CowChips4Charity

Design Document

Team Number 16
Client: Ken Johnson
Advisor: Lotfi ben Othmane
Members:
Justin Lee: Admin Panel Team Lead
Dustin Schultz: Test Engineer
Tyler Bartleson: Game Team Lead
Elizabeth Li: Game Developer
Meghna Vaidya: Project Manager
Brandon Bui: Admin Panel Developer

sdmay20-16@iastate.edu
https://sdmay20-16.sd.ece.iastate.edu/

Revised: 04/23/2020 Version 1.2

Executive Summary

Development Standards & Practices Used

List all standard circuits, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- Version Control: 3 main branches: master, dev, QA
- Code Review: Pull Requests were made before committing to main branches
- Standard Practices: conform to existing standards within partially-completed project we received
- Specialization: members had focus areas to reduce redundancy
- Use Cases: Use Cases were organized on GitHub Project Boards

Summary of Requirements

Functional Requirements

- The User shall be able to choose a square(s) depending on how many squares they bought
- The User shall be able to choose which game to watch
- The User shall be able to watch the game live
- The Game shall be able to choose an unbiased square for every game
- The Game shall inform which square won
- Multiple games should be able to be run at once
- Game time should last between 5-10 minutes
- Game should have a limit to how late a user can choose a square(s)
- The Admin shall be able to see various data in the data panel (donations per game, the number of people who donated, donations per team, etc.)

Nonfunctional Requirements

- The Website shall look aesthetically pleasing
- The Website should be able to work on any device in any browser
- Security
    - The entire credit card transaction information will be encrypted
- Usability
    - The user shall be able to complete the transaction in one page of the web app
- Reliability
    - The application shall have 100% runtime during football games
    - At least 1000 users will be able to use the web app as it is designed to work
- Operational
    - The web app will be compliant with all policies and regulations set forth by the Boo Radley Foundation
- Performance
    - The application shall be able to support scalability for multiple football game users
    - The user shall be able to login to his/her account in less than 15 seconds

  - The winning user shall receive a notification within 12 seconds of the cow finishing
  - The web app shall be able to calculate the entire cost of the donation once the user is checking out within 3 seconds

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- COM S 309
- SE 319
- COM S 228
- COM S 311
- COM S 363

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Vue.js
- Node.js
- Express
- Heroku
- mLab
- AWS
- Stripe

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Our team would like to thank the Boo Radley Foundation and our client, Ken Johnson, for allowing us to work with them for our senior design project. Ken has been very helpful for the entire creation of the project. Ken has also provided financial aid for the product whenever it was necessary. Additionally, we would also like to thank our faculty advisor, Professor Lotfi Ben Othmane for volunteering his time.

1.2 Problem and Project Statement

The Boo Radley Foundation funds the research of diseases that are common between humans and household pets. They believe research on these animals is a better way to find cures and treatments for animals and humans with the disease. To fundraise for their cause, the Boo Radley Foundation hosts an event called CowChips4Charity. This event is intended to be held at halftime of college football games between two participating schools. The event is a bigger scaled version of cow chip bingo. Participants will select a square on a 36x36 grid and if the cow selects your square, you win. Currently, the foundation does this in a real field with a real cow and presents the outcome via video feed on YouTube. The cost for the cow and time to make the grid are too high for the foundation. Our goal is to convert the cow chip bingo process to a digital version. We believe this will considerably decrease the costs for the cow chip bingo game and increase the participants and donations of the event.

Our team will develop a cross-platform web application for the CowChips4Charity event. The Boo Radley Foundation will use the web application to enable people to participate in the event. Participants will sign in to the application, select which football game they want to play at, and then select square(s) on the grid to buy. Once the user has at least one square selected they will then pay via credit card. This will occur all the way up to halftime during the football game. During halftime, participants will then go to our app to watch the outcome of the CowChips4Charity event. Our web app will then show an on-screen animation of a cow that the defecates on a certain square in the grid. People who picked the winning square will then be notified that they won and to claim their prize.

1.3 Operational Environment

- For any end product other than simply a calculation or simulation, it is essential to know the environment in which the end product will be used or to which it is expected to be exposed or experienced. For example, will the end product be exposed to dusty conditions, extreme temperatures, or rain or other weather elements?
- This information is necessary in order to design an end product that can withstand the hazards that it is expected to encounter.

Due to the fact that our end project was only software and did not involve any hardware it will not be subjected to any form of physical hazard such as weather. However, the project is designed to be used during the half-time of college football games both by users watching from home and users in the stadium. This environmental factor caused us to have to take into account spotty cellular and internet connection that is often found in college football stadiums when we created the project so that even the users in the stadiums can still use the product.

1.4 Requirements

Functional Requirements

- The User shall be able to choose a square(s) depending on how many squares they bought
- The User shall be able to choose which game to watch
- The User shall be able to watch the game live
- The Game shall be able to choose an unbiased square for every game
- The Game shall inform which square won
- Multiple games should be able to be run at once
- Game time should last between 5-10 minutes
- Game should have a limit to how late a user can choose a square(s)
- The Admin shall be able to see various data in the data panel (donations per game, the number of people who donated, donations per team, etc.)

Nonfunctional Requirements

- The Website shall look aesthetically pleasing
- The Website should be able to work on any device in any browser
- Security:
    - The entire credit card transaction information will be encrypted
- Usability
    - The user shall be able to complete the transaction in one page of the web app
- Reliability
    - The application shall have 100% runtime during football games
- Operational
    - The web app will be compliant with all policies and regulations set forth by the Boo Radley Foundation
- Performance
    - The application shall be able to support scalability for multiple football game users
    - The user shall be able to login to his/her account in less than 15 seconds
    - The winning user shall receive a notification within 12 seconds of the cow finishing
    - The web app shall be able to calculate the entire cost of the donation once the user is checking out within 3 seconds

1.5 Intended Users and Uses

This product is intended for two colleges to go against each other to raise money for the Boo Radley Foundation. The end product should allow for users to donate through the website to support their college and the Boo Radley Foundation. The end product should also allow for the user to see the game on their device.

1.6 Assumptions and Limitations

Assumptions

Users will have a device from which they can access can access the internet

- In order to be able to get to the website to play the game, users will need an internet connection

Users will have a credit or debit card

- In order to bet on a square or squares in the game

The product shall comply with all laws and regulations

- States have different privacy laws

Admin panel users will not have an in-depth software development knowledge

Limitations

Internet connection in the college football stadiums tends to be spotty.

1.7 Expected End Product and Deliverables

The deliverables we will produce will be amendments to existing documentation, and an expanded final project cross platform web application. A large portion of the documentation for the project already exists, but the additional changes we made will be reflected in updated documents. The new/adjusted deliverables were: a new use case diagram for the analytics portion, an updated initialization document, and a scope and requirements document for the amendments we will be making.

Analytics Use Case Diagram

One of our major changes is a large expansion of the analytics portion of the admin dashboard. This describes the interactions that an admin will have available to them, and the flow of events of those actions. This document solidifies the client and team's understanding of the expansion's expected flow.

Updated Initializations Document

To ease the startup process during handoff, we've created a document that describes the process in detail of getting the project running on a fresh machine, so future developers will be able to work with much less transition time. Handoff can be complex, with this project requiring a large set of domain knowledge. This document eliminates this requirement.

Scope and Requirements Document

This document will serve as a high-level guide for the desired output of our work on the application. It will be frequently evolving as time allows additional expansions. Despite this, an initial version will be approved by the client to ensure a full understanding of the expected work.

Final Version Web Application

A fully functional version of the expanded web application will be the ultimate deliverable. It will include all our new features, including game generation, data visualization, and analytics.

2. Specifications and Analysis

2.1 Proposed Approach

The main components that we implemented for this project is the game animation and the admin panel analytics.

For the game animation, we discussed what potential tools and libraries we could use that would fit into our existing framework and adhere to the requirements. After updating a lot of the current pages on the website, and experimenting with Javascript, css, and HTML for the game page, we decided to use already existing vue components to create the final game board. For the animation needed for the game we just decided to use photoshop.

For the admin panel analytics, we investigated data visualization libraries that would fit into our framework. Also under our consideration, were which analytical tools would provide output for use in our selected data visualization techniques. We ended up using Vue Bootstrap for the graph prototyping and the general UI of the admin panel. We also used Socket.io for live data sampling.

We strived to follow the IEEE Standard Std 829-1998 within our test planning.

2.2 Design Analysis

Our components fit into the existing application, with exposed endpoints in the code ready for development. After we examined these openings, and discussed with our technical contact we concluded that this was the correct approach.

The implementation of new libraries for data visualization, analytics, and animation were beneficial to both development time, and maintainability of the product. This conformed to the standards already existing in the project, which already used libraries where possible to avoid redundant development.

While it was a time consuming activity to select and use the correct tools mentioned, the benefits outweighed this weakness as we had faster development and better maintainability.

2.3 Development Process

For this project we followed the Agile development process. We settled on using the Agile development process because we thought that it would be the easiest and most efficient way to have meetings, work together, and organize the tasks required to complete our project. More efficient meetings meant that we were able to make the most of whatever time we were able to have all of our team members together; allowing us to come together and discuss what we had all gotten done, what we all still have to do, and what problems any of us were having. Agile helped work together easier because we knew the current progress of our project. Finally, the Agile process helped us organize the tasks required because it allowed us to know who has been assigned which task and the approximate time of when each task should be complete.

2.4 Conceptual Sketch

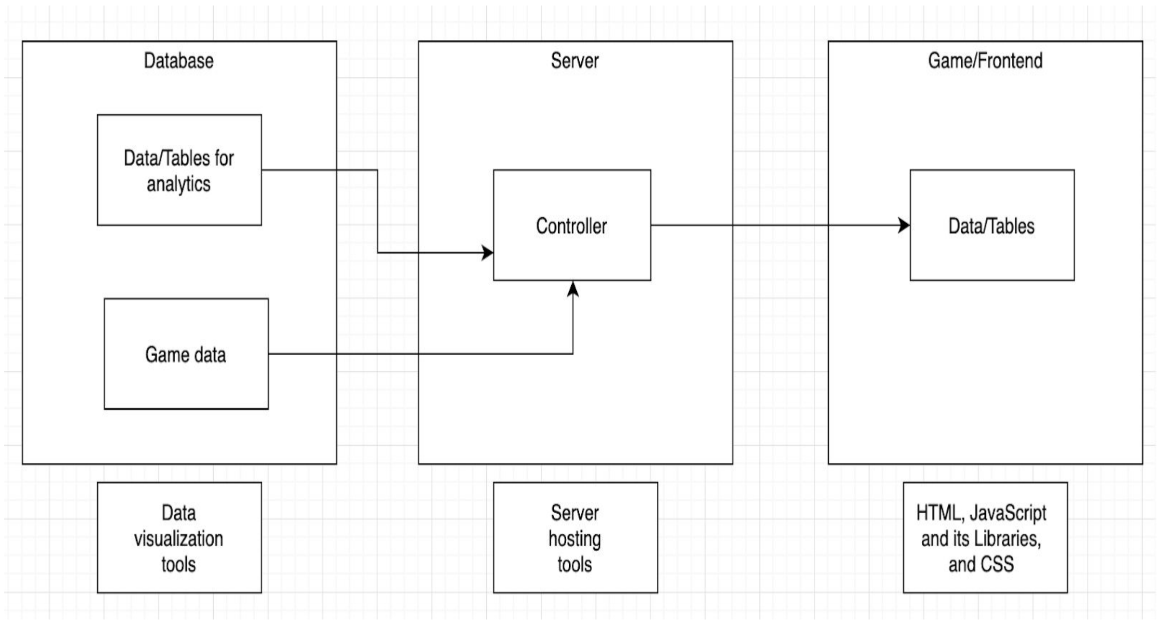

Figure 1: Conceptual Sketch

3. Statement of Work

3.1 Previous Work And Literature

This project is basically a bingo game that involves betting on some squares, which are two basic ideas that have probably been done many a time.
We have access to the previous teams work on this project, and it can be used as a basis for our work this time around.

Advantages

- The access to the previous teams work, gives us more of an insight as to what our client's requirements actually means.
- It gives us a visual on what our end product should look like and on what parts we can build upon and make better

Shortcomings

- We are limited to the technologies we are using, we have to use what the previous team used, as to keep everything on the same page.

3.2 Technology Considerations

We used the previous group's technology stack (Vue.js, Node.js, etc.). We addedChart.js, which is compatible with Vue.js and Node.js. This saved us time by using readily available, easy to use components that are well-documented and easy to edit. Additions and changes to the analytics pages using these charts will be easy because of the developed documentation. Similarly, we used the well-known socket.io for realtime communication. This popular module is familiar to many developers, so future maintainers will spend less time learning new technology.

3.3 Task Decomposition

There are three main aspects to this project: the game itself, an admin panel with analytic data from the games that have been played, and the scalability of the project itself. In order to complete all of these aspects of the project it will help to break them down into multiple tasks as follows:

<u>Admin Panel Tasks:</u>

- Analytics/Statistics
  - Specific Statistics
    - Per Game
      - Donations per game
      - Number of people who donated
      - Donations per team
      - Amount of winners per school
    - Per University
      - Lifetime total
      - Personal best game (donations, players, winners)
    - Admin Only
      - Live total money

  - Adjust data collection to track
    - Keep tile coordinates
    - Donation: get game ID with team association
- Data Visualization
  - Add subsection to admin panel
  - Support for the specific analytics
    - Pie charts
    - Bar charts
- Realtime data
  - Setup event-driven (upon donation received) data updates to web app

<u>Game Tasks:</u>

- Update website UI
  - Update home page
    - Update logo
    - Change buttons
    - Add link to Facebook page
  - Update donation page / credit card pages
  - Update about page
  - Update account page
  - Update register page
  - Update select org page
  - Update select game
  - Update select team
  - Update your tiles page
- Create look of game
  - 6 x 6 grid
    - Randomly numbered squares
  - Animated cow
  - Digital field & patty
- Add game functionality
  - Make cow move randomly
  - Make game last between 5-10 minutes
  - Make cow move over winning tile as the cow patty is generated

Scalability:

- Make able to appear correctly on different sized devices
- Allow multiple games to be run at once
- Allow multiple users to play a game
- Make able to appear correctly on different internet browsers

3.4 Possible Risks And Risk Management

A part of this project that would slow us down would be our collective team's knowledge of the technologies we are using. We were going to be using technologies that most of us on the team were not knowledgeable about, including Vue.js, Node.js Express, Heroku, mLab, AWS, and Stripe. While some of us may have had some exposure to these, nobody was proficient in them, and required all of us taking the time to really learn what each one of these do, and how they played into our overall project.

3.5 Project Proposed Milestones and Evaluation Criteria

The milestones of our project largely revolved around the two different aspects of our project. Some milestones of the Game aspect of our project included: creating working visuals for the game, creating backend so that the game can be run from start to finish, one game is able to be played at a time, multiple games are able to be played at a time, the game can scale to multiple sized devices, the game can be run on multiple browsers, and the game can run with little internet connection. Some milestones for the Admin Panel aspect of our project included: the ability to collect data from the game, the ability to store data from the game, the ability to analyze /calculate additional data from the data collected from the game, and being able to show that data as graphs and diagrams. The final milestone for the project was our final presentation to the client at the end of the year.

3.6 Project Tracking Procedures

Our team used github to track our progress. We used a storyboard with use case cards to keep track of what needed to be done, who was working on which card, which card needed to be tested, and which card was done.

3.7 Expected Results and Validation

A desired outcome is to have a working simulation game of Cow Patty Bingo. This includes an animated cow randomly roaming a digital Bingo card that looks like a pasture. Each square will be numbered and be able to be bet on by a user. The cow will then randomly deposit a cow patty indicating which square wins. Another desired outcome is a working admin panel that can analyze and display various data desired by the client in the form of easy to read graphs and charts.
We have confirmed that our solutions work by testing our code on a dev environment hosted on Heroku. Here, we made sure all our implementations work as planned. Because all our deliverables work, we pushed to the production server.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

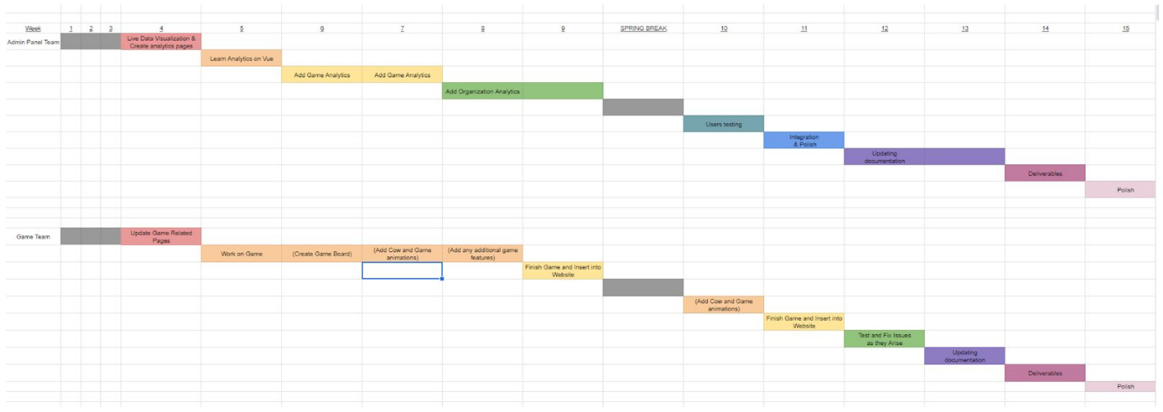

Figure 2: Timeline

4.2 Feasibility Assessment

The realistic projection of our project will be that the website will be able to run multiple games at the same time while the admin panel will be able to see various data on the games live. Some challenges we thought of were adapting and understanding the previous group's project. This project's feasibility will also be affected if the users are not able to connect to the internet and donate and play the game. Another factor that could affect the project's feasibility is getting a fanbase to play. Because the game is niche, not everyone might want to play this game. It is also hard to start a fanbase and keep it growing.

4.3 Personnel Effort Requirements

Task	Description	Approximate Hours
Update Front End	The current frontend will be updated to look more modern, and more aesthetically pleasing.	20 hours
Support Real Time Data	For the analytics, real time data needs to be implemented to get a more accurate understanding of the current live games.	100 hours
Add Analytics to Back End	Various statistics will be useful to get an understanding of how users interact with the game. These will be usable by administrator users for promotion and marketing.	50 hours

Add Game	The game itself is the most significant addition. It will be the simulated cow portion that enables the whole operation to function in a digital environment. There will be animations that respond to incoming data for positioning.	150 hours
Add Data Visualization	Once analytics are generated, visualizing them will be the next step. These visualizations will work for promotion, as well as getting an understanding at a glance.	75 hours
Integration/Testing	Final testing and integration will be done to ensure whole system functionality. Bugs discovered in this stage will be patched..	50 hours
Update Documentation	We will enhance and expand current documentation to reflect our changes, as well as clarify portions that we found challenging to understand as an initially uninformed third party. This way future maintainers will benefit from our initial misunderstandings.	15 hours

Table 1: Personal Effort

4.4 Other Resource Requirements

Due to the fact that this project is entirely software the only resources we needed were different softwares and frameworks.

4.5 Financial Requirements

Our project did not have any financial requirements.

5. Testing and Implementation

We integrity tested our data, playtested our game, and conducted user-studies to see how the UI reacts to each scenario. It will be essential to test the entire bingo game with every rule, the ability to choose squares, watching the game live, determining the winning square, the ability to run multiple games at once, the game time, user limits, and the admin privileges that include donations per game, those who donated, donations per team, and other data.

Our test cases are based off of the game rules- the ability to choose a square by the player based off the squares purchased, user being able to choose which game to spectate, watching the game live, unbiased square, determine winning square selection, having multiple games run at once, the limited time frame for each game, limits to how late a user can select a square, and having Admin be able to accurately see the data such as donations per game, the number of those who donated, and donations per team.

The anticipated test results for the actual game are based off of the rules of how Bingo works and having the constraints properly set for one winning square per round, having more than one game run at once, having the game only last between 5 and 10 minutes, limiting how late a user can select a square, and accurate data be exact to the predicted data that is calculated by the tester.

We used Test Driven Development practices, with test cases being written before the story's code. This ensured that the story's requirements were fulfilled, and that test cases created that can be run again later to test for regression.

Once the features were implemented, we performed the actual tests and evaluated the results to improve our product and retest, after documenting our entire process and results. We can run those tests and a lot of our results are based off of what is simulated in the game. We believed that the biggest challenge would be integrating the frontend and backend and allowing the user to watch the games live and having multiple games go off at once.

5.1 Interface Specifications

Our expansion of the game functionality will include new endpoints for data transfer

Game Endpoints

- GET /event?id=<number>/game - Retrieve updates for the current game (cow sprite movement, end declaration)
- GET /event?id=<number>/stats - Retrieve updates for the current game's (active users, team percentage, etc)

5.2 Hardware and software

We used a CI/CD pipeline run primarily by Travis CI, with Jest and Cypress ran as plugins. Travis CI was the driver of the pipeline, with Jest and Cypress test cases being run as part of the build process run by Travis. Every commit/merge to a branch kicked off a build within Travis, allowing errors caused by the new commit to be found immediately.
Jest served as our backend testing of components. It executed test cases, similar to JUnit like the team was already familiar with. All existing Jest test cases were run with every build to ensure no regression.
Cypress served as our frontend testing. It executed test cases by simulating user input to a browser, testing the functionality of our site from the user perspective. These test cases were also run on every build for regression.

5.3 Functional Testing

We used unit testing and acceptance testing.
Unit testing was the primary focus while developing. Some tests were developed before code was written to ensure proper testing of individual parts of the code is complete. These tests revolved around the main objectives of the code, as well as any internal components of more complex code

determined on a case-by-case basis. These test cases were included in the CI/CD pipeline and automatically run upon every new commit to prevent regression.

Acceptance testing when significant components were completed. These tests were done by hand, with a developer (or client, when final testing was done) acting as a user, navigating the site to test the flow, usability, and functionality of the application.

Smaller sub-portions of acceptance testing were done using Cypress. These will be considered part of the unit testing since they focus on a more micro level.

5.4 Non-Functional Testing

We used integrated systems testing and acceptance testing.

Integrated systems testing will occur to test the application under a more realistic environment. This includes a high amount of concurrent traffic, ability to function with imperfect connection, time to load, etc. We will also measure time taken to perform basic functionalities, under normal traffic and high amounts to ensure the systems are fast enough to support the users.

Acceptance testing will occur similarly to integrated systems testing, with a focus more on normal conditions rather than stress testing ones. Usability will be emphasized, with individual users (ideally some unconnected to the development of the project) performing the tests, to gather feedback on the usage flow, and intuitiveness of the UI. Feedback will be discussed with the client (who also functions as an acceptance tester), and implemented as deemed appropriate.

5.5 Process

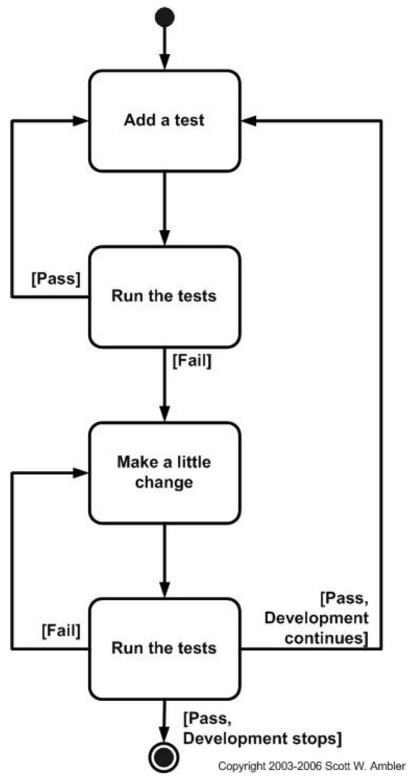
Add a test
Run the tests
[Pass]
[Fail]
Make a little change
Run the tests
[Fail]
[Pass, Development continues]
[Pass, Development stops]
Copyright 2003-2006 Scott W. Ambler

Figure 3: Process

Our process started as soon as the feature we wanted to test was implemented. We go through the process of creating the test cases, run the tests, and if tests fail we go ahead and implement the necessary changes to get them to pass and rerun the tests. The cycle is repeated until there are no more features to test.

5.6 Results

We anticipated how testing would go and learned how to test and improve based off of the results. It was assumed that all team members would know how to run unit tests, playtests for the game, and debug accordingly. Any form of errors were possible when testing, so we decided it was best to get our project up to speed as best as we could be to be able to resolve the conflicts we had as soon as possible.

Since our project involves a majority of UI, a lot of the testing was visual and dependent on the results of the game actions. A large amount of the testing was a result of playing the game and seeing the UI interact with it and based off of the tester's analysis of the visuals.

This project truly tested the knowledge and capabilities of every team member. The biggest challenges consisted of getting the frontend and backend working together and implementing the analytics. Another major issue was understanding how to use the environment since the project pre-existed and a lot of documentation was missing with how to set it up, how it all connected, etc. One of the biggest hurdles early on was making sure everyone's environment was set up correctly and still, not everyone's environments were properly working even to this day.

However, despite these hurdles, we were able to meet the standards of the specifications and ultimately satisfy our client. We successfully created the game page and game aspect to the project and all of the required inner-workings of the admin panel and have the game work.

6. Closing Material

6.1 Conclusion

Our goal for this project was to be able to provide the technological solutions our client needs for the CowChips4Charity initiative to raise money for the Boo Radley Foundation. In order to do this we worked on creating an interactive web application where users can bet on square to play the CowChips4Charity game. We also worked on creating an easy and organized admin panel for the admin user to be able to see analytical data about any of the games that are played. The intention of the project to be able to raise awareness and money for the Boo Radley Foundation.

6.2 Engineering Standards and Design Practices

Throughout the project we adhered to the organization standards of the Boo Radley Foundation by using the Foundation's logo and other supplied images throughout the renovation and updating of the website. We also had the constraint of continuing the production of the project with the design standards set by the past team.

6.3 References

We have been extensively using the documentation created by the previous group for understanding the existing implementation. It has been critical for our success. Similarly, our points of contact Ken Johnson (business side) and Daniel Lev (previous team member, technical side), have provided us immense support in our work.

6.4.1 Appendix I

General Setup and How To Run:

1. Download Code, Download Node.js and Mongodb (Community Version is preferred)
2. In all 3 subfolders (admin/backend/frontend), npm install to download node modules
3. In each subfolder, copy .env-dist, rename to .env and save
4. For backend .env file:
    a. Set DB_URI to "mongodb://localhost:27017/cowchips4charity" (assuming default mongodb settings)

   b. Set DB_USER and DB_PASS to “”
   c. Set JWT_SECRET to “test” (Can be any string, but not blank)
   d. STRIPE_SECRET_KEY should follow Stripe Setup below
5. For frontend and admin .env file:
   a. Set VUE_APP_BACKEND_URL (default: http://localhost:3030)
6. For admin .env file:
   a. Set VUE_APP_WEBSOCKET_URL (default: http://localhost:3031)
7. FOR MACS: To start mongodb (Windows auto-runs mongodb on startup once installed):
   a. cd into the mongodb folder, cd into bin
   b. Run `./mongod`
   c. When you want to kill mongod, use ‘ctrl + c’, or else it’ll be running the background and you will have to manually kill it
8. In the backend folder, run `npm run initdb`
9. Open up four command line windows:
   a. Window #1: If not already running mongodb, run mongod
   b. Window #2: Cd into Backend subfolder and run `npm run serve`
   c. Window #3: Cd into Frontend subfolder and run `npm run serve`
   d. Window #4: Cd into Admin Panel subfolder and run `npm run serve`
   e. Window #5: Cd into CowPattyBingo subfolder and run ‘http-server’
10. Change stripe key in cowchips-front-master/src/views/donation.vue
    a. Only when pushing to production

Stripe Setup

1. Make account on stripe.com
2. Get Private key from Stripe Dashboard and paste into Donation.Vue in Frontend subfolder
   a. let stripe = Stripe(“your_pk_here”)
3. Update Backend .env file with the secret key from your Stripe Dashboard and update STRIPE_SECRET_KEY