the web application to maintain and improve the application over time. Documentation deliverables include: use case diagram, component diagram, communication diagram, scope and requirements document, local database setup instructions, and instructions on how to set up and start the application. A more extensive version of the schedule with other set deliverables can be found in the project plan document on our website. The deliverables will be shown in the gantt chart at the bottom of the project plan document.

Scope and Requirements Document (Delivery: 9/8/2018)

The Scope and Requirements Document was created after our initial meeting with the client. This document is to be used as a general guidance tool for the development of the CowChips4Charity web application. The document outlines the functional scope of the web application and how the product will be utilized in our client's Boo Radley Foundation. In addition, the document also specifies the usability requirements of the web application in order to fully accommodate for our client's needs. Once the client assumes full ownership of the web application and source code, this document will serve as a foundation for future maintenance and development.

Use Case Diagram (Delivery: 9/19/2018)

The use Case Diagram was created after the completion of the Scope and Requirements Document. This document describes how users will navigate through the flow of the application. The document identifies two different user types for the web application and their respective capabilities. It is essential that this document is correct and approved by our client as it is one of the building blocks that determines how the application will be built.

Component Diagram (Delivery: 10/4/2018)

The Component Diagram was created after the completion of the Use Case Diagram. The diagram analyzes the flow of the screens to determine what individual pieces or components will be needed. This document is used by the developers and project management to break down the application into individual user stories in order to efficiently and correctly allocate work.

Communication Diagram (Delivery: 10/4/2018)

The Communication Diagram is used to provide a visualization of how different components within the web application interact with each other. This document will be used as the main reference for developers when they develop and connect different components throughout the development lifecycle. Future developers of the CowChips4Charity application can reference this document when performing maintenance or creating updates for the app.

Local Database Setup Instructions (Delivery: 9/9/2018)

The Local Database Setup Instructions are a set of technical instructions to be used by developers who will be working on the application once our current engagement has concluded. During future development iterations of this application, it is important for developers to not clutter up the production database with false/test data. This instruction document will provide easily interpretable directions on how to setup a local database to prevent the previously mentioned issue.

Application Startup and Setup Instructions (Delivery: 5/1/2019)

The Application Startup and Setup Instructions will be delivered alongside our final application. This document will give step by step instructions to admins and future developers on how to setup and start the application correctly. This document can be useful for the cases when our client wishes to save money and turn off the application when it is not being used, or if the client decides to move to a different set of servers. The document will be simple enough for people not from a software development background to use.

Final Version Web Application (Delivery: 5/1/2019)

The web application deliverable will be a production scale web application capable of handling hundreds of users at any point in time. The application will be intuitive and easy to use, be aesthetically pleasing, and be secure. Additionally, the database will be secure and redundant.

2. Specifications and Analysis

2.1 Proposed Design

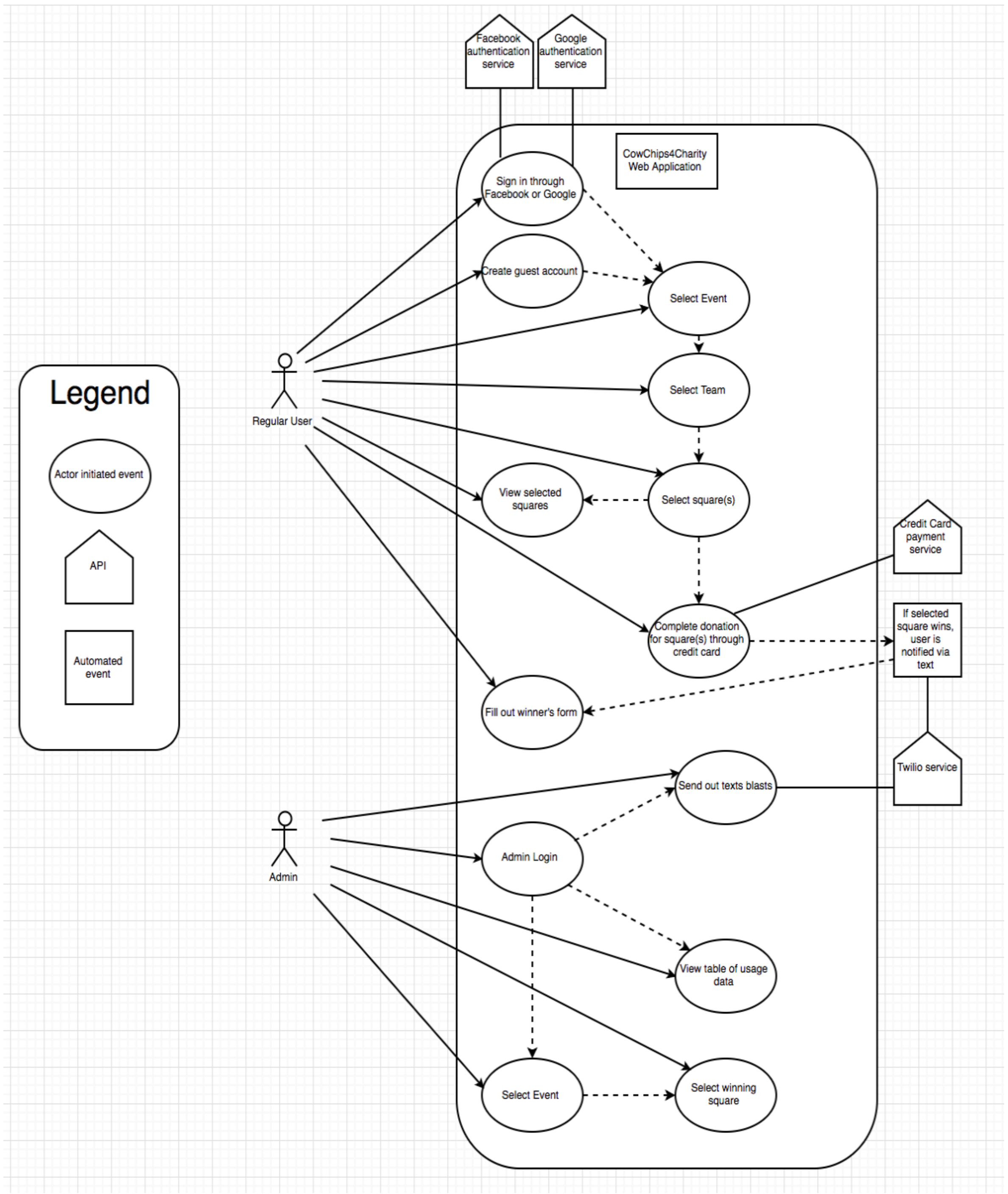


Figure 1: Proposed Use Case Diagram

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 seperate 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 text message blast. 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 database which gives information regarding the Users and donations. The last major component is the API's that are being used these are all individual components for example one API is the login through facebook which uses the facebook API. We also leverage an API to validate credit card and other online payment options so there are a variety of ways for users to be able to donate.

Table 1: Use Case Diagram Table - Use Case and Actors (Continued on next page)

Use Case	**Actor(s)**	**Input and Output**
Sign in through Facebook or Google	User	The user will input their credentials through a redirect from the respective social media site. The output will be the users credentials to the web app so they can sign in and have their info synced.
Create Guest account	User	The user will be able to create a guest account which takes input as their credentials. The output will be the webapp storing their information and getting a successful response from the database.
Select Event	User/Admin	The input will be the event they want to select and donate for. The output will be the confirmation of the event selected.
Select Team	User	The input will be the team they want to select and donate for. The output will be the confirmation of the team selected.

Use Case	**Actor(s)**	**Input and Output**
Select Squares	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 Text Blasts	Admin	The admin will be able to send text 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 a text is sent out to the respective users.
View table usage data	Admin	The admin will be able to see usage of different things. The different things can be something like donations per team, donations per event, how many users have signed up. The request will be what the admin wants to view and the output will be fulfilling that request.
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 recieving texts to fill out information so they can receive their prizes.

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

Adjacent System/Components	**Description**
Social Media authentication	This will be social media platforms like facebook,twitter, etc. The user will be able to login with their respective accounts and not have to make their account or remember their information. These API's will be connected to our web application.
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.
Twilio	The twilio service will send texts to users that the admin wishes to send texts 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.

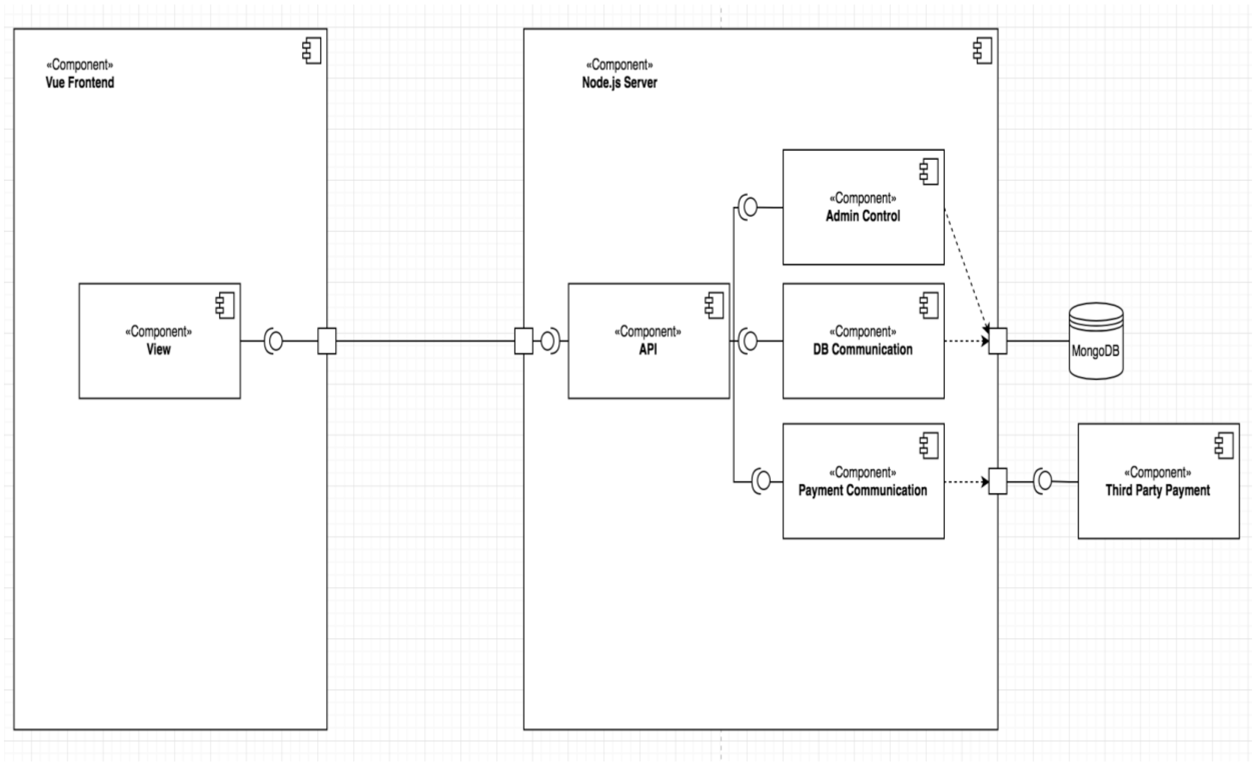


Figure 2: Proposed Component Diagram

Description of Component:

Admin Control component
The admin will have different functionalities and features than a regular User. The admin will leverage a Node JS server to accomplish all of the admin functionalities. The admin component will also communicate with the MongoDB. The reason the admin component will communicate with the MongoDB server is to be able to view usage data. The Admin component will also communicate with the database to select the winning user and notify them that they have won.

View component
The view component will house much of the frontend aspects of our project including the entire user interface of our web app. We will leverage vue and vuetify for all of our frontend design components including all for the User and Admin to ensure that we have a consistent design for all features and aspects of our project.

Payment Communication component
The payment communication component will be used by Users to confirm their donation as well as to purchase squares to play the Cow Chips Bingo game. The payment communication component will communicate with the third party payment component including stripe to validate all payments and ensure a secure transaction.

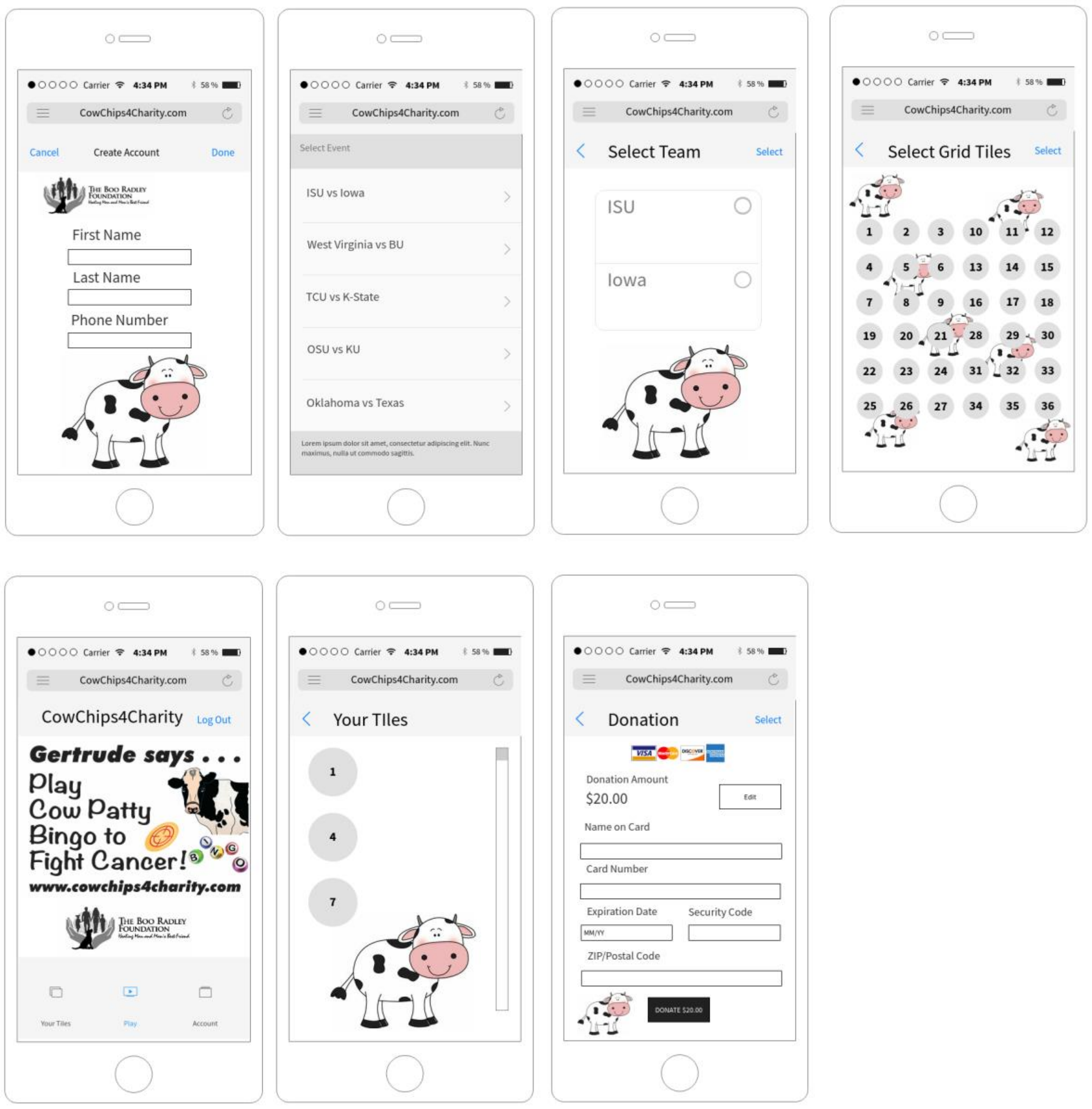


Figure 3: Wire Frames based off Use Case Diagram (Regular User)

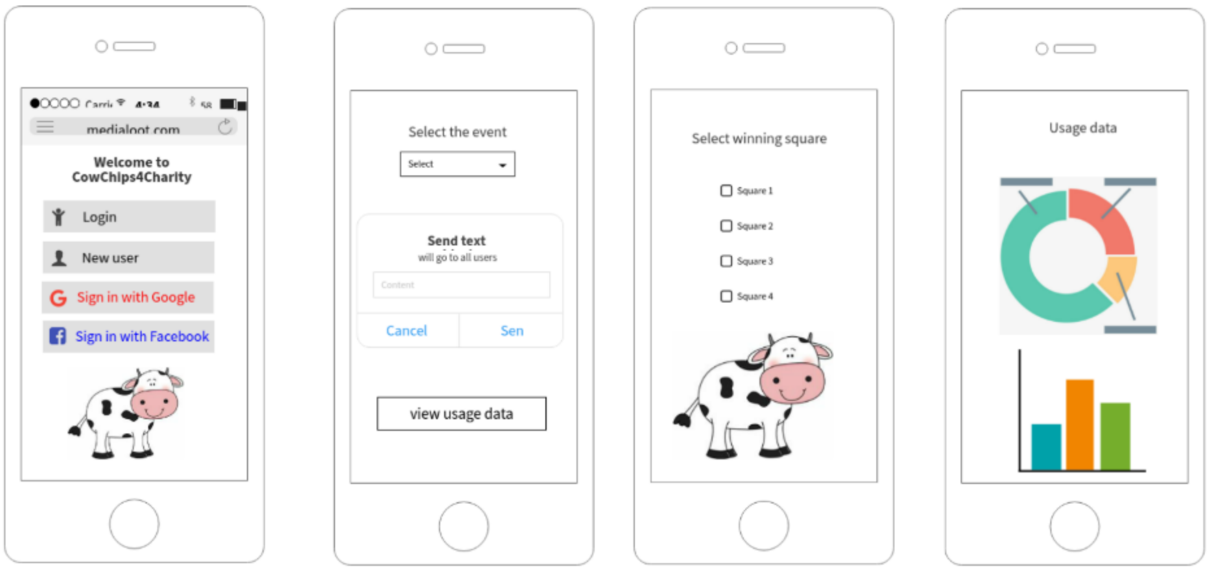


Figure 4: Wire Frames based off Use Case Diagram (Admin User)

Functional:

Front end:

1) Login via facebook, google, or guest
2) Credit card verification for payment
3) Be able to select multiple squares from a 6 x 6 grid
4) Admin page
    a) Select Winning square
    b) Add users,ban users, delete users
    c) View previous users from database with filtering

Back end:

1) Store square purchases to specific user profiles
    a) A user profile may have multiple square purchases
    b) Users acquire squares through donations
    c) Multiple users can own the same square
2) Store school selections to specific user profiles
    a) Record total amount of money donated from specific schools
    b) School info should be stored and easily accessed for future marketing activities
3) Twilio implementation for prize disbursement
    a) Winners will receive a Twilio text message form for address and contact information
    b) Prizes will be mailed out to participants and will NOT be monetary
4) Winning square(s) is selected through admin input

Non-Functional Requirements:

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 user shall be able to login through an existing Facebook or Google account in less than 15 seconds (performance)
8) The web app shall be able to calculate the entire cost of the donation once the user is "checking-out" within 3 seconds (performance)
9) The winning user shall receive a notification within 12 seconds from when the admin selects the winning square (performance)
10) The web application will be compliant with all policies and regulations set forth by the Boo Radley Foundation (operational)

2.2 Design Analysis

Our team has developed requirements and design documents that allows us to visualize how this application will look and perform. Previous to approval we shared these documents with many different parties including our: professor, colleagues, and our faculty advisor. Therefore, through many rounds of critiques and feedback our team was able to come up with an exceptional design. All of these requirements and design documents have been approved by our client and will give us a template for how the application will perform.

Our team has also created wireframes for the screens that the user and admin will see when they are using the application. The wireframes have been approved by our client and our team has moved into implementing the actual product. We have been using an agile approach to development where we continuously work and show our client the results after our two week sprints. Doing this type of agile development we are given time to change if anything is wrong and or misunderstood from our perspective.

Our project has very strong set of requirements and scope, we also have thorough documentation and wireframes that will aid us in making sure that our team meets all of the clients requirements. The gathering process for these requirements took quite a long time. We were able to gather these requirements from the approach of interviewing and prototyping from our client. Our client also used the approach from the users perspective to help us gather requirements. Our team was able to assist in providing our client information about the best practices of applications and make him aware of what will work the best. Through having open and exceptional communication with our client we were able to figure out all of his requirements both functional and non functional and we could tailor our documents directly to that.

Moving forward our team will update our wireframes based on the client feedback. Our team has already started development of the project. We hope to continue with this agile strategy where we develop a wireframe and show him the wireframe and explain the functionality and then go and work on the implementation of this wireframe. The reason for this is our client isn't very technically knowledge and UI is a very important thing for him. He would like to see the design and in regards of functionality he trusts the team and knows we will follow the requirements document and implement all of those. Therefore this technique of development has been successful for us thus far and we will continue to use this approach going forward.

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. For our example we were able to tailor or 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.

Assessment: Benefits and Constraints of Design

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. 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 best industry practices which we applied when making the design of this project. Our design also reflects the type of project management and implementation we will be doing which is going to be agile. The requirements and screens are separated so we can execute them through different types of sprints. After talking with our client 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 clients main concern. Our client isn't as concerned about a lot of the non functional requirements, but wants the application to be simple and to have all of the main things 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. We are utilizing API's also because this will save us time in development and testing. Also we will be transitioning this project when we are finished and the API's that we are using are continuously updated therefore the person who we transition will not have to worry about updating their application because the API will asynchronously do that for them. Some of the API's we are using are the facebook API for logging in, not only does this save the user time but it also is beneficial in terms of non functional requirements for security purposes. The same thing goes for our Stripe API which we are using for card processing, therefore our client isn't liable for having stand alone code that does credit card processing the API (Stripe) is liable and if anything goes wrong they will help solve the issue for our client. Another API that we used is twilio, this is for sending texts to our users, the reason we utilized this instead of making our own code is because for scalability and reliability purposes the testing is already done for us and conclusive that it works.

The benefit of our design is that it is simplistic, we didn't want to over complicate the design or the implementation. Our client has informed us that the person who is taking over the project isn't as heavily technical as us, therefore if we made the design very complicated and had a steep learning curve to pick up then the transition would be difficult or non successful. 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 of had more control of the functionalities, but no one would be able to update it in the future. Therefore I still believe 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 scalability and reliability of the application which is something that is important and not taught in class sufficiently. We have also been taught to consider 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.

Process details:

- Our design consists of two actors (regular user and admin)
- We leverage several APIs for logging in including Facebook and Gmail
- 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

3. Testing and Implementation

3.1 Interface Specifications

Our team will utilize 2 sets of API's: one set that the single page frontend web application will connect to, and one that the admin panel will connect to. The first API will be for the frontend single page application. The Endpoints that need to be defined are as follows:

- Payment Endpoints
    - POST /donate - This endpoint will allow for a user to make a payment. The endpoint will take credit card information, which will be sent by the server to stripe to complete the payment.
- User Endpoints
    - POST /login - Users will send their password and email and will receive a JWT token upon confirmation.
    - POST /register - Users will register with the required information through a form on the front-end. If the given email is not already in use, the new user account will be made and the user will be logged in.
- Events Endpoints
    - GET /events - This endpoint will return a list of all current and future events.
    - GET /event?id=<number> - For this endpoint, the client will send the id of the event they would like to view. This will return the event information.
- Admin Login
    - POST /admin/login - The admin group will use this endpoint to login to the Admin Panel.

The admin backend will need to have the following CRUD (Create Read Update Delete) endpoints. These endpoints will give the admin full control over the database for viewing, filtering, and altering database information as needed:

- Users CRUD
    - POST /admin/users - this endpoint will allow the admin to add new users.
    - GET /admin/users/:id - this endpoint will be used by the admin to retrieve user information, such as email, name, donations made, and other user attributes.
    - PUT /admin/users/:id - this endpoint will be used by the admin to update user information.
    - PATCH /admin/users/:id - this endpoint will be used by the admin to modify user information.
    - DELETE /admin/users/:id - this endpoint will be used by the admin to delete a user from the database.
- Events CRUD
    - POST /admin/events - this endpoint will allow the admin to add new events.
    - GET /admin/events/:id - this endpoint will be used by the admin to retrieve event information, such as time, location, and donation statistics.

- PUT /admin/events/:id - this endpoint will be used by the admin to update event information.
- PATCH /admin/events/:id - this endpoint will be used by the admin to modify event information.
- DELETE /admin/events/:id - this endpoint will be used by the admin to delete an event from the database.

3.2 Hardware and software

Jest - Javascript Automated Test Runner. We will use this to run JS 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 ran during the continuous integration process in order to verify that new features work correctly, and that there are no regressions. Another main advantage 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.

3.3 Functional Testing

The main functional testing our team will use will be unit testing, integration testing, and end to end system testing. These will manifest themselves in ways that are slightly different depending on the project.

Unit testing will consist of testing the basic functionality of the small individual methods of our program. Our unit tests will be extensive, with a goal of unit test code coverage of greater than 90% in our application, and thus there will be too many to list. Unit tests will be written for every new function that our team pushes to our source code, and will be added to the repository of tests. All of the unit tests will be run every time a team member pushes code to ensure that functionality has not been broken with these changes.

Integration testing will consist of testing how the individual components interact with each other. These tests are vital, as they make sure that the larger components work together as we expect. Some examples of these tests include: integrations with APIs, interactions between pages, routing, etc.. Unlike the unit tests, these tests will not be included in our CI/CD pipeline, but are expected to be performed by team members on a regular basis.

End to end testing will be performed when a large portion of the application is completed. This tests will ensure the functionality of the entire application that has been completed. They will be performed by hand, with a checklist of the important functionalities that need to be working correctly. The tests will ensure the correct functionality of the newly implemented parts of the app as well as the previously implemented sections. These tests include: user tests, live demos, etc...

For the front end our team will implement unit testing that mounts the components to a virtual DOM and then simulate actions and verify internal states of all components. Our team will also mock out the Axios HTTP client allowing us to verify that the correct calls are being made to the backend API.

For the back end our team will implement unit testing for individual components and methods that do not interact with the database. Additionally, our team will implement integration testing to test and verify that API calls are fully implemented to the spec. The combination of these tests will make it so that our team has high confidence in the API's correctness.

3.4 Non-Functional Testing

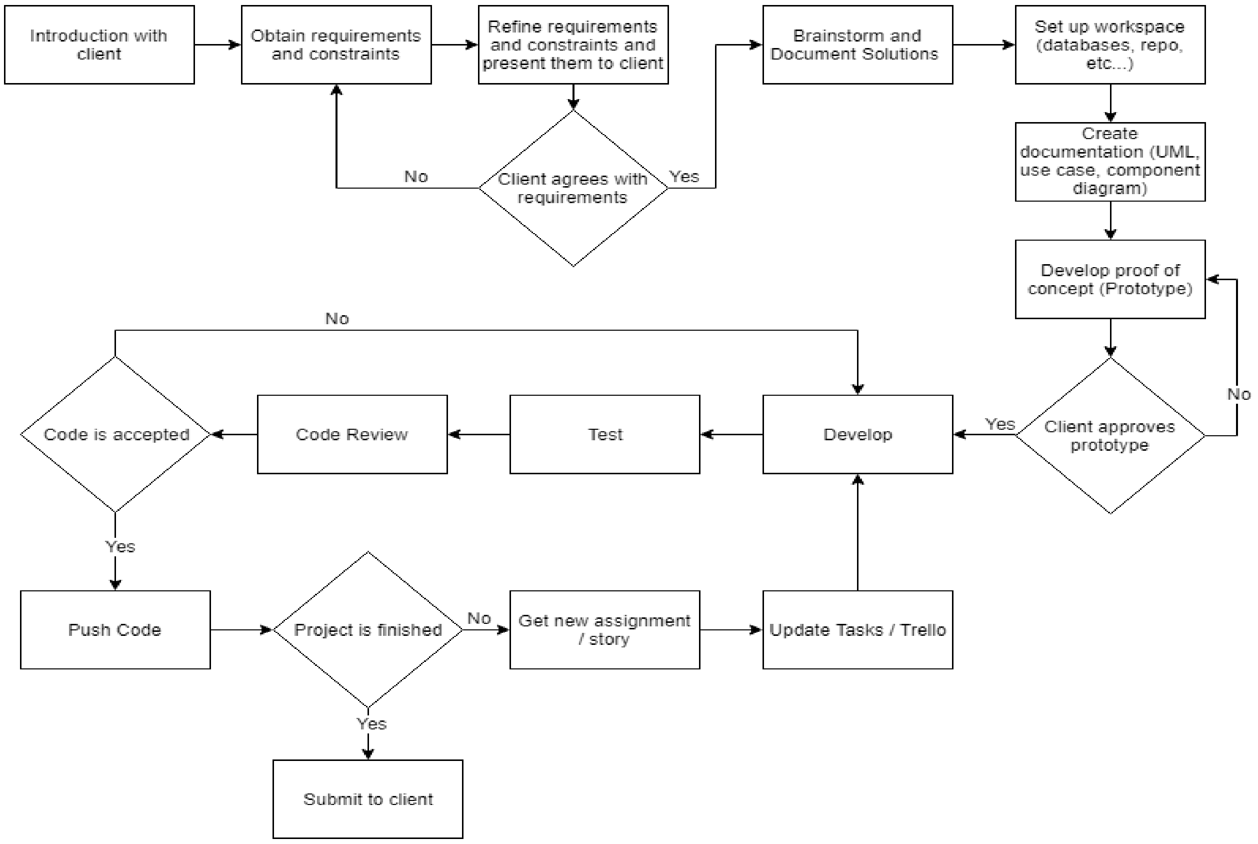
Our team will be performing two different types of non-functional testing, integration system testing (IST) and user acceptance testing (UAT). While performing IST our team will test all of the non-functional and functional requirements. Some of these tests include: the ability to support a large number of users at the same time, the ability to have 100% uptime during football games, etc.. We will be performing tests to measure the time it takes for a user to load the website, how long it takes to complete a payment, as well as reliability tests. These tests are vital to our application, as these non-functional requirements play an integral part in the usability of our application. We plan on having all of the functionalities and list of descriptions on how to run the functionalities. We will have our testers run through the scenarios and then report any bugs that come out of IST. Once the testers are done reporting the bugs and the functionalities from the bugs we will then have our developers make the fixes and iterate through the next part of IST. We will keep on running through IST until all of our bugs in the system are resolved.

When performing our UAT, the team will present the fully functioning application to our client. Some of the main concerns are reliability and performance during a football game when many users will utilize the application; therefore our team has decided to do load testing. Our client has stated that he would like us to test the application live at a pre-season football game before it goes live at the beginning of the football season. Additionally, we will be performing demos of the components that are completed to our client during our bi-weekly meeting with him. Similarly to the IST part of testing we will have all of the functionalities including NFR for the users to test. We will have a detailed document showing the expected results for each functionality and test data that they can input. If any of the output data or functions don't work properly we will report these bugs and work on fixing them. Furthermore we will continue iterating through UAT until all the bugs are worked out and the users report all of the functionalities are working.

3.5 Process

Our team will start by retrieving and refining the requirements from our client. After all requirements are gathered, we will focus on brainstorming ideas and designs for the product. Then, we will set up our databases, repos, and tools necessary for development. After creating diagrams and documentation, including a wireframe prototype, we will begin development on the product.

Our teams development cycle will include developing, automated testing, and code review. After completing a development task, our Project Manager will assign a new task.



Introduction with client
Obtain requirements and constraints
Refine requirements and constraints and present them to client
Brainstorm and Document Solutions
Set up workspace (databases, repo, etc...)
Client agrees with requirements
No
Yes
Create documentation (UML, use case, component diagram)
Develop proof of concept (Prototype)
No
Code is accepted
Code Review
Test
Develop
Yes
Client approves prototype
No
Yes
Push Code
Project is finished
No
Get new assignment / story
Update Tasks / Trello
Yes
Submit to client

Figure 5: Project Process Diagram

Table 3: Project Process Dictionary (Continued on next two 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 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*.

Step	**Description**
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 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.

Step	**Description**
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 Tasks/Trello	We will then update the tasks on trello to show that the old story was finished and that a developer will begin completing another task on our backlog.
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 development and maintenance of the system.

3.6 Implementation Issues and Challenges

One of the main issues that the project team is currently attempting to solve is our automated testing service, which is currently failing, although the project works. Our frontend lead is currently working on fixing this issue. Fortunately for out team, this issue came into light early on in the development process. One of the project team members discovered that although their implementation of the login screen was correct, the source code was not able to pass the CI/CD test integrated into our source control tool.

Another issue our team has encountered is related to framework knowledge. Going into this project, some of our team members did not have the necessary knowledge to complete their development work. To fix this issue, the front end developers made sure to read and watch tutorials for Vue.js while the back end developers made sure to read and watch tutorials related to the development of Node.js APIs. In addition, the frontend lead created a standard tutorial document for the rest of the project team to reference when dealing with Vue.js, issue creation, and issue addressing.

3.7 Results

Throughout the semester we have been doing agile design and showing our progress to our client and faculty advisor along the way. We have been developing features based off of the wireframes that we presented to our client and faculty advisor. The screens that we have done so far are the login screen, home screen, and the payment option which isn't currently hooked up to the backend. The main issues that we have had with our development to this point is being able to get the continuous development and continuous integration to work across all of our developers. To fix this issue our lead tester is working on figuring out a way that we will all be able to utilize our CD/CI. We have our backend completely working and have been able to test it using server code and applications like Postman API. In regards to the design processes we have been able to have all of our diagrams and also wireframes approved. This is very integral to continue working on the implementation of the project, because we have been basing the functionalities and design off of the documentation we have been currently utilizing.

To summarize we have been able to complete functionalities of the application already and have started integrating the backend and the frontend. We will continue to make progress on the application and we plan on having complete code coverage for the rest of the code as we have been testing all of our progress so far. Our team has all of the API's that we need and have made developer accounts for them (i.e. Facebook developer, coreUI, Stripe). We plan on continuing to work as an agile development team and report our work at the end of each sprint to our client.

4. Closing Material

4.1 Conclusion

Through completing a successful design thinking and creating design documents our team has prepared to move forward on implementation of the application. Focusing on requirements both functional and non-functional is integral for completing an application on time and with all of the functionalities. The functionality of our project has been documented using different mediums for example: requirements enumerating, use case diagram, and process diagram. Utilizing this documentation, we have been able to successfully gather requirements from our client and also start implementing the product. The process of design creation has been an agile process through communicating with the client and faculty advisor. The feedback from both of the respective parties has been crucial to finalize the design documents.

We also plan on creating a transition document that will allow our client to progress the application in the future and let another team of developers take it over. The team has been keeping exceptional documentation to all of our design and design decisions. The code for the application will be packaged and along with that will be a document how to run the code and explaining what different modules of the package do in regards do functionality. Overall our team has developed a plan to implement a fully functioning application and also allow it for our client to utilize it in the long term.

4.2 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 clients 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 clients past work to design our project.

Cited Sources:

"cowchips4charity." *Cow Chips 4 Charity*, The Boo Radley Foundation, 2018, cowchips4charity.com/.

"Facebook for Developers." *Facebook for Developers*, Facebook, 2018, developers.facebook.com/.

Johnson, Ken. "The Boo Radley Foundation - Healing Man and Man's Best Friend." *The Boo Radley Foundation - Healing Man and Man's Best Friend*, The Boo Radley Foundation, 2018, www.booradleyfoundation.org/.

Johnson, Ken. "Pre-Launch Demo (Final) by Booradleygames." *Itch.io*, Boo Radley, 2018, booradleygames.itch.io/pre-launch-demo-final?secret=wl657EJQhf0Erg5sv7OALj0.

4.3 Appendices

http://cowchips4charity.com/ : Our clients website which he currently has.

http://wish.org/ : A similar non-profit donation website