THE REGIS COMPANY

Final Report

By: Jacob Parker, Daniel Glynn, Alejandro Belli

The Regis Company

June 15, 2021

Table of Contents

Introduction

For our project, we were working with the Regis Company to design a market-simulation game centered around the timber industry. One of The Regis Company's areas of expertise is working on designs and developing market-simulations for other companies that model various scenarios. For our specific timber industry project with the Regis Company, we are developing a web-based market-simulation game that may be later used as advertising for The Regis Company.

In the game, the user starts with a factory in a certain region, and that factory produces stock which can be used for various actions such as selling the stock for cash, moving it around to other regions, and other events. The cash made from selling stock can be used to buy other factories, upgrade already existing factories, or to increase/decrease the price of stock in a particular region. The game has a total in-game run time of one year that is scaled to run for one week in real-time. Any user can participate in the game and have a live influence on the game that will affect any user playing the game, and will have their progress saved any time they rejoin the game within the week, however, when the week is over, all data will be reset and the entire game will restart from the beginning.

Our project will mainly be updating the user interface and useability of the front end of a pre-existing project. The previous team that worked on the project mainly worked on back end functionality of the game and focused on game-functionality and server construction. For our project, we have rewritten the old code to improve upon the user interface while adding useful features such as live chat, tooltips, and overall functionality to improve upon the already existing functionality. Additionally, we added various features involving front end support for the player or enabling more interactions between players. We have also changed the current playstyle of the game to function correctly and be playable on desktop, mobile, and other devices. The final product should be able to be used by anyone and scale for any device, has an overhauled UI, and can be played without errors. We have also added documentation via comments within the code itself to ensure high quality coding style as we developed the project.

The project was built in JavaScript using the React library for the front end user interface design. React is a JavaScript library for creating interactive UIs that efficiently render and update. Additionally, we used the Material-UI React framework for the specific UI components.

Requirements

For our project, our primary focus was a front end user interface overhaul and redesign of the project. One of the biggest items we had to tackle was changing the color scheme of the project to improve upon the old design. The old user interface used a lot of red and grey which made some of the functionality confusing on whether some of the buttons were bad or good, so we were tasked with outfitting the project with a new color scheme to fix this issue. Another big UI component we had to revise was the overall layout of all the buttons and region placements in the game. Previously a lot of the buttons were clumped together and had large outlines underneath them that cluttered the game and all the sizes and dimensions were hardcoded to be displayed properly at one resolution, so we were tasked with designing the layout of the project to scale properly, look more clean and organized, and make it more intuitive to the user where each button is in the game. Lastly, the main UI redesign feature we had to complete was to move the data for the user from the base starting screen to pull-out drawers accessed via buttons instead. With the limited space the game has, especially on a mobile device, having all of the data on the main screen took up too much space and appeared cluttered. As a result, we were tasked with creating multiple drawers that contained all of the useful data such as log entries, player financial data, and chat features that were previously on the main screen (aside from the chat button which is a new feature we implemented). This helped keep the main screen clean and organized while also keeping all the functionality that the game had originally.

The previous project didn't have a lot of UI functionality to account for different screen sizes, so our other primary goal was to allow the game to resize itself based on whether or not they are on a desktop or mobile device, and also to scale with the specific resolution and dimensions of the device they are using. We also had to do a bit of redesigning the overall layout of the game for mobile to make it look cleaner, such as reducing the size of the Regis Company logo, and moving a couple of the components around to fit better on mobile.

Lastly, we needed to improve upon the game functionality for the user by adding tooltips and error handling. For the tooltips, previously there was no tutorial or anything to tell the user how to play the game correctly, so it was confusing and difficult to figure out how the game worked. By adding tooltips to each of the components of the game, it allows brand new users a way to read about how the game works and start off strong rather than having to guess about what everything does when joining the game for the first time. For the error handling, when our team took over the project there was nothing stopping players from spending money they didn't have on in-game functionality. This meant that players could potentially have negative money if they spent more than they had. With error handling, we added barriers that will prevent this to occur, and will give the user a notification when they are trying to spend more than they have to prevent any kind of negative values from showing up.

Moving away from our primary requirements we had, we were given a few stretch goals to work on if we had time at the end. One of the bigger stretch goals we had was adding functionality for a chat feature. During our development we managed to make a skeleton of this, but we did not get the full functionality of being able to communicate with other players live while playing the game. Another feature we would have liked to implement was a smart help bot to assist players with any questions they might have during the game. This would function by reading in key words given by the user and giving a response based on the keywords. Additionally, we would have liked to get in a leaderboard system that would allow users to save their high score at the end of the game reset. This would function much like old arcade games where an account would not be required to play the game but if the user won at the end, they would be able to input their initials to save their score. We would also have liked to add a little more security to the game, and one of the stretch goals we had to accomplish this was to add Google SSO as a form of verification for the game. The game would not use any account information from google but would force the user to have a google account in order to play the game. This would help prevent robots from attempting to play since a person would need to sign in to access the site. Lastly, we had plans for adding functionality that would allow users to form teams during the game. This would help make the market-simulation more realistic, and would allow for cooperation between players.

For our non-functional requirements, we had a few tasks that we had to ensure were being met as we progressed throughout the project. One such task was that all final changes we had made, mainly at the end of each sprint, needed to be pushed to the master branch of the git repository for The Regis Company. This was useful for the company to take a look at the work we had finished and to do a quality check of our code. We also needed to ensure that the code functioned on all browser types, such as chrome, edge, or safari browsers. As we coded, we also had to do a full documentation overhaul since the pre-existing code had little to no comments in it, so it was our responsibility to add comments to both the old and new code that was written to ensure a higher quality software. Lastly, when adding our new functionality to the game, we had to pay attention to the runtime to make sure we didn't significantly increase the execution time of the code and didn't slow down the game noticeably.

System architecture

Wireframes

For our initial design of the project, we designed a wireframe flow diagram to represent each of the particular UI designs that will occur when each button/tab is selected. We initially designed our wireframes to all be for mobile since that was the focus of our UI redesign, but the UI for desktop will be very similar. In order to better organize our wireframes, we included a basic flowchart as shown in Figure 0 that shows the user flow of how the user will move through the mobile webpage, including each screen that the user may interact with. This way it will be easier to see how each of the buttons relate to each other and how the UI flows as a whole.

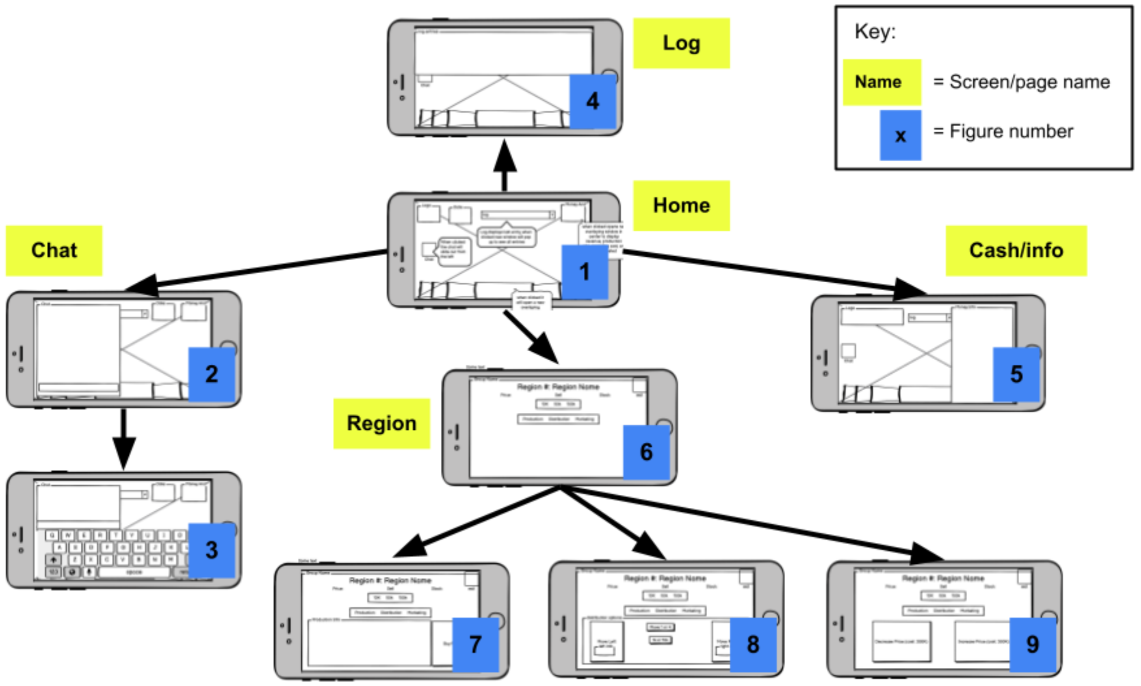


Figure 0. Wireframe Flowchart

To start, in Figure 1 this is the starting screen when viewed on a mobile device. The company logo in the upper left is shown. The date information is next to the logo, the log drop down drawer is in the center then the money and player information button is next to the log. The chat icon is on the left side of the screen and will open a chat box (Figure 2) when clicked. At the bottom you have a carousel of the different region buttons that open the region box (Figure 6) when clicked. The X in the background represents the background image which will remain constant for all screens.

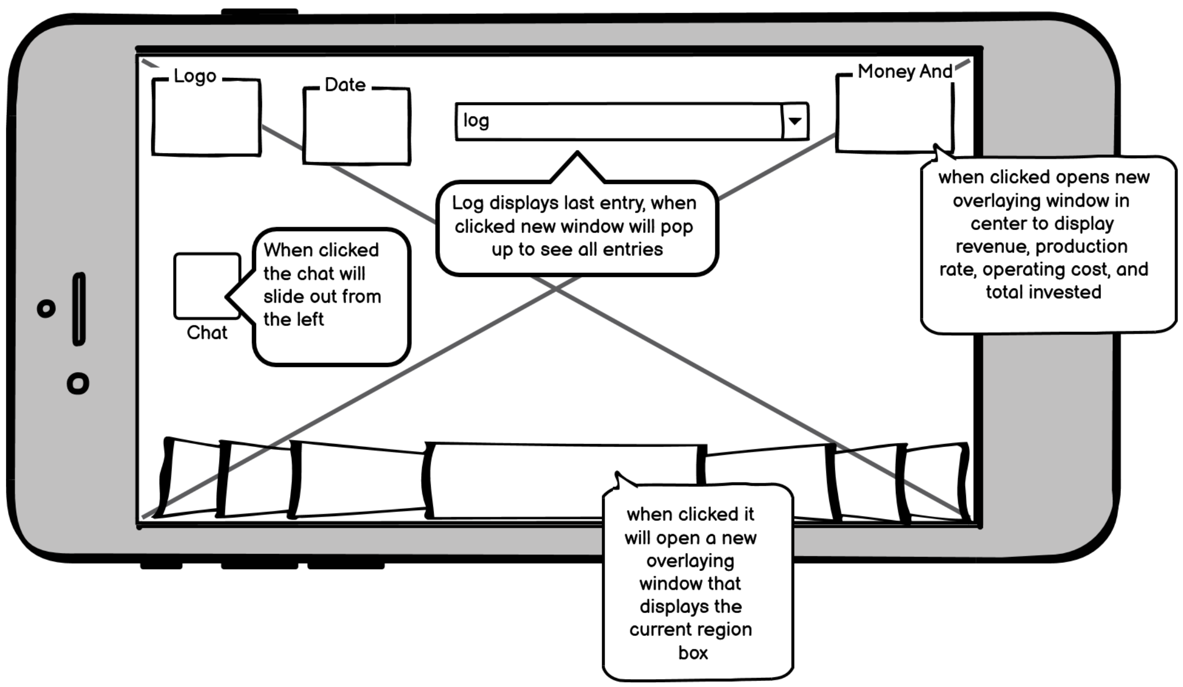

Logo
Date
log
Money And
Log displays last entry, when clicked new window will pop up to see all entries
when clicked opens new overlaying window in center to display revenue, production rate, operating cost, and total invested
Chat
When clicked the chat will slide out from the left
when clicked it will open a new overlaying window that displays the current region box

Figure 1. Mobile home screen wireframe

In Figure 2, this is the screen that will display the chat messages and is accessed by the user when they click on the chat icon in Figure 1. This screen will show the user past chat messages and will allow the user to type a message (by clicking the text field) opening the screen in Figure 3, and send a message to the other players. In Figure 3, This is the screen that displays after the user clicks on the text field in Figure 2. This screen will allow the user to type and send a message to the other players in the chat.

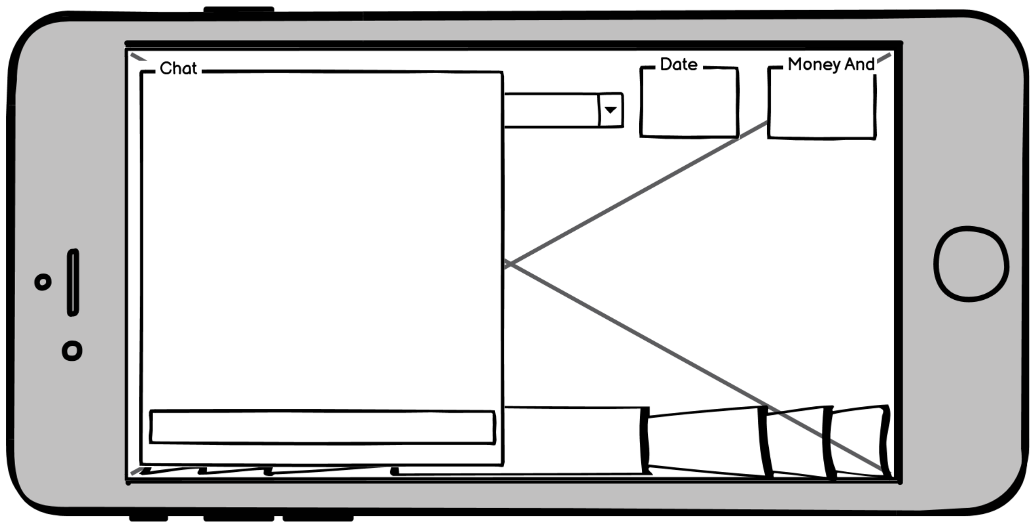


Figure 2. Mobile chat screen wireframe

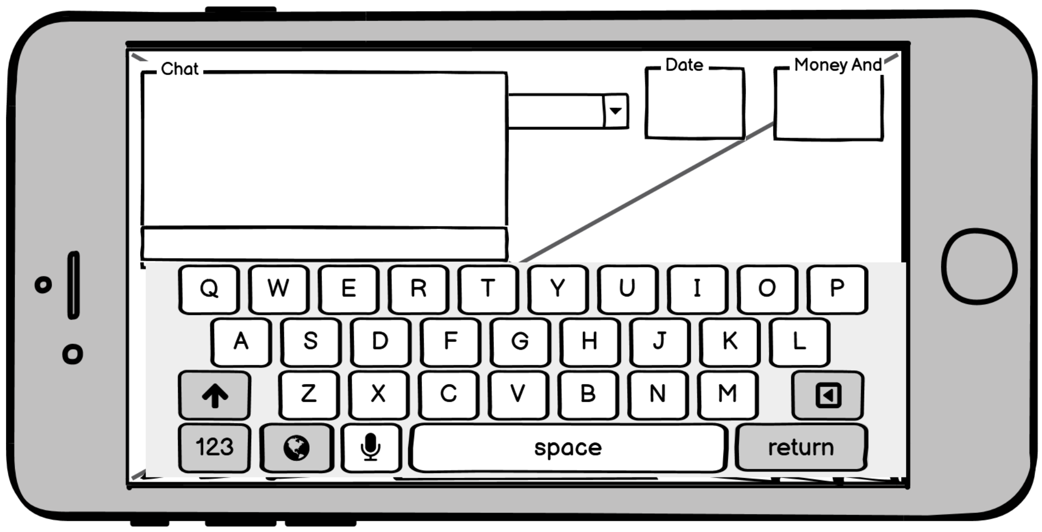


Figure 3. Mobile chat screen (with keyboard pulled up) wireframe

In Figure 4, this is the screen that displays when the log button in Figure 1 is clicked. It allows the user to see all prior log entries of all players by allowing them to scroll through them. Each log entry has an associated timestamp connected to it along with the name of the player who committed the action. The log entries are displayed in reverse, ascending order, with the most recent entry being shown at the bottom. This screen will collapse if the user clicks outside of the log entries box.

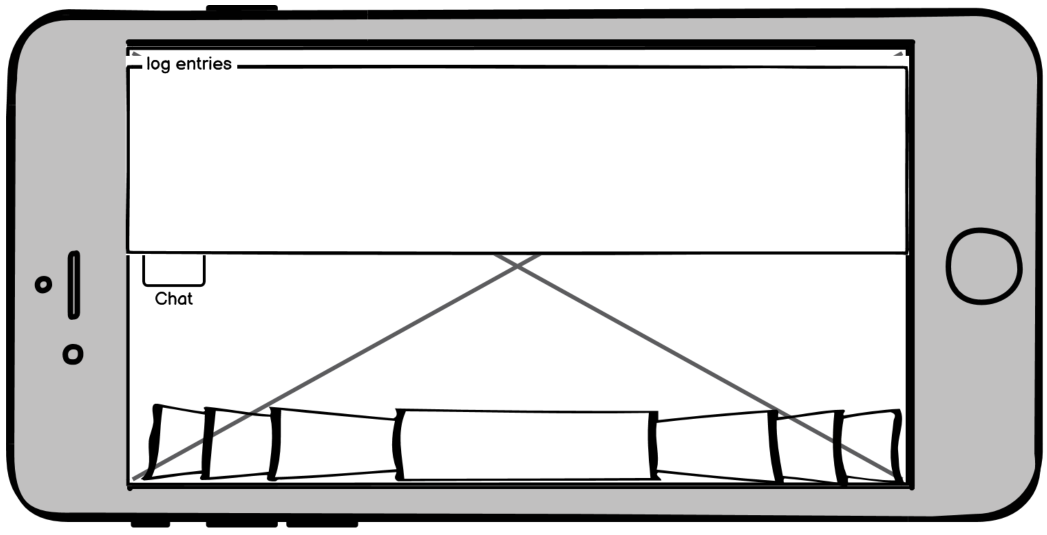


Figure 4. Mobile log screen wireframe

In Figure 5, this is the screen that displays when the money button in the top right corner in Figure 1 is clicked. The drawer contains useful player information such as the player name, total revenue, production rate, operation costs, and total invested stock. This information can be used for in-game features.

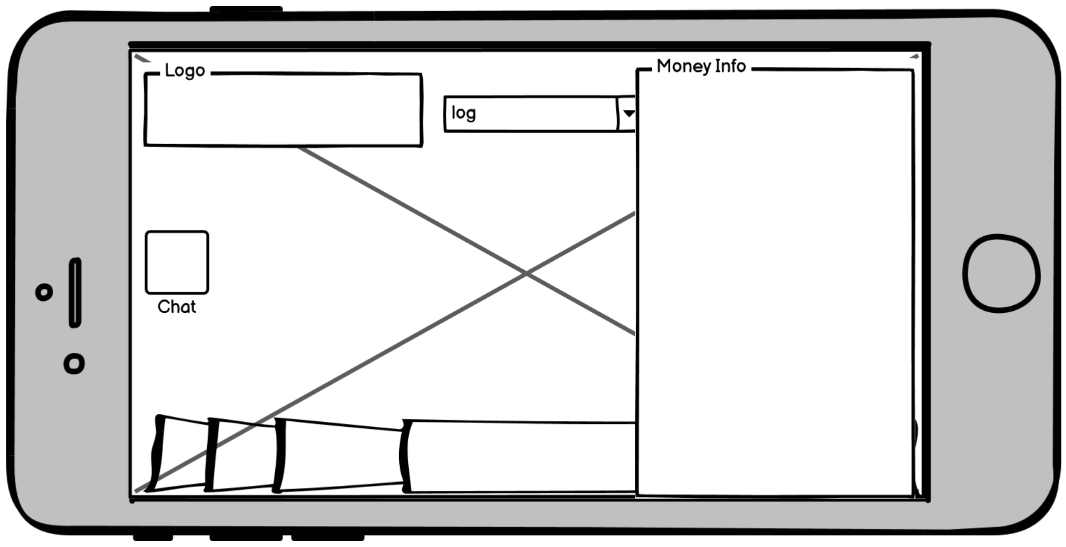


Figure 5. Mobile money/info screen wireframe

In Figure 6, this is the base screen when a region is selected and checked in Figure 1. Here you can see information about that particular region, such as the price of stock, the quantity of stock in that region, and the different tabs you may select. You may also choose to sell your stock in that region in quantities of 10k, 50k, and 100k. For mobile there will be an exit button in the top right so you can exit.

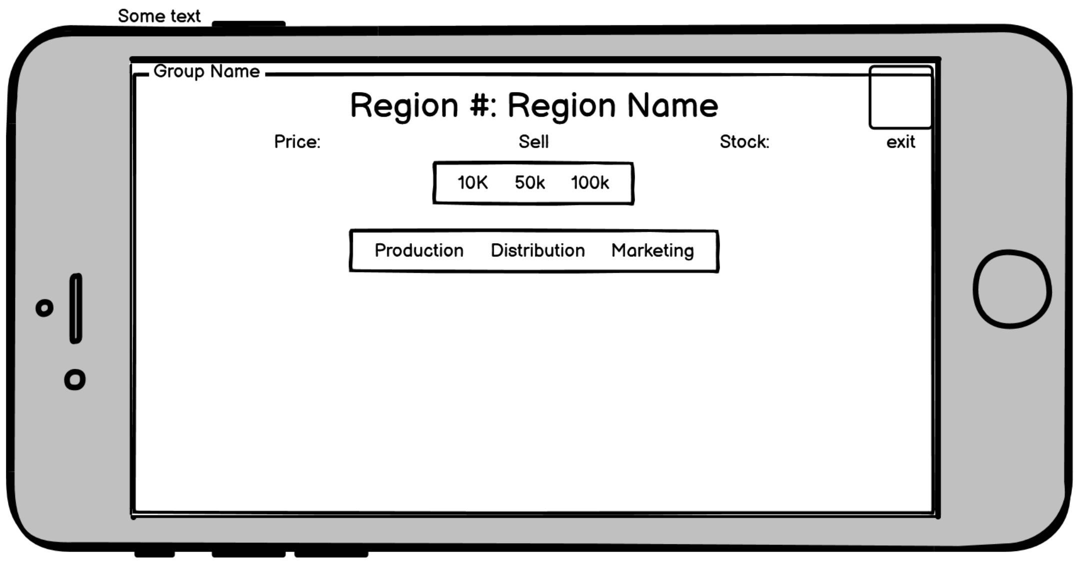


Figure 6. Mobile region box default screen wireframe

In Figure 7, this is the screen that shows up when the production tab is selected. Here you may see the info of a particular region, and also choose to buy a factory in that region to increase stock. Once bought, you may also upgrade that factory, but at a higher cost each time. Then, in Figure 8, this is the screen that shows up when the distribution tab is selected. Here you may move stock to a particular region based on the drop down choice made, and you may select the amount to be moved by the drop down box as well. Then you may either move the selected parameters to the left or right on the map. The buttons contain the region that the stock will be moved to to make sure the proper region distance is correct. Lastly, in Figure 9, this is the screen that shows up when the marketing tab is selected. Here you may increase or decrease the price of stock in a particular region. The cost of doing this action is shown.

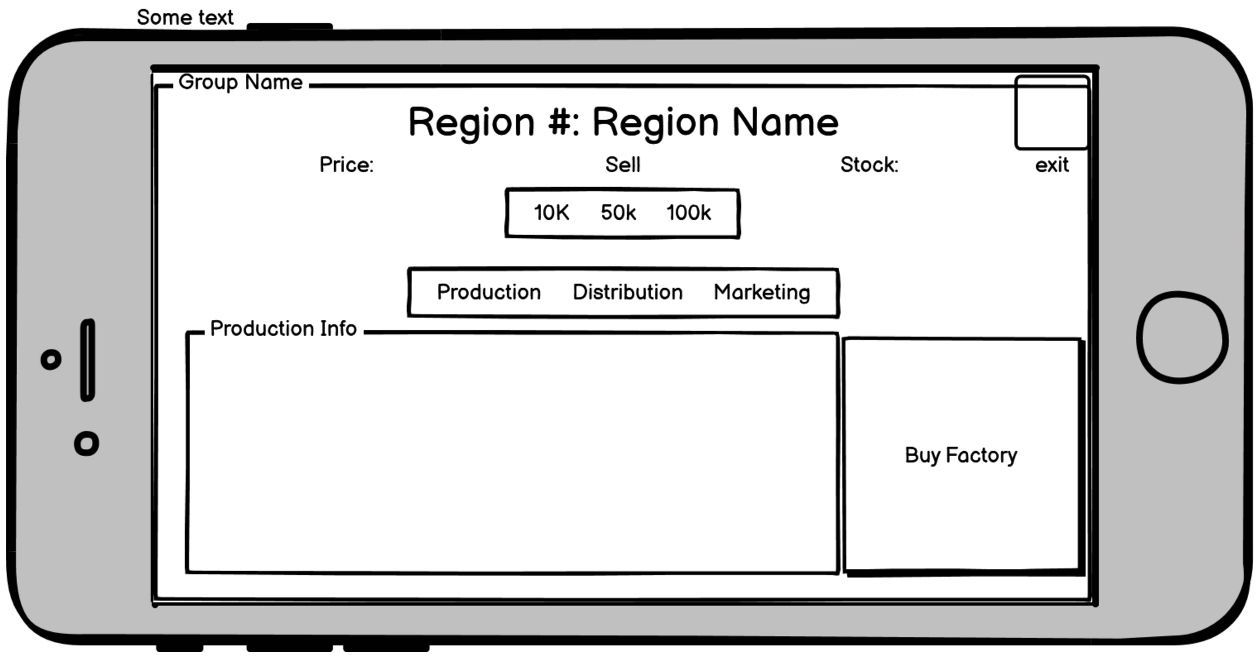


Figure 7. Mobile region production screen wireframe

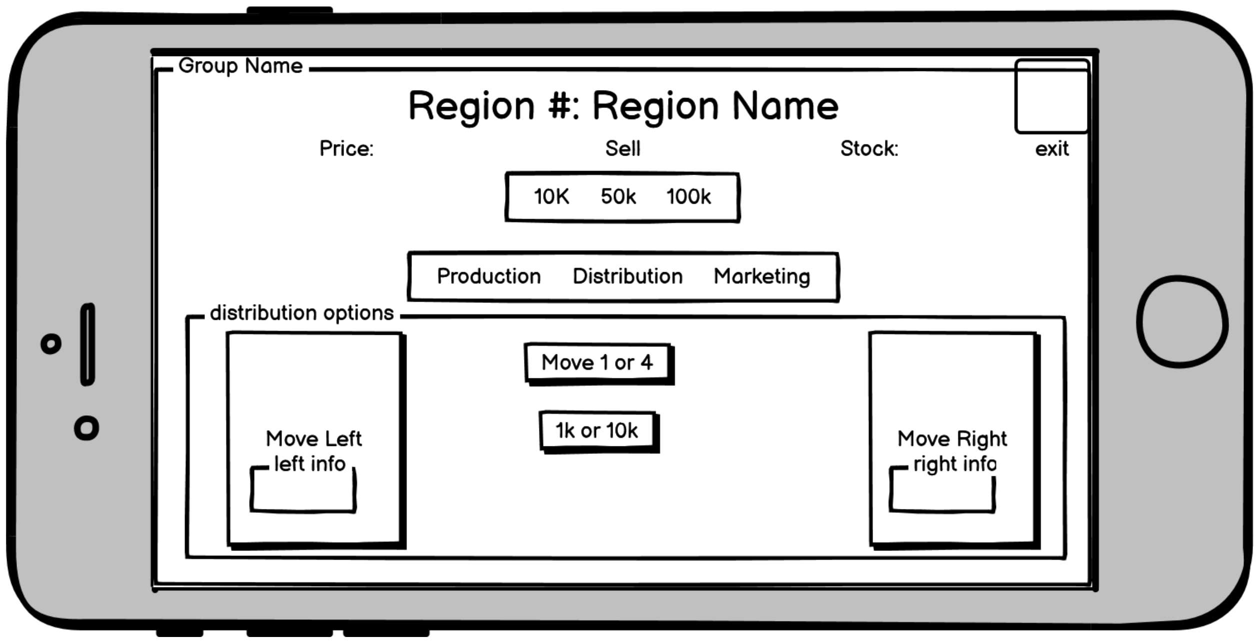


Figure 8. Mobile region distribution screen wireframe

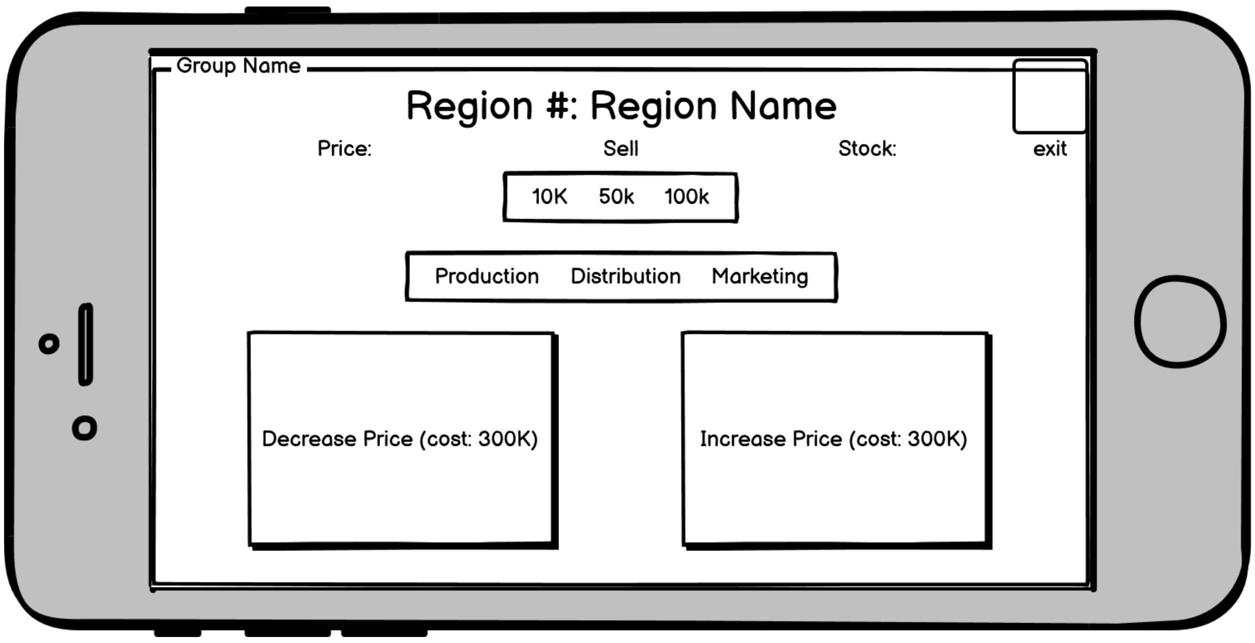


Figure 9. Mobile region marketing screen wireframe

For the UI redesign that we did for desktop, we created a basic wireframe of the only changes that will be made specifically for mobile. Figure 10 is the starting wireframe for the desktop user interface. This wireframe is the only one that differs from the mobile wireframes, all of the other desktop wireframes/UIs will be the same as the mobile wireframes simply scaled properly for desktop. The two key differences are that the logo on desktop will use the full Regis Company logo and on mobile will use only the symbol and the second difference is that the date appears below the logo on desktop instead of to the right of the logo on mobile.

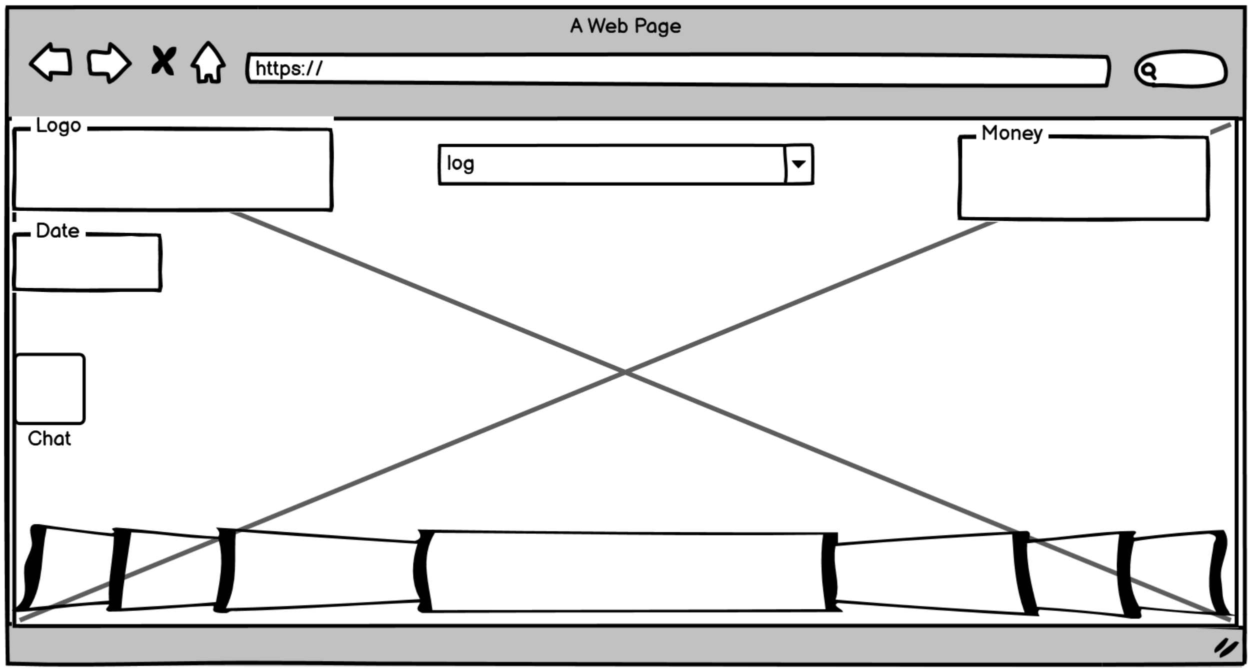
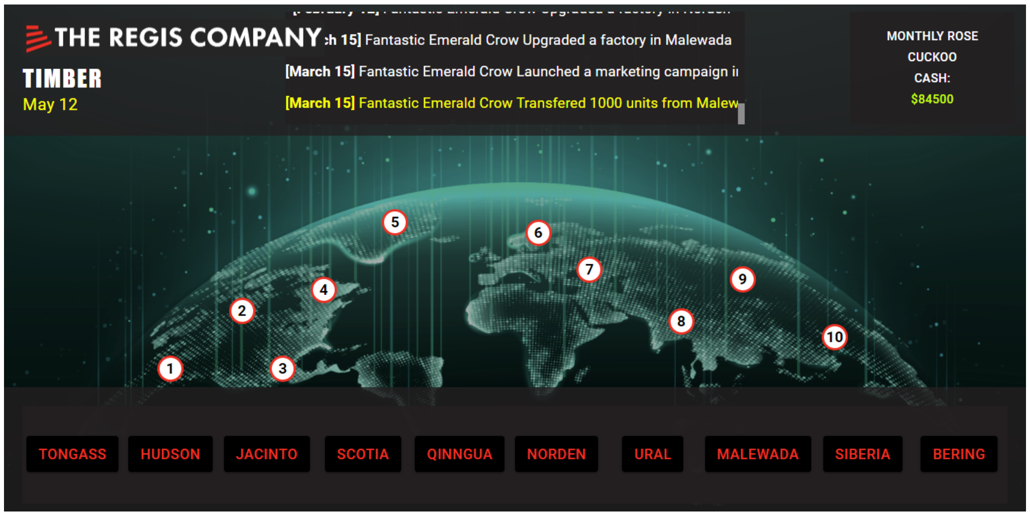


Figure 10. Desktop home screen wireframe

Technical Design

One of the interesting design implementations that we included in our project was the use of pullout drawers and popups to keep the code organized and clean. Previously, during the last project, much of the information was just kept on the main screen, as shown in Figure 11. The log entries were shown on the main screen, the cash and player info were kept always visible, with a few elements kept in an on-hover box that collided with the rest of the design.



:h 15] Fantastic Emerald Crow Upgraded a factory in Malewada
[March 15] Fantastic Emerald Crow Launched a marketing campaign i
[March 15] Fantastic Emerald Crow Transfered 1000 units from Malew

Figure 11. Old desktop home page

What we were able to do was condense a good chunk of the information kept on the main screen into drawers to keep the less needed information out of constant sight on the main screen. In Figure 12, the UI redesign for the desktop is shown, and how the buttons and information have been kept out of sight to keep things looking much cleaner and more open. The previous information is now kept in their respective drawers that can be opened by clicking on their respective buttons. In Figure 13, the UI redesign for mobile devices is shown, and the size changes have been accommodated for the small screen that mobile users will be using, which will allow for a much easier time clicking on the buttons on a much smaller screen than a desktop will have.

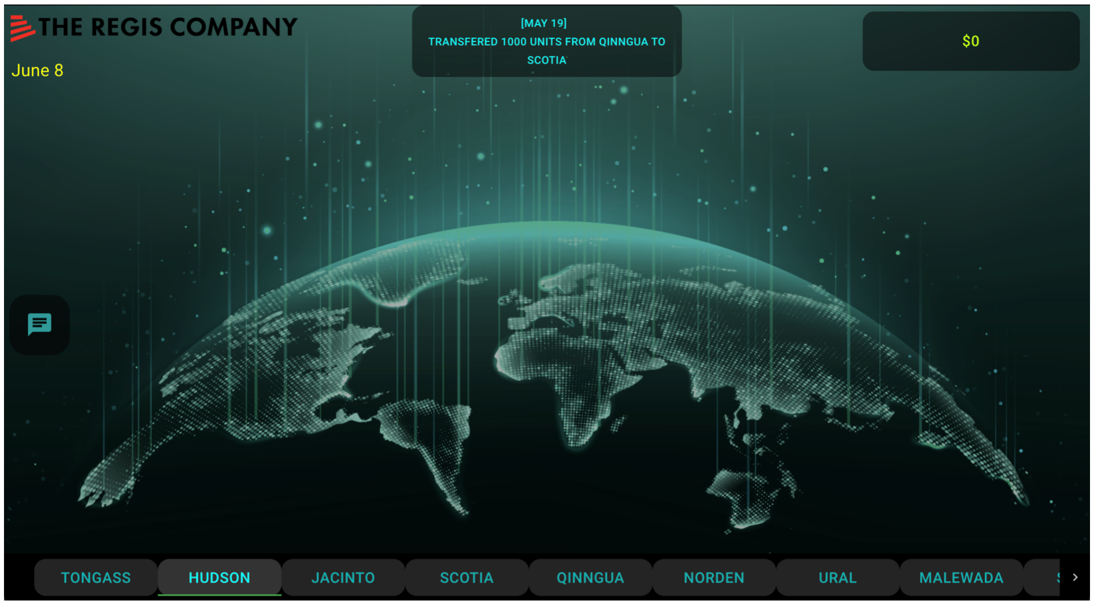


Figure 12. New desktop home page

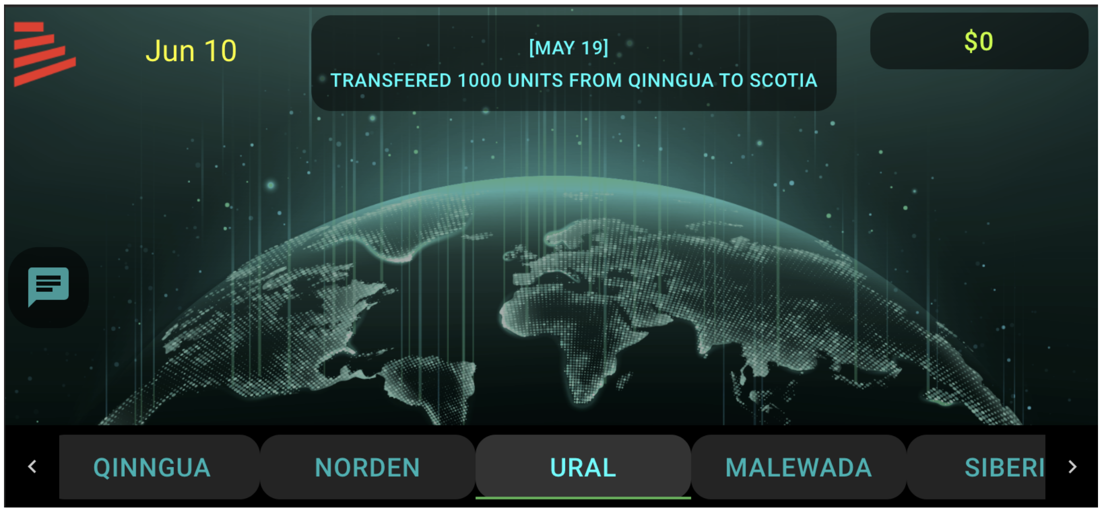


Figure 13. New mobile home page

Another specific technical feature that is interesting with our design is the carousel region buttons at the bottom of the screen. This feature allows for the list of region buttons to be scrollable so that not all the buttons have to be displayed at once for them to be accessible. This was especially needed for mobile devices as there wouldn't be enough room to have all ten region buttons displayed at the same time. We tried using a few different methods to implement this. At first we tried using a traditional carousel but ran into a number of problems. The main problems aside from getting it to function properly at all were scaling the carousel and styling it to match the rest of the theme. The second method we used was implementing a tab bar. For styling we used the Material-UI React framework. In Material-UI there are many different predefined components that you can implement and style for your website. The default tab bar required some modifications in order to function properly for our implementation (it needed to act like a carousel rather than a traditional tab bar). The first thing we needed to do was to make it scrollable. It then needed to function properly when the “tabs” (previously buttons) were clicked. This means the correct region screens still needed to display when their corresponding buttons were clicked. This also needed to scale properly for both desktop and mobile devices. After all of the modifications were made we were able to implement a nice looking feature that both looks good on any device and functions as it should.

Software Quality Plan

For our team’s software quality plan we implemented many strategies to ensure that we were writing high quality software. The first strategy that we utilized was unit testing. When we took over the project there were unit tests that existed from the prior team and in order to guarantee that we were writing high quality code, in addition to writing new tests to make sure the new features we implemented functioned properly, we made sure all prior tests still passed.

The next strategy we used was manual User Interface (UI) testing. This was one of our primary ways of testing new features and functionality. By running the code in a local environment immediately after making changes to the different elements to see first hand what the changes actually did to the UI was extremely helpful when implementing new features. By manually going through the elements we added to the project, the manual UI tests ensured both everything functioned properly with the proper orientation and that no errors/mistakes made it to the final product.

The third strategy we utilized was automated UI testing by using cypress. This software allowed us to create automated tests that check the functionality and placement of the UI elements without the need to manually look at everything with a manual test. These tests function very similarly to the manual UI testing but save some time by automating some of the process.

The next and possibly most important strategy we utilized when working on this project was paired programming. Most, if not all, of our code was written using this method which allowed us to be more efficient when programming while simultaneously preventing possible errors since multiple people were looking at the code being written. Although we did not do a formal code review for the code we wrote, we made sure to go back and document any code that we didn’t document at first (via comments), which provided us with another opportunity to review the code, clean it up, and make it more readable.

This leads into the next software quality strategy we utilized, documentation. When we took over the project there were very few comments throughout the project. In order to provide high quality software we not only added comments to all of the new code we wrote but also went back through and committed the preexisting code. By adding this documentation it allowed for ease of reading and a higher quality environment that ensured in the future more code can be added and implemented without having to reanalise and decipher all of the preexisting code beforehand. This strategy also helped keep the code organized and clean for both future and current use.

The last strategy we utilized in our software quality plan was a client quality check. After we completed features the client reviewed the changes we made before the new code was pushed to the live website. This ensured that the code is up to standards for the client and gave us additional insight on what the client wanted from the product.

Results

With the primary goal of this project being to overhaul the frontend UI and make the website usable on mobile devices, this project was tested thoroughly across multiple browsers and device sizes. From our performance testing results, the project can run on multiple devices and platforms, including mobile and desktop. It can run in multiple browsers, including safari, chrome, and firefox. It also runs on both mac and windows. By testing the website using inspect element in a browser and changing device width and height manually, the website scales properly for many device resolutions including the vast majority of mobile and desktop configurations. In addition, manual UI tests were successful and existing unit tests still passed. Our UI redesign was tested with other people to ensure positive feedback from the redesign.

All primary functional and non-functional requirements have been met and the features that we did not have time to implement were all a part of our stretch goals. Among these features include the chat feature, although a frontend prototype was developed that can be used once support for chat is added in the backend. The smart help chat bot, leaderboard system, Google SSO, and ability to form teams were also not implemented due to time constraints.

Lessons Learned

Documentation was very helpful when trying to decipher the pre-existing code in order to determine how different components of the webpage were displayed and how they functioned. Specifically in the regionbox element of the existing code it was very difficult to figure out what the different pieces of code did, to solve this we ended up deleting the section to see what went away then commenting it after we put it back to determine functionality.

The agile process was helpful for organizing the tasks that need to be completed during the current sprint. By breaking down the project into smaller components the agile process was an effective strategy for completing our big project. This process allowed us to work on individual components of the UI one at a time, and made sure we coded and tested it fully before moving on to other components. Overall the agile process helped keep us focused and on track for completing the project.

When working with Javascript we discovered that we needed to be careful with what you type and the changes you make because the code will not always display errors when it compiles, but it will break during runtime instead. When implementing the hardcoded chat messages there were a lot of formatting issues and errors that still compiled in vscode but when the code was run in the local environment it would break and have to be deciphered in the console of the inspect element.

Communicating with the client was very useful in outlining exactly what needed to be done, and helped us gain knowledge of how to properly get the project done rather than just guessing and having to fix it later. Our client told us about not doing in-line styles for the UI which was very useful since we would have had to go back and change all of our in-line styling if they didn’t tell us about it

React was a useful javascript framework when implementing web page UI when paired with Material UI. This development framework allowed our team to break up UI elements into different components and was helpful for organization and styling. In order to develop an understanding of the different components and how they are implemented our team found researching the Material UI components, documentation, and code snippets to be very helpful.

Future Work

Among features that could be added in future development of this project include all the stretch goal features that we did not have time to implement mentioned above in the results section. The easiest to be added in of these is the chat feature since the frontend prototype has been developed.

More complicated features could also be added in the future like having more information displayed on the map(e.g. factory production) for each region, as well as having animated visuals in-game for a user's factories or when the user buys/sells/moves stock. This would be a more difficult and time consuming feature to add as it is as much a graphic design challenge as it is a software development one. Additionally, a useful feature that should be added in the future is a starting display with a visual of which regions the user owns a factory. This should be an easier feature to add as it could simply be a dialog that pops up when the user joins the game from the website or a simple color indicator on the map.

A more robust map component could be added that would allow for a more visually interactive experience in the game. For example, each region on the map could be clickable and the region box component could be brought up from there for a more intuitive user experience.

Appendices

Once the project was fully turned in to the client, the product was fully functional with a few exceptions. With our skeleton chat implementation, it will be an easy plug-in for the company in the future, the only future requirement that will be needed is that the server needs the ability to read in the chat log and return it to the website, which the company can complete at any time. Currently there is an issue with moving the stock to different regions which is caused by a back end error. The stock will disappear if stock is moved, there is error handling for it but currently it is not patched. All the code has been commented to allow for ease of readability which also allows for easy additions in the future. A function that did not work fully was the scaling for ipad devices. This may not work as it should due to the screen size. We implemented a mobile UI, a desktop UI, and a large desktop UI, but no resolutions in between mobile and desktop. The way we achieved the was by pulling screen resolution to check device type (if the resolution was large (or very large) treat as desktop, if the resolution was small treat as mobile) but a third medium UI was not created (the project currently treats ipad resolution as the desktop UI).