Cow Chips 4 Charity

FINAL REPORT

Group 8
Ken Johnson
Daniel Lev, Ben Meeder, Lotfi ben Othmane
Chloe Sabado - Frontend/Testing
Clint Lamar - Frontend/Animation
Helen Woldesenbet - Backend/Testing/ CI/CD
Jake Liebman - Frontend/Animation
Jared Schuckman - Backend/Analytics
Swecha Ghimire - Backend/Project Manager
Xander Apponi - Back End/ CI/CD/ Scribe
sdmay21-08@iastate.edu
https://sdmay21-08.sd.ece.iastate.edu/

Revised: 04/25/21 Version Final

Table of Contents

List of figures

List of Tables

1 Introduction

1.1 Acknowledgement

We would like to thank Ken Johnson, Daniel Lev, and Ben Meeder for their help with this project. Their feedback and guidance enabled us to create an excellent product and has helped us grow professionally. Our team also thanks Professor Lotfi Ben Othmane for volunteering his time to be the faculty advisor for this project.

1.2 Problem and Need Statement

The Boo Radley foundation is a non-profit organization created to promote research for diseases that are common to humans and animals. Being a non-profit organization, funding for the organization is acquired through donations and events. One of the most important events for funding is CowChips4Charity. CowChips4Charity is currently held as an in-person fundraising event, where participants select one of many squares in a pasture. If the cow roaming the field selects a participant's square, they win, and are given a prize. Any funds collected during this event contribute to The Boo Radley Foundation charity. Presently, while beloved as a state fair event, CowChips4Charity lacks the outreach it needs to acquire more funding for the charity. On top of this, the current need for drones (and other management equipment) ends up costing a sizable amount, lessening the total funds used for the charity.

To reach a wider audience and cut down on unnecessary expenditure, we seek to create a fully functional digital version of CowChips4Charity. To create this digital version, we plan on expanding the existing codebase with features such as an animated UI and accessible menus for a wider audience, as well as running a web service with virtually non-existent maintenance costs.

1.3 Operational Environment

Our application will be run on the cloud, so there are no physical risks. While there are still possible outages, they will be minimal due to the use of a major company's servers. As for the client application, little processing power will be needed, as the game itself can easily be handled by modern smartphones. The application will be used at large sporting events where there is cellular connectivity. Due to the use at sporting events our application will need to be reliable and easy to use.

2 Requirements / Specification

2.1 Functional Requirements

For our application the functional requirements include a user being able to utilize the application easily without confusion, the application being able to scale to have many games for the user, users being able to play from both mobile and desktop and lastly administrators being able to view usage statistics from the site. The first major component of the application is what our team referred to as the frontend where 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 frontend.

2.2 Non-functional Requirements

Our application has three main areas of non-functional requirements. First, there is scalability. The software needs to be able to serve many users quickly. With many sporting stadiums able to seat tens of thousands of

people, our software must be able to scale up to meet that demand. The events only happen intermittently though, so we do not want to be paying for high server costs when they are not needed.

The second non-functional requirement is maintainability. The software should allow simple admin and user control features. For admins, the people administering the games may not have a technical background. The admin interface must be simple enough that a non-technical user can easily learn and use it. Yet, it must still have enough features to be able to properly administer the games. Game playing users will quickly lose interest if the UI is difficult to understand, so the flow of playing the game must be simple, and fun.

Thirdly, our application must have good performance. The software should not be bloated with unneeded data for it's API calls, and users must have a quick response time. Since the application is used in sporting arenas where cell service may not be strong, it is essential that our application is as quick and lean as possible.

2.3 Engineering Constraints

As a software project, there are not many constraints that we must operate within.There are a few constraints in which we must follow though. First, we must work within the existing tech stack and code base from previous years. Also, for financial expenses, we must not spend any additional money than is necessary. If money is needed for resources, we must gain specific approval from the project client.

2.4 Market / Literature Survey

Our Project is to be built on two previous years of senior design work. As such, there is a pre-existing website that will be built upon for this project. This website incorporates many of the necessary features to make the game 'CowChips4Charity' functional. For the frontend this includes a hosted website, login/verification, a CowChips board, and communication with the backend. The backend currently has a hosted website, administrator login/verification, database storage, and communication with the frontend. To make all of this functionality possible, the website currently uses technologies such as Javascript, Vue.js, Node.js, MongoDB, CoreUI, and Heroku. Combined, these technologies offer a robust modern system, allowing for a good looking (and effective) frontend, while also simplifying the backend communication and data storage. The largest downfall of these technologies is our groups lack of knowledge surrounding them. In order to prevent this negative, our group has designated a period of time to understand the current technology in use (see section 2.4). Our additions to this project will largely revolve around an embedded animation that will play during an event, modernizing the user interface, and recording/displaying statistics that administrators can view after an event. Core technologies from previous senior design projects will be heavily utilized within our new features along with new technologies such as Google Analytics and Unity.

2.5 Deliverables

Due to use of an agile workflow, as well as a pre-existing codebase, design deliverables are not a core aspect of our project. Rather, our deliverables generally take the form of implementations of functional requirements.

Tech Stack Documentation

Information regarding the technologies to be used on the various ends of the project will be detailed here. New technologies that weren't used in past projects will particularly be used for animations on the client, data aggregation on the server, and card verifications on the server.

Game

The game will include a fully functional (playable) game, as well as an initial implementation of the visual style requested by the client. The playable game will allow control features from the administrator panel, simplistic menus.

Administrator Panel

The admin panel will include all required functions for administrators, with visuals to focus more on functional elements. Requirements, mainly revolving around data aggregation will be in place, with customization options in place for future testing.

User Acceptance Testing

User acceptance testing began after completing all our stories and gaining approval from our client and mentors. We started user acceptance testing in house first with our client, mentors, and team. After initial user testing and fixing those reported bugs we moved onto real user testing with friends and family. We provided a beta test script that walked users through our application and expectations. With that test script users were able to report any bugs or comments at each step. This user acceptance testing will involve testing both usability and reliability. By recording the feedback gained from these testers, we will be able to determine any necessary features or fixes.

3 Project Plan

3.1 Task Decomposition

We had 4 main subtasks each with their own requirements:

Design a CowChips animation to be embedded into the website: This task can be further broken down into creating the environment for the animation, animating the environment to the specifications of the client, and embedding the animation at the correct place in the website. This task is dependent on the UI sending a signal to start the animation and the Results screen to distribute rewards effectively.

Update the administrator panel for hierarchical privileges and increased data: This task can be broken down into creating a user hierarchy, implementing more usage statistics, and redesigning the look of the existing screen for these new features. This task is also dependent on the UI as well as the new data analytics framework.

Modernize the UI for cleanness and conciseness: This task can be broken down into updating the start screen UI, updating the admin panel UI and updating the result screen UI. This task is dependent on all of the existing elements as each new and old feature will need to be accessible to certain tiers of users.

Create a framework for data analytics: This task can be broken down into gathering and storing appropriate data, Analyzing the newly grabbed data, and displaying the data to the appropriate users. This task is dependent on the UI, the admin panel and the results screen as all of these pages have worthwhile data to collect.

3.2 Risks And Risk Management/Mitigation

Design a CowChips animation to be embedded into the website: None of us have experience with animation. This presents a risk of meeting our deadlines, because we may have a larger learning curve than expected. Risk Factor: 0.4. We can buy pre-built models with animation rigging set up off sites for $100-$400 if we are unable to become competent in the required skills on time.

Update the administrator panel for hierarchical privileges and increased data: The only risk for this is the interdependencies of this with other parts of the project, such as the backend systems, and data analysis. We have competent programmers on our team who have built similar things before. Risk Factor: 0.1.

Modernize the UI for cleanness and conciseness: This task should be simple as the elements already exist, and just need to be updated. This will require competency in Vue.js framework, which none of us have past experience with. Risk Factor: 0.3

Create a framework for data analytics: Similarly, to the admin panel, this task has a heavy reliance on the backend data systems to function properly. Also, there is not much test data to create visualizations from and perform analysis. Risk Factor: 0.5. Our risk mitigation plan is to generate our own synthetic test data to perform analysis on.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Based on the task decomposition above, we have proposed the following milestones, metrics and evaluation criterias

Design a CowChips animation to be embedded into the website

This milestone is going to be measured by the usability metrics for effectiveness which will be calculated by completion rate. Meaning, how long a user will take to complete the task when navigating the animation that is embedded into the website. we will use 80% competition rate as our goal metrics. Which is the number of tasks completed divided by the total number of tasks undertaken.

Update the administrator panel for hierarchical privileges and increased data

For updating the administrator panel for hierarchical privileges and increasing data, we will be measuring the response time as metrics and evaluation. Since, this part of the project is the core backend system of the project metrics and evaluation will depend on overall efficiency and usability of the administrator panel. Which shouldn't be longer than 10 ms per request.

Modernize the UI for cleanness and conciseness

For modernizing the UI, we will be measuring this milestone by using usability metrics for the time it takes to finish a task. Meaning, how long a user takes to navigate to a specific dashboard or link. This can be calculated by subtracting the end time from the start time. Finally, we'll average out the data based on how long each task is supposed to take. We will aim for an 80% completion rate.

Create a framework for data analytics:

For data analytics, since our core functionalities are gathering and storing appropriate data, and analyzing the newly grabbed data, and displaying the data to the appropriate users. Our evaluation criteria for data analytics is that we are able to store and display 100% of the data we are analyzing. For instance we don't want the number of donations and amount of money generated from a specific game to be inaccurate or missing certain donations.

3.4 Project Timeline/Schedule

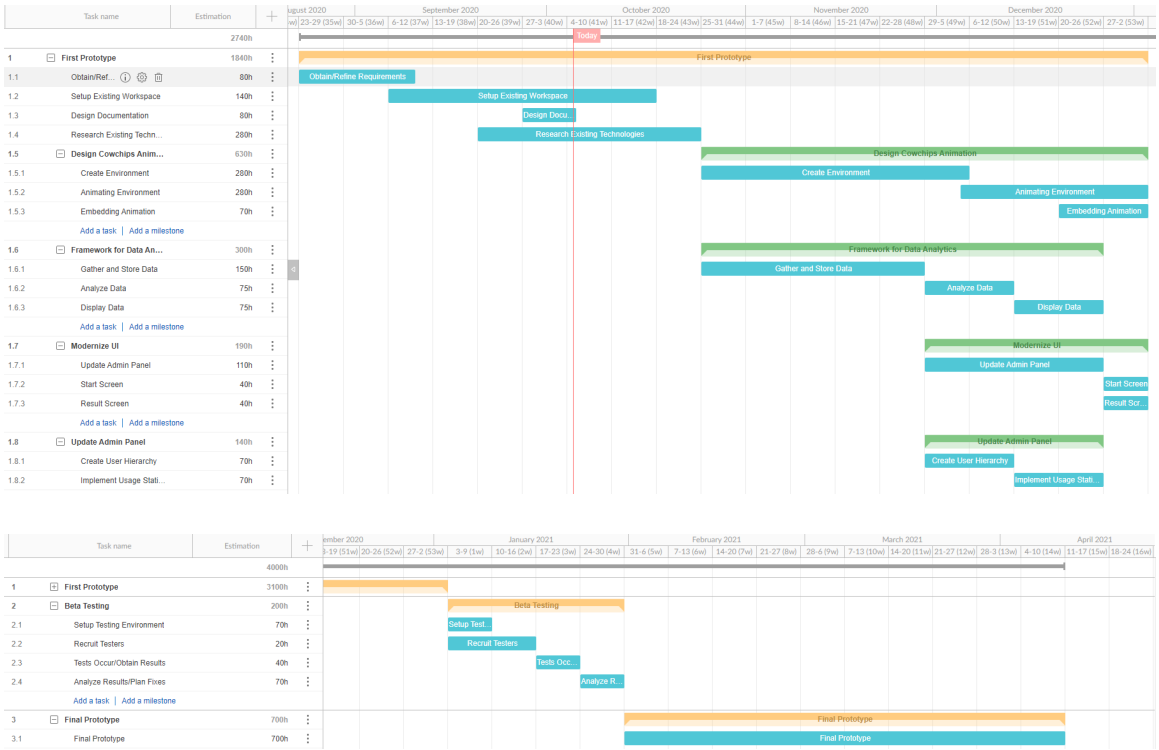

Figure 1: Proposed Project Timeline

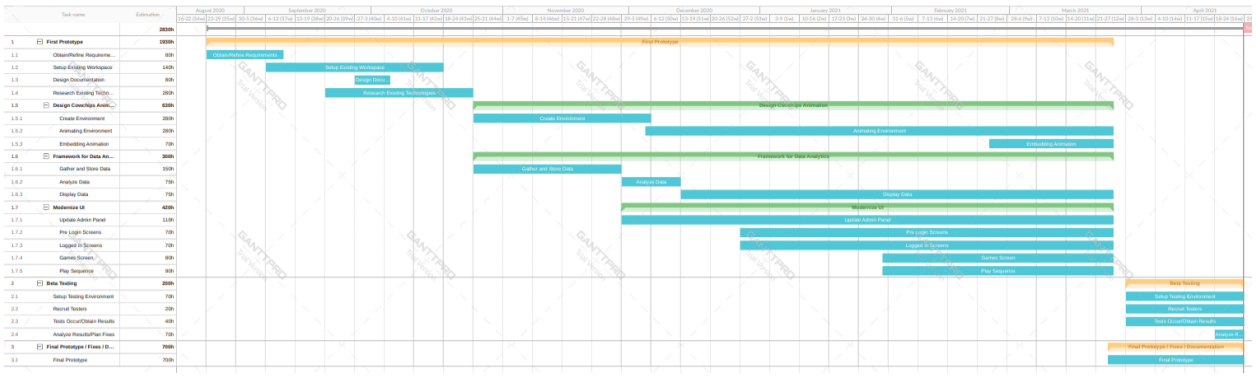

Figure 2: Actual Project Timeline

3.5 Project Tracking Procedures

Our group will be using Github and Jira to keep track of tasks. We will keep the repository in Github, we will keep track of stories and required tasks on a Github issue board, and lastly, we will use Miro during team meetings as a group whiteboard style. Progress will be assessed upon completed stories and successful implementation of features in the repository.

3.6 Other Resource Requirements

We do not require external hardware or other resources to complete our project.

3.7 Financial Requirements

We have financial requirements for the animation and for the hosting of the site. We realized we needed a professional 3D artist, so our client hired a professional animation team. The team was included in the interviewing process and project handoff. We also are using Heroku to host our site.

4 Design

4.1 Concept Description

The CowChips4Charity application will be a web application that will be used at sporting events in order to generate funds for the BooRadleyFoundation. An admin will create a game in the admin panel by entering a start time, end time, winning tile, name of game, and organizations participating in the game. The frontend of the application will have users register, login, or view more information about the application. After logging into the application the user will then be able to play a game that an admin has created. The user will then select an organization to support, select tiles, then donate an amount based on the number of tiles selected. After donating the user will be able to view the game within the games list. Once the end time of the game has passed the user will be able to watch an animation of the game and then enter their address if they selected the winning tile. The admin will then go back into the admin panel and view the winners and addresses and send them their prizes.

4.2 Concept Sketch / Block Diagram

The two diagrams below depict the functionality available to the users of the admin panel and the users of the frontend. Both show functionality available to users at certain stages of the application. And show how the user can flow to the next stage or a different stage of the application.

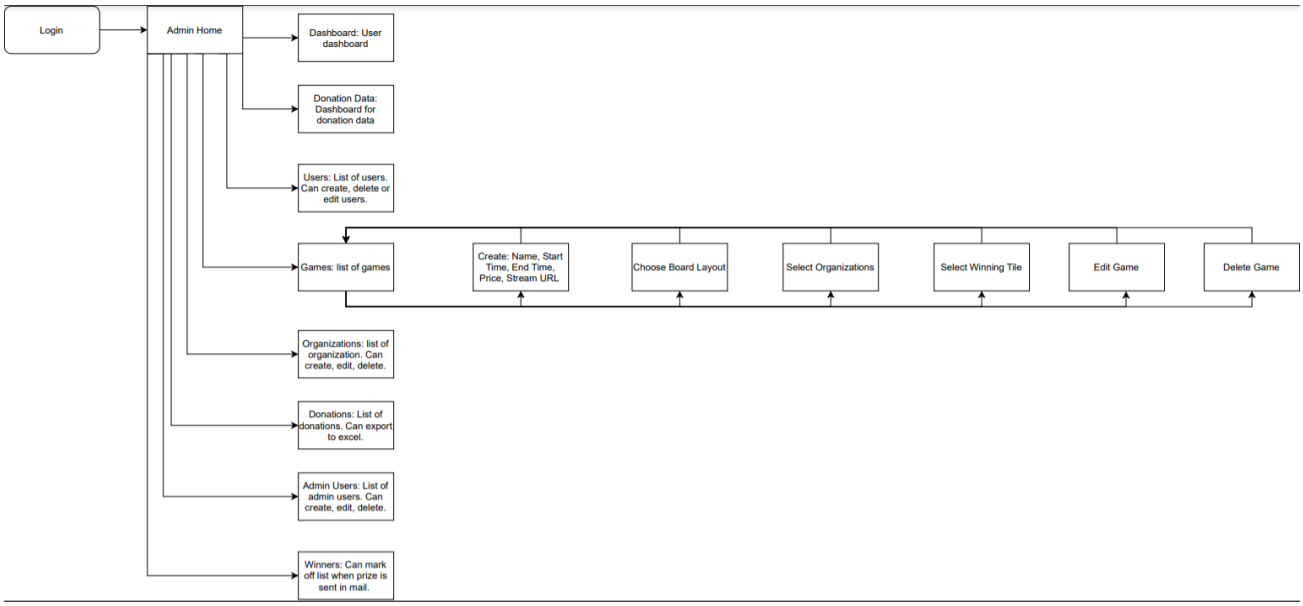

Figure 3: Block Diagram For Admin Panel

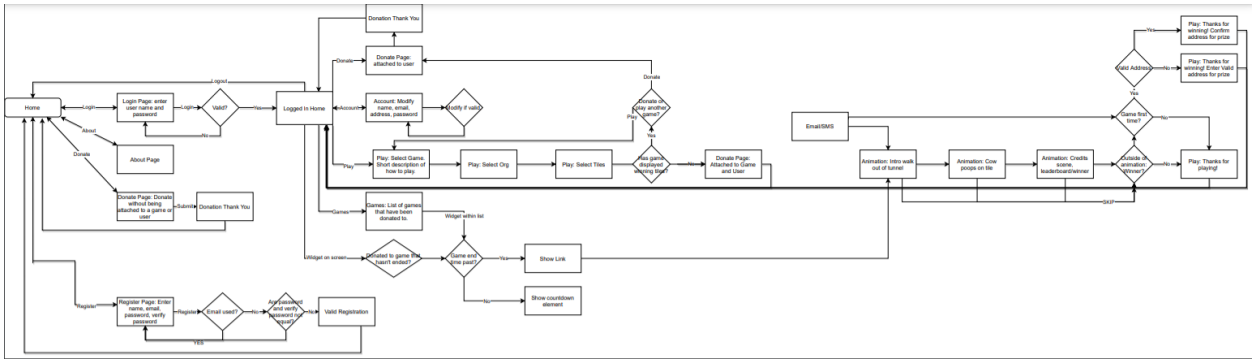

Figure 4: Block Diagram For Frontend

4.3 Proposed Design

One of the big challenges of this project regards the animation aspect. We have been in contact with a recent master graduate who is an expert in video game animation and design. Leveraging her experience and knowledge, we are working on a plan to make sure that the animation aspect of the project is done professionally and on time. Moving forward, we will be using Blender and Unity to create and embed the animations needed into our project.

Regarding the requirements of the project, our design will work well to satisfy all items listed in section 2. Since this project has already been in development for 2 years, we plan on leveraging the existing tools and technologies as much as possible, whilst working within our constraints to achieve the requirements. This means that the default design for most of the project is dependent on the frameworks that are already in place.

We have decided to change out the CI/CD tool from Travis CI to GitHub Actions, as it is cleaner and more fully functional. In addition, we have decided to move forward on the animation design with the tools previously mentioned.

We also decided to utilize google analytics in order to track user traffic within our site and engagement with different portions of the site. In order to track donations we will use the schemas being used to save donation information created by a previous project. We will access this information by creating a backend endpoint which we can query similar to the google analytics api so that it is easy to populate our graphs for donations.

4.4 Development Process

The process we use for this project is Agile, but more specifically SCRUM. After initially conferring with our client/mentors, we determined that a waterfall approach would lead to an overly loaded second semester. Along with that, having a pre-established codebase to build off of requires less overall design work, and more implementation. This fact works perfectly with SCRUM. By compiling a list of necessary components and breaking them into bi-weekly segments (including their respective planning), we can immediately make progress on this project, and continue it until the final day. Another benefit to using a SCRUM model is the risk aversion. Simplifying our tasks into smaller iterations allows us to verify each task with our clients/mentors at meetings, ensuring we are creating exactly what they want, while also ensuring the task is possible (in its current design) before dedicating a large amount of time to it. Overall, our decision to use SCRUM is to provide the flexibility a client-facing project such as this requires.

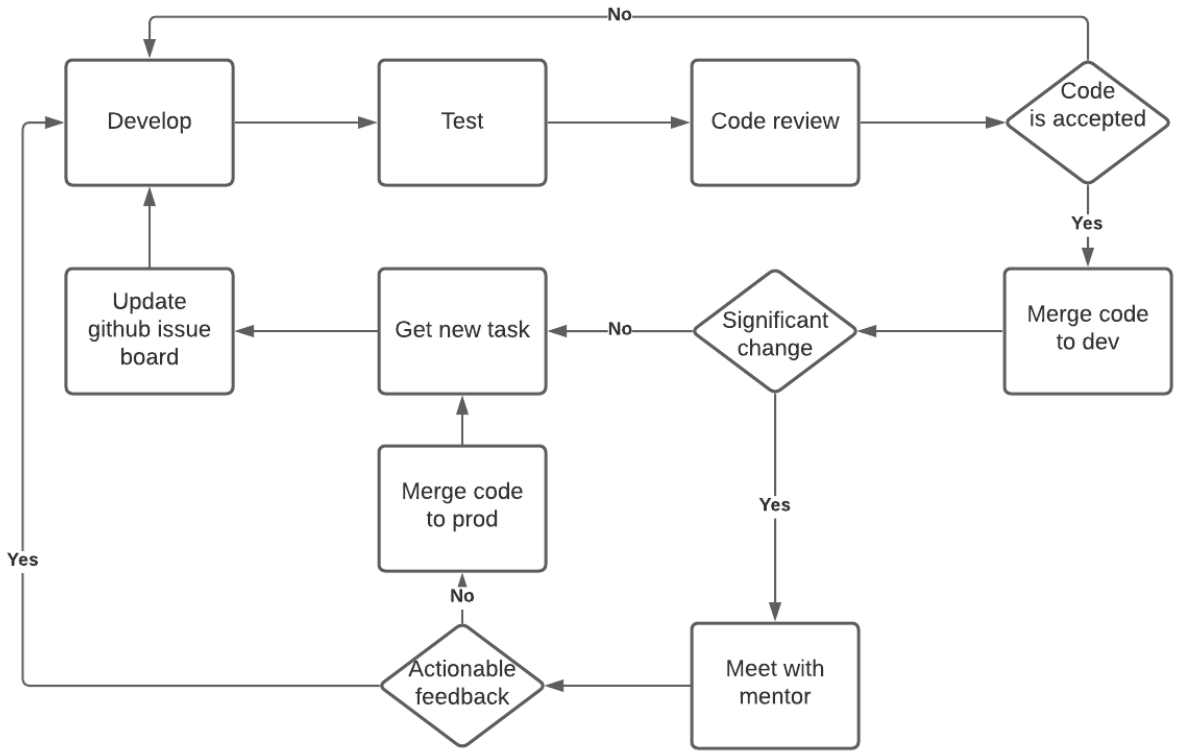

Figure 5: Development Cycle Diagram

4.5 Design Plan

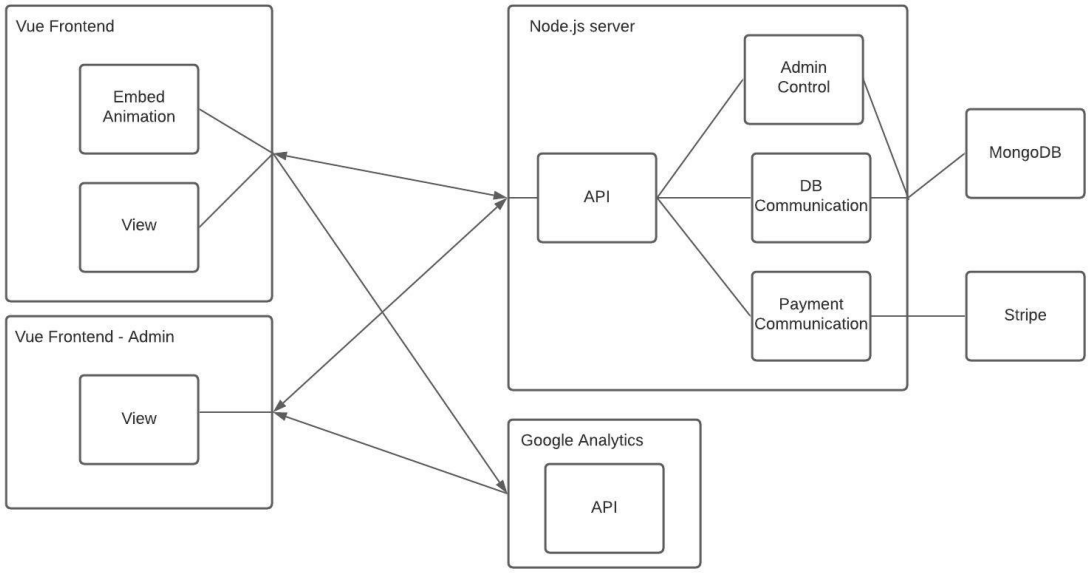

Figure 6: Component Diagram

Description of Components:

Vue Frontend

This is the frontend of the application which is located at https://www.cowchips4charity.com/ . This module is the interface of the app that is visible to the users. It leverages vue.js components for a consistent design and seamless experience due to the site being loaded once. This component consists of modules such as the embed animation which is where we will embed the animation for the cow chip bingo game. And view which is all of our frontend components. This will communicate with the node.js backend api in order to complete tasks such as login and retrieving and sending data about games and payment.

Vue Frontend Admin

This is the frontend of the application for administrators. This provides the interface in order to create games and choose winning squares. This also will provide graphs and information about the number of users, the amount of donations per game, total donations, what the users are interacting with on the site. This will communicate with the node.js backend in order to retrieve information about the games and donation values and will send data in order to update games. And this will communicate with the google analytics api in order to retrieve data about user traffic.

Node.js Backend

This is the backend component of the application which communicates with the admin and user frontend. This provides an api for the frontend to communicate with. The backend also takes care of payments through stripe. And communicates with the database in order to update information about games, users, and donations. It will also be able to retrieve data about donations for the admin panel.

4.6 Evolution of Design From 491

Our design from 491 has evolved considerably. As we dove deeper into the project we discovered new things that we had not thought about at the initial onset of the project. We thought of certain situations where a user could stay on the tile selection screen until after the animation had started and then pick the winning tile based on seeing what the winning tile is on another device. We added a stage in our flow seen in figure 4 which checks if the game time has passed before allowing a user to move onto the donation screen. We also needed to add a stage after the animation which ensured we received the winning address from winners as we were not prompting users to enter an address in order to create an account as we wanted to make the process to donate as easy as possible for a user. We also decided to include a games list that included a countdown to the endtime of a game versus having a countdown screen immediately after donating. This gives the user an easier experience to find where to access a game. There were also many small evolutions in design we made of which not all can be listed here. We also made

many iterations on the design and implementation from week to week based on feedback from our clients and mentors as we were utilizing an agile methodology and trying to receive as much consistent feedback as possible in order to provide the best end product for our client.

4.7 Security Concerns and Countermeasures

Our main security concerns are keeping user information secure and also ensuring secure payment. In order to keep user information secure we ensure passwords are hashed so that passwords can't be found by looking at the database. We use Stripe for payment so we keep payment secure by utilizing Stripe instead of implementing payment processing ourselves. Also no credit card information is stored in our database. This will ensure any user of our application will not have their credit card information stolen and card used for transactions they do not approve of.

5 Testing

5.1 Unit Testing

We test the basic functionality of small units of our software in isolation. Our goal for unit testing is to have code coverage above 90%. This will give us great confidence that our units are performing as expected and it will make it easier to debug if something is failing. We used a Test Driven Development approach and wrote the unit test for the code before we developed it. This helped provide better quality of tests and helped to ensure the tests fail until implemented. All of our unit tests were run every time a team member pushes code to the repository to ensure that the code is still performing as expected.

5.2 Interface Testing

Interface testing will consist of how multiple components interact with each other. Interface testing is vital because it will make sure components are working together as expected. Some of the tests will include: integrations with APIs, interactions between pages, and routing. These tests will not be included in our CI/CD pipeline but will be expected to be run by team members on a regular basis.

5.3 Acceptance Testing

For our acceptance testing, we created demo videos and submissions after any completion of a task and presented it to the team and clients during our weekly meetings. During that time we would evaluate each task and receive feedback from the client and make the appropriate fixes. After team and client testing, we created a beta test script of the application that we sent out to our beta testers which consisted of friends and family. The test script was written with full instructions of the purpose of the test script, steps to follow, sets of expectations per step, and a

feedback column for any comments or bug reports at each step. This was to help us to test the application with real users and to understand more about user experience. Using this process we were able to get feedback from a variety of users to help expose a variety of bugs for the team to fix and apply those fixes to our final product.

5.4 Results

Since we are working on an existing application, there were tests already in place for the application. After fixing the previous frontend and backend tests and the newly implemented google analytics for site tracking, we were able to focus on other methods of testing. We continued to have weekly meetings with our mentors and clients after new contributions in order to receive feedback and make appropriate changes according to the feedback. After completing all our deliverables we set for ourselves and gaining approval from the client, we started focusing on beta testing the entirety of our application, we focused on initial in house testing that consisted of our mentors and team members. After completing one week of in house testing we created a bug report that lists all the initial bugs found. We followed up with our new bug fixes during our biweekly meetings and received feedback to ensure our fixes were accepted. Once our in house testing was complete we made a beta test script that was sent out to family and friends. We were able to receive feedback from real users and they provided feedback on bugs we haven't caught before. We got feedback about the UI and application functionality, so we quickly fixed those bugs and got those approved by our mentors and client then pushed them to production. Using this process we were able to get constructive feedback to ensure our client and user acceptance.

6 Implementation

6.1 Implementation Details

We will implement the game using Vue.js for the frontend and Node.js for the backend, as well as MongoDB for the database to store information for the application. The game should be able to have reliable connections in areas with poor connectivity such as a football field. For this we will be utilizing Amazon Web Services CDN to help with the service. Hosting and deployment will be done using Heroku, which is a cloud platform hosting service. Additionally, we utilize Stripe to make payments for our application. Our animation for the bingo game is created using Unity3D, C# and Blender.

The application is implemented to be both mobile and web based. A user joins a Cow Patty bingo game with a valid donation. After a donation is made for the game, the user is redirected to the page with their bingo tiles. There will be a countdown to wait until the bingo game starts and once it reaches the endtime, the user will be redirected to a link where they can view the animation that will show the winning tile for the bingo game. Winners will receive a prize through the mail, and all donations will be given to the Boo Radley Foundation.

6.2 Related Literature and Products

Our reference to literature would be through the book To Kill a Mockingbird. The name of the foundation is based after Boo Radley, who is our project manager's dog who has passed away from cancer. Boo Radley is also the name of a character in the book To Kill a Mockingbird, and the character who Ken Johnson named his dog after. Some related products would include Bingo, Cow Patty Bingo Games and 50/50 Raffles. Our product is quite similar to 50/50 raffles as all of the money raised goes towards veterinary research and the supporters have a chance to win a prize.

6.3 Rationale for Technology / Software Choices

There are a lot of technologies that could be used so it was tough to decide what to use. We opted to use Vue.js, MongoDB and NodeJS as the three main languages and frameworks. A large part of this decision was that these were the languages the existing code base consisted of. The strengths of these tools are that they are very fast, modifiable and lightweight. We wanted to make sure our website could run on any machine to provide increased accessibility. The main weakness of these technologies is the team's lack of proficiency going in. Most of the team had little to no experience with these frameworks which makes it more difficult to develop with. However, as a team we decided it was well worth the learning curve to be able to continue using these frameworks. If we wanted to do things differently than the existing code base, we discussed other javascript frameworks such as React. However, since we also had limited React experience we decided it was safer and smarter to just use the existing frameworks. One last technological consideration was deciding what tools to use for the animation. Our options were to use a 3d modelling software such as Blender or Maya, use a game engine such as Unity or Unreal, or use Adobe AfterEffects or Acrobat. We decided to use Blender for 3D Modeling since it was free and some of our members had experience with Blender. We also chose to use Unity for the game engine due to past experience team members had with the product.

6.4 Standards

We followed IEEE engineering standards closely for our application. Standards made an important impact on everything from the development to the testing of the application. The standards we utilized are listed below with descriptions of how we implemented them into our project.

IEEE/ISO/IEC 23026-2015 Systems and software engineering- Engineering and management of websites for systems, software, and services information:
We utilized this standard to help manage our animations and images for accessibility purposes. We added textual information for our users with visual disabilities and provided color combinations to help out users who are color blind. Incorporating this standard provides inclusivity for our users with visual disabilities and challenges. Since our website is public, it needs to be accessible to as many people as possible, so it is very important that we adhere to these standards during our development process.

IEEE 1008-1987 IEEE Standard for Software Unit Testing:
Since we need to test a wide range of components in our project, we follow this IEEE standard for software unit testing to ensure proper component testing. Specifically, this standard mentions which components to test and how to effectively unit test so we can guarantee a well developed application for our users.

IEEE 1016-2009 IEEE Standard for Information Technology:
Since our stakeholders are a main proponent in how the application will be designed and developed, communication between the stakeholders and us as developers is important. This standard helps us communicate design decisions with our stakeholders. Primarily we use this standard to figure out issues with quality assurance. This applies to our project as we have a lot of communication with our stakeholders and being able to communicate effectively leads to happiness from both sides.

IEEE/ISO/IEC 24748-5-2017 - ISO/IEC/IEEE International Standard - Systems and Software Engineering--Life Cycle Management--Part 5: Software Development Planning:
This standard involves planning for software development, and how to improve its consistency. It gives us a set of steps to be followed in a specific order to help us with our development process so we maintain proper planning and meet our deadlines.

7 Closing Material

7.1 Conclusion

In conclusion, our team has largely completed the goals set out for us at the beginning of this class. The core two goals we were given are (broadly) creation of an administrative statistics interface, and the creation of a dynamic animation. The first goal was completed relatively quickly in the life of this project, and the additional time following its creation was used to refine it following comments from our client. The second goal wasn't completed to the extent it was initially intended. Instead, it was refined over time as it was discussed and iterated upon. We have created the framework as well as documentation to simplify the creation of artistic aspects of the animation.

Overall, we have greatly enjoyed working with Ken, Dan, and Ben on this project. We have been able to learn many things that we couldn't have learned through classes, and it was all done in a constructive and personal way.

7.2 Lessons Learned

Going into this project, plenty of our members had light experience in software development and the patterns surrounding it. We all knew either how to program in the languages, or how to quickly gain efficacy in them. The main thing we lacked experience with though is the use of these languages and practices in a real world environment. Having to apply them to these non-college scenarios allowed us to rethink our use of these skills, and reassess their importance.

Take for instance our uses of meetings. In the beginning our meetings were relegated to once every two weeks, where we would cover our progress between meetings in a loose, unspecific way. This is the same meeting strategy we would usually employ in other classes with our peers. By the midpoint of our second semester, this completely changed to a meeting every week, where completed tasks were organized by both section and person. This change by our mentors led us to be much more organized and on task, a lesson we'll all take into our future software management situations.

As for the technology itself, the code of previous senior design groups has shown us their massive emphasis on testing and CI/CD. These existing implementations have made further development much simpler than it would have otherwise been. Because of this, we all know firsthand the importance of testing and early CI/CD implementations.

7.3 Future Work

Moving forward, the Cowchips4Charity project has one main method for expansion, the animation. Many of the core game features are completed and even stylized, to the point where a complete game could be played successfully by beta testers. Artistic aspects of the animation are one of the last remaining pieces of the original vision - consisting of animations and models to be slotted into the code frame created through the semester. Along with that, the beta testing scripts we created could also continue to be used to keep refining the Cow Chips application until it is ready for public use.

7.4 References

Cowchips4charity.com. n.d. Play Cow Patty Bingo To Fight Cancer! - Cowchips4charity. [online] Available at: <https://www.cowchips4charity.com/> [Accessed 14 November 2020].

Appendix I - Operation Manual

A.1 Frontend Manual

Frontend Quick Start Guide
==========

This guide is intended for new developers to get started with the frontend development of CowChips4Charity. The frontend technology used is Vue.js, HTML, CSS styling. You may use any IDE of your choice, previous developers used Visual Studio Code or Intellij for suggestion.

==========

Setting Up Your Environment:
The following steps will help allow you to run the application on your local device. Any changes to the application will only be seen on your local machine, until you push your code to be reviewed and pushed to production.

1. Download the entire CowChips4Charity repository off GitHub and save those files to one folder on your device.
2. Open up your terminal and navigate into the "cowchips-front" folder. Run the command "npm install", this will install the proper libraries and dependencies in order to compile and run your code.
3. After the installation is complete, open another terminal and navigate into the "cowchips-back" folder. Run "npm install".
4. After the installation is complete, run the command "npm run serve" to start the backend connection.
5. Go back to your other terminal that's in the "cowchips-front" folder and run the command "npm run serve". The application will be hosted on "localhost:8080". Navigate to your preferred browser and enter " http://localhost:8080/" into the address bar. You can now see the application on your local computer.

==========

How to test the system:
The following methods of testing are suggestions that will help allow you to test the frontend manually. These steps do not cover automated testing done by the CI/CD.

- Write test scripts. It is recommend to write test scripts for functionality and components. You can write test scripts and add them to the "tests" folder in the "cowchip-front" folder. Decide which type of test you created, unit or end to end (e2e), add the test to that specific folder. After writing the test run the following commands according to your type of test.

    - Unit Test: “npm run test:unit”
    - End to End: “npm run test:e2e”
    - Both: “npm run test”
- Manual testing is an option too. It’s recommended to do manual testing for the UI updates/fixes. While your server and frontend are running, make any changes in code and save it. Open your browser and refresh to see those changes made.
    - To help with individual UI testing you can right click in the browser and click “inspect”. Using inspect will allow us to see the application on different devices, and it can also be used to fix the UI temporarily without disrupting the entirety of our code.
    - To test games you will need to open the admin panel on your browser and set up games. **see admin manual**

After completing the testing, push the new tests or changes to your branch for peer review.

==========

How to beta test the system:
The following helps instruct you to our beta test. It is recommended to figure out a system of how your team would like to collect and store bug reports. We utilized jira tasks to keep track of what the bug is, the status of when it was fixed, how it was fixed, and the results.

Before testing, navigate to the admin panel and create a game for each day once testing has begun. This will ensure there are playable games no matter what day we are testing. This will be beneficial especially once user testing is started, it will allow us to ensure there is a game running no matter what day they decide to test. **see admin manual**

1. Beta Test In House: First, we suggest doing in house testing with the team, client, mentors, before having real time users go through the application. The goal of in house testing is to find initial bugs, so those are fixed before the application reaches real time users. The goal is to find edge cases, and test the application on multiple devices. This is where “inspect” on the browser is helpful in testing multiple devices. As a team you can also utilize the beta test scripts which will walk you through the application. The beta test script will be provided after this.
2. After completing in house testing ensure those initial bugs are fixed and those are pushed to production. You can send out the beta test script to all outside users. They can leave feedback on that script. It’s recommended to try to get a variety of users playing the game simultaneously to test the ability to host many users on our servers at once. It is also recommended to find users with a variety of different devices to test the application and how well it works with different devices. **Because we are testing the winning tiles, make sure you update the test script and provide the users with the winning tile so that we can ensure some testing for winners.**

The following is our beta test script that focuses on helping the user create an account, understand the different features of our application, and how to play a game. We will also be able to keep track of the devices used.

Steps and expectations for beta testing

The following steps are to play a game. The left column states what actions are to be performed. The middle column is the results that should happen/be seen after performing the step in the left column. If your results are different than what is expected please use the third column to provide a description of what happened (provide any type of documentation of this issue i.e. screenshots). If the results match then leave blank. If you have any other comments/critiques during a step please provide those notes in the third column corresponding to that step! Thank you once again for helping the CowChips team beta test our product!

Before beginning please provide the device you are testing with: **HERE**

First, let's navigate to cowchips4charity.com

Steps	What Should Happen	What Happened
1. Go to https://www.cowchips4charity.com/ on a mobile device (optimized for mobile but also works on desktop)	You could see the homepage with options Login, Register, About, and a PayPal option to donate	

Next, let's create an account

Steps	What Should Happen	What Happened
1. On the homepage, click Register Enter name, email, password, and verify password, then click submit	You should be directed to a form that asks for name, email, password, and verified email Once you've clicked submit you should be redirected to the logged-in homepage else you will get another error if there is already an account with the same information	

Let's play a new game now!

Steps	What Should Happen	What Happened
1. Click play on the homepage	You will be redirected to our game form	
2. Click the organization you want to support, then click next	After clicking play you should see a set of organizations to pick from. After clicking next it should stay on the same screen but the tile select will display	
3. Select the tiles you believe will be a winning tile. (The winning tile is the day you're playing the game, please select this tile in your selection), then click next	You should be able to select as many tiles as wanted. You should see the price of each tile. Your donation amount should update as you select tiles, as well as a list of tiles you selected... Click next for the next step of donation payment information. You should not be able to advance if you have selected no tiles	
4. Enter this information for your card information: Card number enter 4242 4242 4242 4242 Expiration date: 04 / 24 CVC: 242 Zip code: 42424	You should see a payment information screen, enter our test card info. The amount in the pay button should correspond to the same on the previous page. Click pay "$$" and you will be redirected to our About screen.	
5. Click the "home" button	You should be redirected to the homepage	
6. Click "games"	The game you just donated to should appear here. There should be the title listed for	

	the game. The organizations that are involved in the game. And there should be an end time that countdown as the end time gets closer. Otherwise, if you have not donated you will see a message that tells you you need to donate	
7. On the game page, find the game card for the game you just donated to and click "View my tiles"	You should be redirected to see your tiles, click the back arrow to be redirected to the game page	
8. Once the timer reaches 0 / the current time is past the end time. The countdown will turn into a link that says Take me to the animation. Click that link.	The link will take you to our animation, there should be a loading screen with a cow it may take up to a minute for the animation to load and your screen should switch to horizontal orientation	
9. The animation will begin. The winning tile is in yellow.	The animation is about a minute long, look out for the yellow tile in the field, and it will lead to the scoreboard with the winners	
10. You will be taken to a thank you screen.	The thank-you screen will say Thank you for supporting the Boo Radley Foundation! If you have won it will ask you to confirm your address. Please enter your address so that the BooRadleyFoundation can send you a prize. If you have already entered your address the page will not ask you to	

	enter your address but it will say you are a winner.	
11. You have now completed the game.	You should have been redirected back to the home page	
12. You're a winner! On the home page click "Account"	You should be redirected to your account information, since you have won a game your address should already be displayed.	
13. Click the back arrow at the top to navigate back to the home page and click "Log out"	You have officially played your first game!	

Now that you have successfully played a game, let's test some other features of our application

Action	**What Should Happen**	**What happened**
1. On the homepage (logged out) click "Login" Enter your email and password and click login	You will be directed to enter your email and password, then you will be redirected to the logged-in homepage	
2. On the homepage click "Account"	Upon being redirected to the account page, your address should be already displayed. Update the address fields with a new address and click "Submit" and you should be redirected to the home page.	
3. Click "Account". After verifying what should happen, click the back arrow to the home page.	Upon being redirected to the account page, your address should be already displayed.	

4. On the homepage click “Log out”	You should be redirected to the logged-out home page	

Testing completed! Thank you on behalf of the CowChips & Boo Radley team for taking the time to test our application

A.2 Backend Manual

Backend Quick Start Guide
==========

This guide is intended for new developers to get started with the backend development of CowChips4Charity. The backend technology used is Node.js and MongoDB. You may use any IDE of your choice, previous developers used Visual Studio Code or Intellij for suggestion.

==========

Setting Up Your Environment:
The following steps will help allow you to run the application on your local device. Any changes to the application will only be seen on your local machine, until you push your code to be reviewed and pushed to production.

1. Download the entire CowChips4Charity repository off GitHub and save those files to one folder on your device.
2. Download MongoDB
3. Follow prompts to set up
4. If service is not already running, run the following command as admin in a terminal bash net start MongoDB
5. Suggested: Download one of the following
    a. Download MongoDB Compass. Be sure to set Versions to a Community Edition Version
    b. Download Robo 3T.
6. Create file .env in the root directory
7. Copy information from .env-dist into .env
8. Fill in the information with your system specific info (most should be the same)
9. Run the following command to initialize db with a pre-created admin user with all permissions set

bash npm run initdb
The admin credentials for the pre-created admin are
{ email: "a@gmail.com", password: "password" }
Note: If monogrestore is not found go to this site https://www.mongodb.com/try/download/database-tools download version for your system. Then move all of the files within the bin into ProgramFiles/MongoDB/Server/4.4/bin. Then add C:\Program Files\MongoDB\Server\4.4\bin\ to your Path environment variable.

10. Open up your terminal and navigate into the "cowchips-back" folder. Run the command "npm install", this will install the proper libraries and dependencies in order to compile and run your code.
11. After the installation is complete, run the command "npm run serve" to start the backend.

A.3 Admin Manual

Admin Quick Start Guide
==========

This guide is intended for new developers to get started with the development of the admin panel for CowChips4Charity. The technology used is Vue.js, Vuetify, CoreUI, and Google Analytics. You may use any IDE of your choice, previous developers used Visual Studio Code or Intellij for suggestion.

==========

Setting Up Your Environment:
The following steps will help allow you to run the application on your local device. Any changes to the application will only be seen on your local machine, until you push your code to be reviewed and pushed to production.

1. Download the entire CowChips4Charity repository off GitHub and save those files to one folder on your device.
2. Follow the instructions to setup the backend
3. Navigate to cowchips-admin and run npm install
4. After the installation is complete, run the command "npm run serve" to start the admin panel.

==========

How to view winners of the game:

1. Go to admin.cowchips4charity.com

2. Click the games tab
3. Type in the name of the game you want to view winners for in the search bar
4. Click the drop down to the left of the search bar. Select name.
5. Click search. And your game should appear.
6. Click the trophy icon within actions for your game.
7. The names of all winners will appear. Their address will be populated here if they have entered it after viewing the animation. If not just their name and email will appear.

How to view website traffic:

Option one: view the dashboard within the admin panel

1. Go to admin.cowchips4charity.com
2. Click the Dashboard tab
3. Click Access Google Analytics
4. A pop up will open. If it says this site is not trusted, click advanced and continue. Then enter your email address and password that has access to google analytics data. If you don't have access to Google Analytics data for CowChips4Charity ask someone who has access for access.
5. The charts will now be populated with the data for the past month.

Option two: view the data within google analytics

1. Go to analytics.google.com
2. Enter email and password
3. Click the dropdown in the upper left corner and select BooRadleyFoundation -> CowChipsFront - Production -> All Website Data
4. You can then click on the reports at the left hand side of the screen in order to see reports of the usage of cowchips4charity.com

How to view google analytics api usage from google dev console:

1. Go to console.developers.google.com
2. Enter email and password to login
3. Click the dropdown in upper left corner and select Cow Chips Admin
4. Click the menu button in the upper left corner and select APIs and Services
5. Then click on Google Analytics API

How to view google tag manager:

1. Go to tagmanager.google.com
2. Enter email and password to login
3. Click the dropdown in the upper left corner click BooRadleyFoundation -> www.CowChips4Charity.com

A.4 Animation Manual

Quickstart Guide

Quick Start Guide
==========

This guide is intended to help a new developer get started with the animation aspects of the Cowchips4Charity program. The core of this aspect is written in Unity, with the build being exported to the website for display. The version of Unity used is the current 2019 LTS version, **2019.4.23.f1**. This choice was to maintain use of Unity WebGL build specifics (UnityLoader.js, json organization), and would result in those aspects not working if the build was upgraded to 2020 or beyond.

Preparation for Development

Steps:

1. Download the Current LTS Version of Unity (**2019.4.23.f1**)
    - Recommended modules: Visual Studio, WebGL Build Support, Documentation

2. Add the Project to Unity Hub from the root file
    `CowChips4Charity-Animation`
3. Review the `ReadmeTableOfContents-Cowchips.md` file to understand the file Hierarchy
4. Start Adding!

Unity Beginner Notes

- All files are in the Project tab (by default in the bottom left)
- Scenes (in the Assets/Scenes folder) can be opened to show a particular view
- Active objects in a scene are shown on the left, in the Hierarchy tab (most are under a canvas object)
- Referenced scripts are attached to various objects (shown in the Inspector tab on the right)
- The play button at the top center allows you to play the game from the current scene

Building the Project

Note: Building requires the Unity environment be set up, which is provided by the **Preparation for Development** section of this guide

Steps:

1. From the Unity Project, Select `File -> Build Settings`
2. Under the **Scenes In Build** Section, Use the `Add Open Scenes` Button to Add Used Scenes
   - This section should already have all required scenes (2 total)
   - Make sure the `Main` scene is selected as the starting scene

3. Under the **Platform** Section, Select `WebGL`
4. Under the **WebGL** Section, Select an appropriate Target Platform and Architecture
   - Other options aren't needed unless you specifically want to use them
   - **Development Build** leaves the build uncompressed but with a watermark saying the build is a dev build

5. Select the **Build** button on the bottom of the window
6. Choose a Location to Build the File (A Build folder exists at `CowChips4Charity-Animation/Build`)
7. The Build is Complete!

Running the Project

Note: The project must be built in order to run it. Use the **Building the Project** section to build it. **Make sure to select the appropriate target platform for where you plan to run it (eg. WebGL or Windows)**

Steps:

1. Find the Location of the Build in a File Explorer
   - The Default location is the Build File in `CowChips4Charity-Animation/Build`

2. Copy the Inner Build folder to the Frontend Build Location
   - The Default Build folder is located at `CowChips4Charity-Animation/Build/Build`
   - The Frontend Build location is in `cowchips-front/public/static`

3. Run the Frontend
4. Navigate to `http://frontend-address/animation` to view the animation

Table of Contents

Cowchips4Charity-Animation Overview
==========

This is the overall structure of the Animation section of Cowchips4Charity. This overview is not designed as a complete description of individual files, but rather a broader description of certain sections. Below the tree structure there are explanations of the most important files (and sometimes important files located inside them)

Folder Overview

```
+---CowChips4Charity-Animation
|   +---Assets
|   |   +---AnimationControllers
|   |   +---Classic Skybox
|   |   |   +---01
|   |   |   +---02
|   |   |   +---03
|   |   |   +---04
|   |   |   +---05
|   |   |   +---06
|   |   |   +---07
|   |   |   +---08
|   |   |   +---09
|   |   |   +---10
|   |   |   +---11
|   |   |   +---12
|   |   |   +---13
|   |   |   +---14
|   |   |   \---15
|   |   +---JsonDotNet
|   |   |   +---Assemblies
|   |   |   |   +---AOT
|   |   |   |   +---Standalone
|   |   |   |   \---Windows
|   |   |   \---Documentation
|   |   +---Models
|   |   |   \---Materials
|   |   +---Plugins
|   |   |   \---WebGL
|   |   +---Prefabs
|   |   +---Resources
|   |   |   \---Materials
|   |   +---Scenes
|   |   |   +---Main
|   |   |   \---SampleScene
```

```
|   |   +---Scripts
|   |   \---TextMesh Pro
|   |       +---Documentation
|   |       +---Fonts
|   |       +---Resources
|   |       |   +---Fonts & Materials
|   |       |   +---Sprite Assets
|   |       |   \---Style Sheets
|   |       +---Shaders
|   |       \---Sprites
|   +---Build
|   |   +---Build
|   |   \---TemplateData
|   +---Core
|   +---Library
|   +---Logs
|   +---obj
|   +---Packages
|   +---ProjectSettings
|   \---UserSettings
+---ModelFiles
\---Renders
```

Important Folders

CowChips4Charity-Animation/ Assets

- This folder holds everything in the Unity project that is not automatically generated (scripts, models, scenes)

CowChips4Charity-Animation/Assets/ Models

- This folder holds all models that are imported into Unity (before they are converted into prefabs)
- Materials and Textures for these models are stored alongside them

CowChips4Charity-Animation/Assets/Plugins/ WebGL

- This folder holds Unity's access to JavaScript functions when built for WebGL
- `javascript.jslib` defines the JS functions to be called elsewhere
- `javascript.cs` imports the jslib functions, and offers simpler access to other .cs functions

CowChips4Charity-Animation/Assets/ Prefabs

- This folder holds the models that have been implemented into Unity (have attached scripts, or are combined)

CowChips4Charity-Animation/Assets/ Resources

- This folder holds Models and Prefabs that need to be dynamically added to the scene
  - This can be done using the function `Resources.Load()`

CowChips4Charity-Animation/Assets/ Scenes

- This folder contains the Scenes used in the animation
- Currently there are only two scenes, Main (the stadium animation) and WinnersScene (The Scoreboard)

CowChips4Charity-Animation/Assets/ Scripts

- This folder holds .cs files that control the progression of the animation
- `BillboardController` This file controls all actions taken by the billboard (creating tags, parsing the data from DataController, and finishing the animation)
- `BoardController` This file controls all actions taken by the tile board in the main animation (displaying the randomized numbers)
- `DataController` This file contains all calls taken by Unity to the backend. All data needed is collected immediately (upon start of unity), then is used by the various controllers to fill them with data.
- `NameTag` This file controls the NameTag prefab, which displays the winning players on the billboard (sets NameTag attributes, scrolls NameTags)
- `PersistThroughout` This file ensures scenes called after Main will keep their active elements. This is particularly with the DataController to offer its data to other scenes.

CowChips4Charity-Animation/ Builds

- This folder holds the builds whenever they are created
- The inner `Builds` folder is the folder to be put in the frontend
- `index.html` allows a quick display of the WebGL build, but it doesn't work because of the JS dependencies in the Plugins folder (tests can only be done from the frontend build)

CowChips4Charity-Animation/ ModelFiles

- This folder is just an easier to access `Models` folder, and is used only as an in-between for `Prefabs` and `Resources`

CowChips4Charity-Animation/ Renders

- This folder is used as storage for any rendered images/videos to be used in the project
    - The loading screen for the Animation is a render that is stored in this way

Notes

- Unity files other than the Assets aren't shown as they are largely automatically handled by Unity