The CSA Survivalist Project Report

Kenneth Chan……..……..…...1009837
Jason Chow……..….1003377
Peter Hudel…….....1012673
Jason Nguyen…….....…1013950
Dixant Patel……1007555
Mitchell Van Braeckel……..……1002297

Table of Contents

Section A: Introduction

The Client

The client who requested this project is the professor of our CIS*3750: System Analysis & Design in Applications class, Dana Rea. He commissioned us to build this application and took the director of engineering role, whom we report to. He provided the specifications of the assignment, as well as selecting the team to complete this task. Other stakeholders may include, but are not limited to, other faculty from the School of Computer Science department at the University of Guelph (UoG), as well as the teaching assistants (TAs) for this section of CIS*3750.

The Team

The team who will be referred to in this article consists of six people: Kenneth Chan, Jason Chow, Peter Hudel, Jason Nguyen, Dixant Patel, and Mitchell Van Braeckel. A detailed description of each person can be found on our group wiki (*see Appendix A1*).

In order to function cohesively as a team, we assigned roles to each person by considering factors such as technical, communication, and documentation skills as outlined in their personal biography. As the semester progressed, these roles evolved and everyone would take on different tasks that they were comfortable with. These roles served as guidelines that allowed us to organize and delegate tasks accordingly.

Executive Summary

For new and experienced members of the university alike, it is often difficult to find the information that they want. Webpages are often disorganized and scattered which makes navigation tedious. It is typically easier to use Google to find the page that you want, rather than accessing it via the website. The current design is unintuitive and inconsistent, causing confusion even when doing basic tasks. Information a student wants to know, such as how to get to a class, is usually buried behind small links and buttons that are irrelevant to the current

inquiry. Why should the student have to go through four different links just to access basic information like the parking costs on campus? Our university's slogan is to "Improve Life", but how can this be accomplished without making students' lives better first and foremost? The *CSA Survivalist* application was developed to solve all of these pressing issues for our user base at the University of Guelph.

The goal of our application is to provide all members of the university with a way to access information directly relevant to students in a well-designed and intuitive desktop program. To do this, our application has three core features. First, it provides students a way to easily discover services, events, and general information about the university that is relevant. Second, it allows the user to set notifications for events and create reminders. Third, it offers the ability to maintain community content within the system through groups or by updating information about the university. Unlike the current university website, this application was made with the student population in mind, providing them a centralized location to perform these three key tasks.

This program not only helps all students at the university, it also helps our client. By providing easy access to this information, staff will not be bombarded with questions that could have been answered by looking online. In turn, this will reduce email and call center volume, allowing these departments to focus on other ways to make student life better.

This document outlines all the steps our team followed in order to create this application, providing the reader with a summary of how we improved student life as well as an analysis of the work that we completed.

Changelog

Here, any changes to our previous checkpoints will be outlined.

- *Checkpoint 1*: Group Biography — Added pictures of each team member.
- *Checkpoint 2*: User Epics and Stories — Added an additional user epic and two user stories under the Eddie Cruz persona to cover missed functionality regarding events and notifications. Also added definitions to persona groups.

User Groups, Personas and Use Cases

Before any actual development of the application could occur, some preliminary steps were required to get organized. To begin, we performed a brainstorming session which contained a list of all potential features that our application could have. During this session, many features were discovered to require a map in order to organize related information desired by a student in a centralized location. In addition, it was noticed that these features could be organized on the map itself as embedded text, where the user would click on a location and the map displays the text. Also, a separate section dedicated to information about the campus, as well as a way for people to join groups, was determined to be necessary. By performing this brainstorming session, ideas that complemented the three core features were identified and used to develop more robust software. Similarly, potentially hindering ideas that were not necessary to the application were discarded, otherwise it would lead to increased cost and feature creep. At the end of this session, the team created an outline of the application's required features. This outline improved our understanding regarding potential users and their actions within the application, which made the creation of personas and use cases easier.

Since the idea generation task was completed, the next activity involved the development of user personas. These personas were a diverse set of people that described potential users of the application with details regarding their needs and desires. These are used as the basis of user stories and user epics that later become the foundation for features that will be included in the final product. After analysis, a major discovery was identified: a significant overlap of persona desires. For example, Bob Jones, Agatha Thornbrough, Dirr T. Dan, and Peter Picky (*see* *Appendix A2*) all wanted information related to something one could find on the campus map. As this was the most requested feature, it was decided that an emphasis on this feature was necessary for the application. Similarly, Lianna Booth, Casper Shad, and Peter Picky (*see* *Appendix A3*) all wanted information about clubs or student life, which meant that group functionality was a priority. In these cases where significant overlap was discovered, the team would heavily weigh the creation of a feature to satisfy the identified desires.

Once all user epics were fleshed out, two fully dressed use cases were made to illustrate something that the average user might want to do. A strong grasp of the exact flow of how an individual would interact with the core features was necessary, in order to determine what was necessary for them to complete these tasks within the application. These use cases were

designed with user epics and the needs of the user in mind, so a deeper understanding of how they will interact with the system could be established. The first use case (*see Appendix A4*) was a way for the user to manage notifications that were sent by the application, whereas the second (*see Appendix A5*) was a content moderator managing a group that they created.

Overall, the implementation was affected by this preliminary work and the paper prototypes looked into how users would interact with the application to perform these common actions. The paper prototypes were modelled after the use cases and during the usability testing phase, users' interactions with the proposed application were observed. Therefore, it was important to fully develop these initial documents. The final implementation was based on the prototypes; however, some aspects of our preliminary work were not implemented in our final product. For instance, the map is missing from the final product and it was also decided in earlier discussions that the map would not be prototyped because this feature would be covered by an external API. Furthermore, a few of our personas, namely Adalrich Schröder and the disabled user group (*see Appendix A6*), wanted accessibility features, but our final implementation did not include this functionality. It was decided that these features would be released in a future update because time and resources were lacking due to unexpected and unavoidable circumstances.

Requirements

After users were analyzed and their possible interactions with the system were identified, the team composed a requirements document (*see Appendix A7*). These requirements were written to mainly support the three core features of our program. In order to support the users' ability to discover services, it must be noted that many user stories involved a campus map. As such, this feature was deemed vital to the application. In terms of managing notifications, a chronological feed was chosen in order for the user to quickly and easily view all alerts. Moreover, a calendar was determined to be necessary to facilitate an easy way to view, create, and edit future events. In order to provide users a way to maintain content within the system, a "groups" page was required. Here, approved members of the community would be able to create clubs for other students to join. Some users will be appointed as content moderators or as administrators to ensure that the content is curated.

The requirements document was prioritized and categorized according to the MuSCoW method in order to enable iterative development. As a result of using this method, the team was able to focus development on requirements with high priority. Core features of the application were put at the top of the document with dependencies detailed below them further down the ordered list. In other words, requirements with a higher ID depend upon those with a lower ID. By prioritizing requirements in this manner, it enabled the team to build a minimum viable product (MVP) at the beginning of development. Furthermore, this was beneficial to the team because at least the MVP was complete even though unforeseen circumstances such as a pandemic impacted the remaining development of the product. More detail about measures taken in response to this external factor will be provided in the implementation section of the document.

Section B: Prototyping

Introduction

For the next phase of the project, the team began working on a very rough draft of what the final application would look like by creating a paper prototype. A paper prototype is extremely useful because it allows for quick iteration of certain UI features in order to swiftly test for usability issues, as well as dismissing attention about non-functional, aesthetic design elements that are unrelated to usability, such as colour and imagery.

Creating the Prototype

Before actually building the paper prototype, it was necessary to decide which functionality was going to be featured in the usability tests. The two fully-dressed use cases were designed for this exact purpose, so the paper prototype modelled these use cases. In addition, these use cases were designed with persona and requirement coverage in mind.

First, a home screen was designed so a user could navigate between the two use cases and different pages of the application (*see Figure 1*). Also note the presence of a menu bar at the top of the user interface (UI) where a user can access their settings and profile (*see Figures 2-3*).

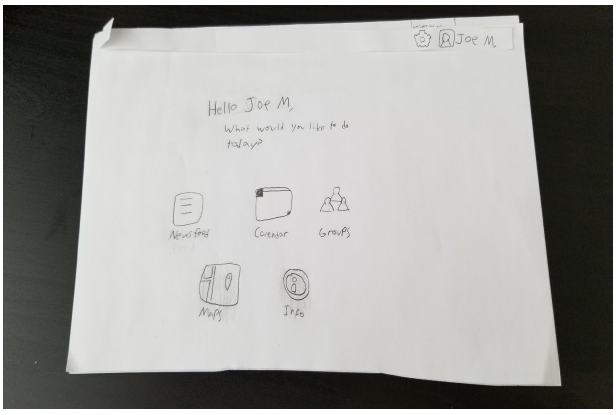
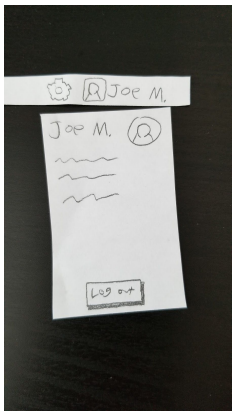


Figure 1: Home Screen



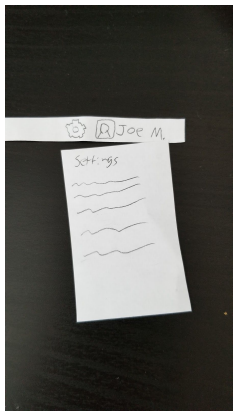


Figure 2-3: Menu Bar Setting & User Options

Now that a user is able to navigate between screens of the application, design of the actual use cases could begin. The implementation of this first use case would be known as the *newsfeed* in our prototype. It went through a few iterations before the team could reach the design present in the prototype (*see Figure 4*).

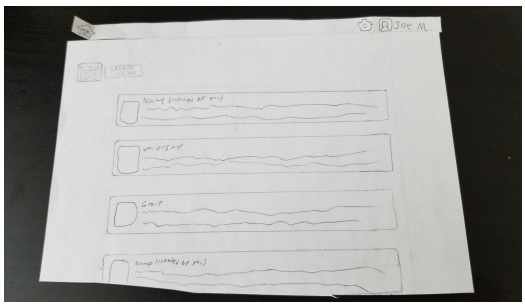


Figure 4: Newsfeed

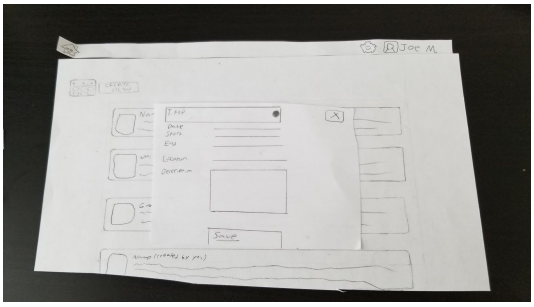


Figure 5: Creating a Notification

After some preliminary testing of the newsfeed, the team realized that this use case failed to cover another core part of the application, the calendar. Similar to the newsfeed, the calendar also allows the user to view, create, delete, and edit notifications and events in an alternative manner. The team realized if a user wanted to modify a notification far in the future, they would have difficulty doing that in the newsfeed, so the calendar offers a solution. Note the pop-up for creating a notification is identical to the one that appears when doing so via the newsfeed (*see Figure 5*).

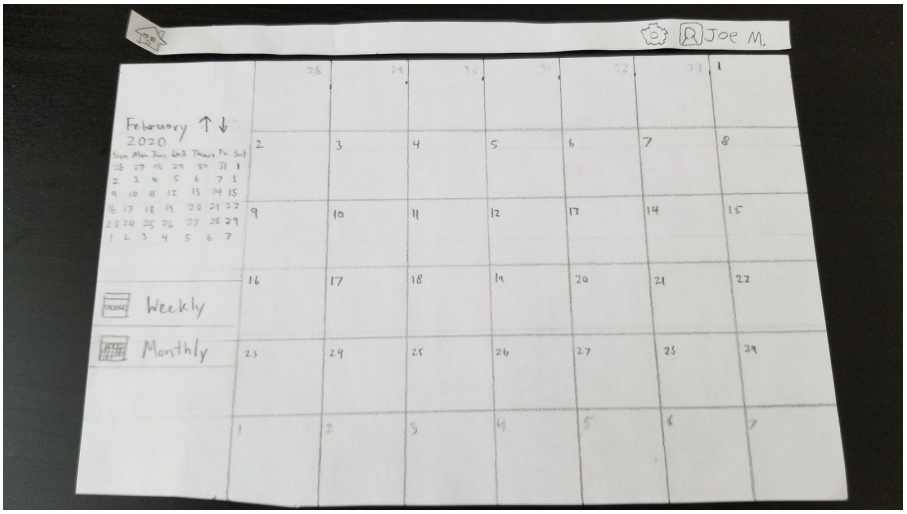


Figure 6: Calendar

Once the calendar was finished being built (*see Figure 6*), design of the second use case commenced. Although the main bulk of users are undergraduates, there will be some people who need to moderate content on the platform. In order to accomplish this, a discovery section for the user to find groups was required.

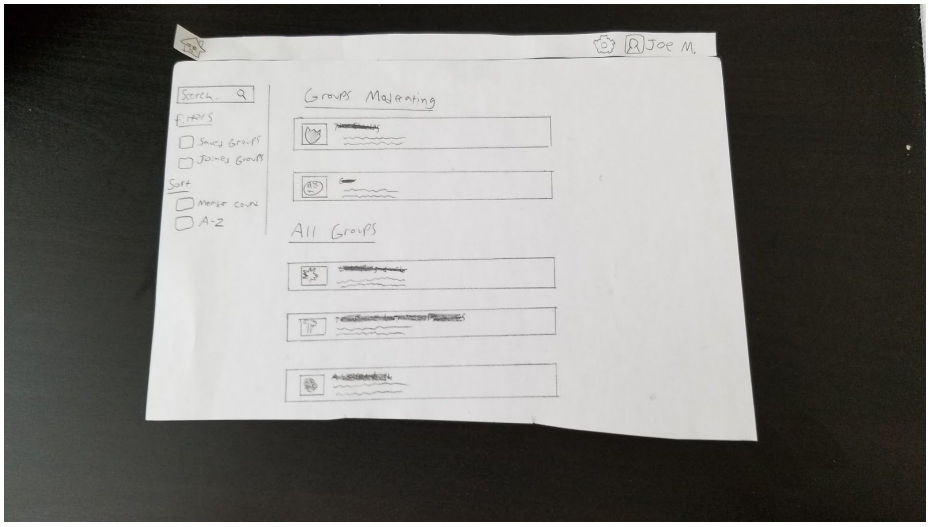


Figure 7: Group Search Page

The user will first navigate to the group search page from the home screen (*see Figure 7*). Then, they can click on a group to bring up the group page which allows moderators to edit content related to the group (*see Figure 8*).

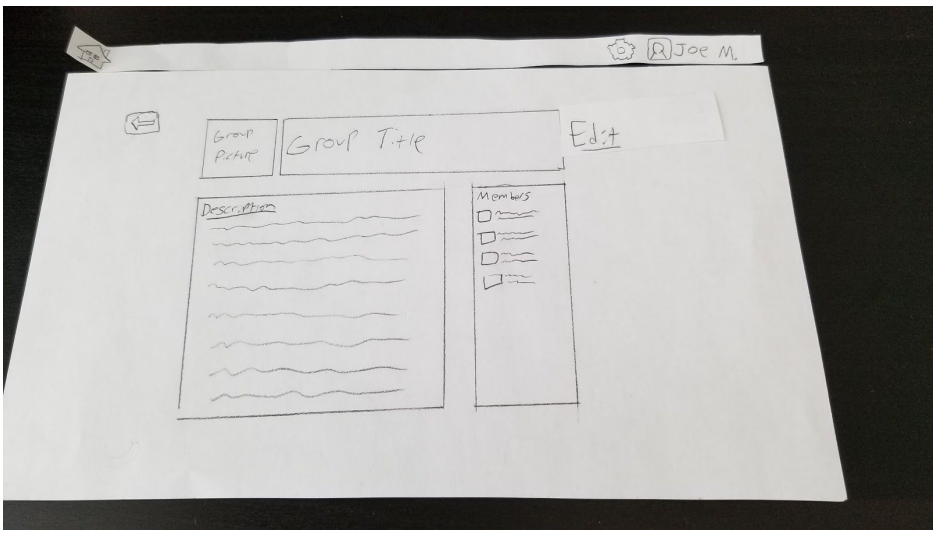


Figure 8: Group Info Page

Similar to the newsfeed, some preliminary testing within the team was conducted to verify that the UI was constructed well. If anything seemed unintuitive, it was back to the drawing board to redesign it while taking previous discoveries into account.

Designing the Prototype

Before the actual design phase of the interface started, some decisions on preliminary factors that would affect all aspects were required. Since this product was a university commissioned application, it was decided that it would conform to the basic theme of the University of Guelph. This would include Gryphon symbolism and key colour combinations utilizing mostly black, red, white, and gold (*see Figure 9*).

Figure 9: Menu Bar

Additionally, aspects of the current university websites were considered in order to improve them while retaining some degree of familiarity. By reviewing user stories, the team realized that a large group of people may not be very tech-savvy. Consequently, aspects of the interface would need to be as clean, simple, and easy as possible. The current university website is difficult to navigate, so the UI/UX implementation was designed with the goal to also be as intuitive as possible to compensate for past mistakes. In turn, large buttons, symbolic and easily understood icons (*see Figure 10*), and interfaces that provide the user with a low number of options to limit unwanted error conditions were all necessary.

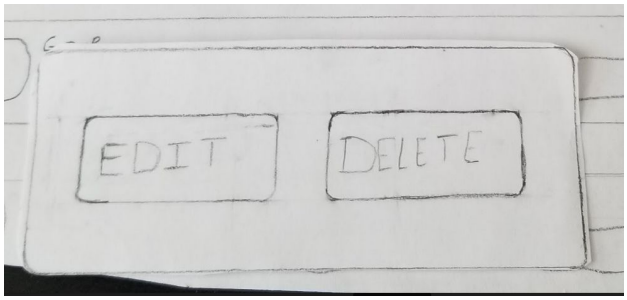


Figure 10: Buttons That Appear When a Notification is Selected

The general approach taken while constructing this paper prototype was, “take like an artist”. In other words, we would incorporate elements of well-known and successful UIs into our application. For example, the calendar part of our prototype is heavily inspired by Microsoft’s Windows 10 Calendar application and the Outlook calendar. In addition to each members’ personal experiences using the calendar application, it was previously known that these UI features have already been thoroughly tested by Microsoft. Thus, the team felt it was a good solution that could be used as inspiration and as an example to model our own design ideas. Once the base UI was developed, it was further specialized to give the calendar a unique identity as part of the *CSA Survivalists* application. For instance, an easily identifiable aspect is the trademark UoG colour-scheme.

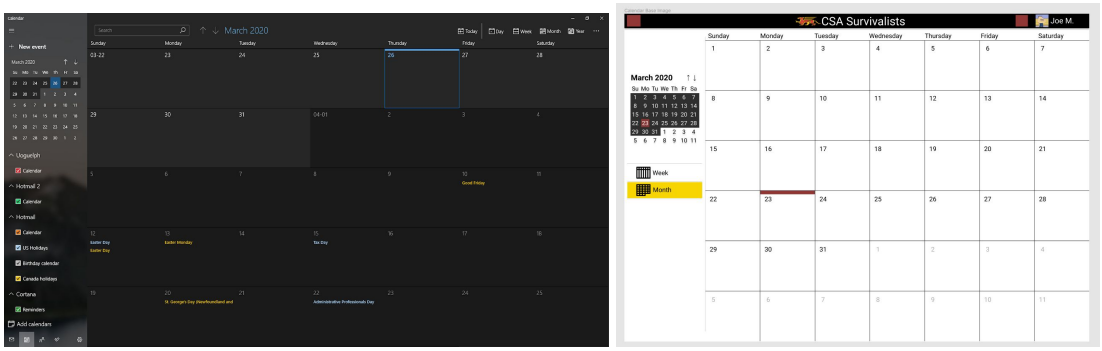


Figure 11: Comparison Between Windows 10 Calendar (left) and CSA Survivalist’s Calendar (right)

This method of “taking like an artist” was also used for the groups management page. A solution for searching for groups already existed through GryphLife, another university-owned website. The team unanimously felt that this was a good minimalistic design of a groups page, so a base design was inspired and created from that.

In terms of designing how a group information page appeared, the group page from GryphLife was referenced and toned down slightly to achieve our goals. This “taking like an artist” is also a good method because users may recognize elements that are similar to other interfaces that they are already familiar with. In the case of searching for a group, many students are familiar with how GryphLife looks, so including those aspects can reduce the learning curve associated with the interface. Furthermore, this idea can be extended to the calendar feature, since many of our users will already be familiar with the Windows 10 Calendar, Outlook, or another similar calendar application.

However, it's important to note that this method of designing an interface can occasionally lead to problems. Most noticeably, since elements are being extracted and redesigned from many different resources, the UI can potentially become cluttered and not possess consistency. Fortunately, this was accounted for during the development of our interface. For example, the newsfeed was visibly quite different compared to all other elements of the interface initially. Only after a trip back to the drawing board, it could seamlessly fit together with other elements of our UI. For example, both the newsfeed notification and group elements share similar appearances and functionality, including title text, description text, and an image located in the same areas (*see Figure 12*).

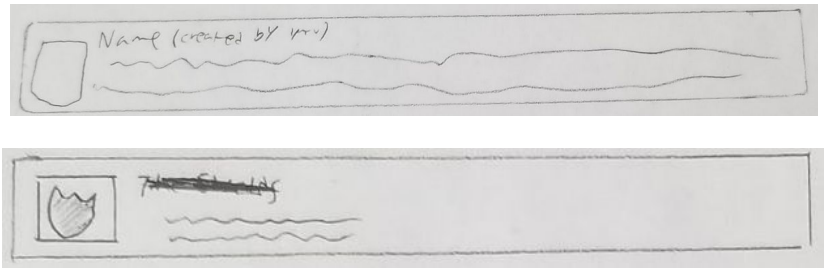


Figure 12: Newsfeed Notification Element (top) Compared to Group Page Element (bottom)

While creating the paper prototype, basic testing of the interface within the team was also conducted to ensure the current solution was sufficiently intuitive. If a team member required clarification, it would be followed up by a brainstorming session to discover how this implementation could be modified to afford its functionality. Through the completion of this activity, the uncertainty that was discovered during usability tests was eliminated. Additional images from our prototype may be found in *Appendix B1*.

Usability Tests

With the design of the prototype finished, the usability testing phase started. In this phase, three different users unrelated to the development team attempted to perform certain tasks that we laid out for them. Based on their interaction and feedback, the UI would be tweaked and modified with improvements. These criticisms became the basis of how our final project would look and function. The format of the usability test is as follows:

1. Ask the user for demographic info, such as name, age, and technological comfort
2. Introduce the user to the application, along with an overview of what the paper prototyping session will cover
3. Brief the user about the first use case and what tasks they need to perform in order to complete the test. As explained previously, the user would be required to create a notification in their newsfeed. To get as much feedback as possible, they will also be tasked to edit and delete a notification, making the test case as robust as possible.
4. Once the user successfully completes the first use case, brief them about the second one. In this case, they are now a content moderator attempting to edit information about a group that they moderate. Similar to the first one, this use case was slightly extended to include providing the tester with the possibility of joining a group, as well as leaving one they are a member of.
5. In each use case, when the tester completed it, they were asked to explore alternate paths that they could have taken to reach the same goal, and get their feedback on it.
6. If the user gets stuck along the way, the facilitator would gently nudge them in the right direction without directly providing them with a means of completing the task.
7. Repeat steps 1-6 with another two other users, all while taking notes of the session.

During the usability testing, there were a few key things that the users did which should be noted. Regardless of technical skill level, each user was able to navigate the application and perform the core path of each use case with little to no guidance. The facilitator rarely had to provide clarification about the system because every user generally understood what they had to do in order to complete each task. The consensus from each tester was that it was all well-organized and detailed. In particular, comments that the interface was easy to use were received because they were already slightly familiar with the design, proving that the method, “taking like an artist”, which was used to build the prototype worked. Another aspect of the interface that users liked was the large button size because it helped with accessibility. This meant that the strategy of building upon the university’s current websites was also successful.

Although the tests went smoothly, there were some unexpected discoveries regarding aspects of the interface. During the prototyping session, a “censor bar” was utilized to hide UI elements that the user could not interact with. Many users thought this was actually a part of the interface and were confused about its existence. If another prototyping session was held, the censor bar would be omitted and those elements would simply not be included (or alternatively just use a blank piece of paper). Furthermore, the session revealed that every user was confused about the term “newsfeed”. The facilitator had to clarify the meaning of the term even though the team originally thought that it was intuitive. Similarly, some users accidentally clicked on the “Leave” button on the group info page because they mistook it as the back button. Evidently, these terms were ambiguous and needed to be revised in order to provide better clarity to the user interacting with the interface. Lastly, all users had some difficulty with editing a notification because the action to initiate the modification of an existing notification was unclear. It was later identified that the root cause of this issue was the lack of any indication that a notification was editable.

Conclusions

As a result of feedback received, it was concluded that some aspects of our application needed modification to improve usability. Also, the team decided against implementing certain criticisms that a user may have given us after careful and thorough analysis. For example, if what came up turned out to be a one-off issue that was not reflective of the application, no modifications were made to the application with regards to that feedback. For instance, only one user had trouble clicking on the calendar icon to navigate to the calendar, which was simply caused by the team not drawing the calendar well for the paper prototype. Every piece of feedback is important; however, the team decided to focus on issues that were identified by multiple users instead.

In the final application, the censor bar that was used to hide unimplemented content during the prototyping session is no longer present because those features have now been implemented. Secondly, some of the button descriptors were changed: “Leave” was changed to “Leave Group” and similarly, “Edit” to “Edit Group” (*see Figure 13*).

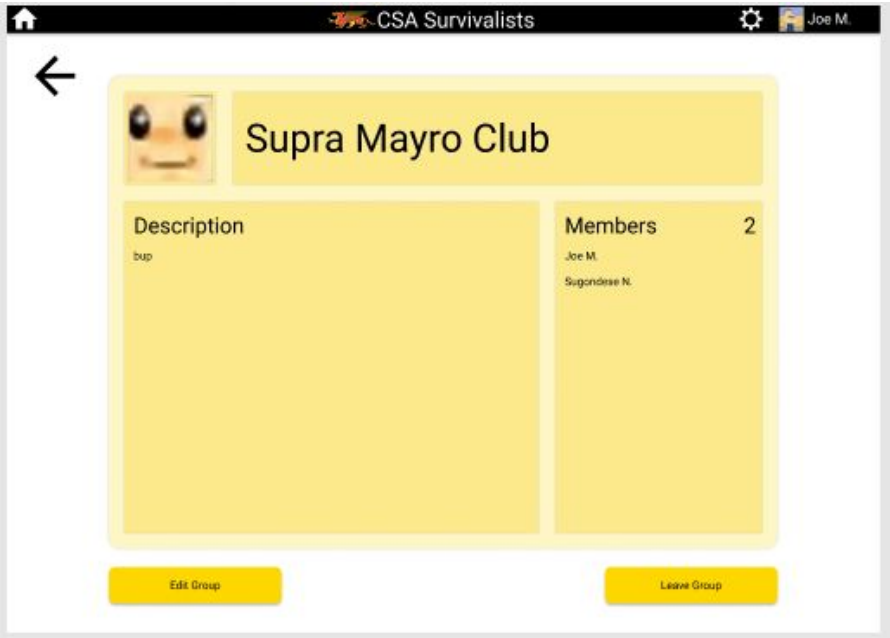


Figure 13: Group Info Page

In regards to the newsfeed, the team unanimously agreed that “Notification Feed” was a clearer identifier and more accurately explained what the page was, helping reduce uncertainty. In addition, a small pencil icon was added to the top right corner of each editable notification in the feed (*see Figure 14*).

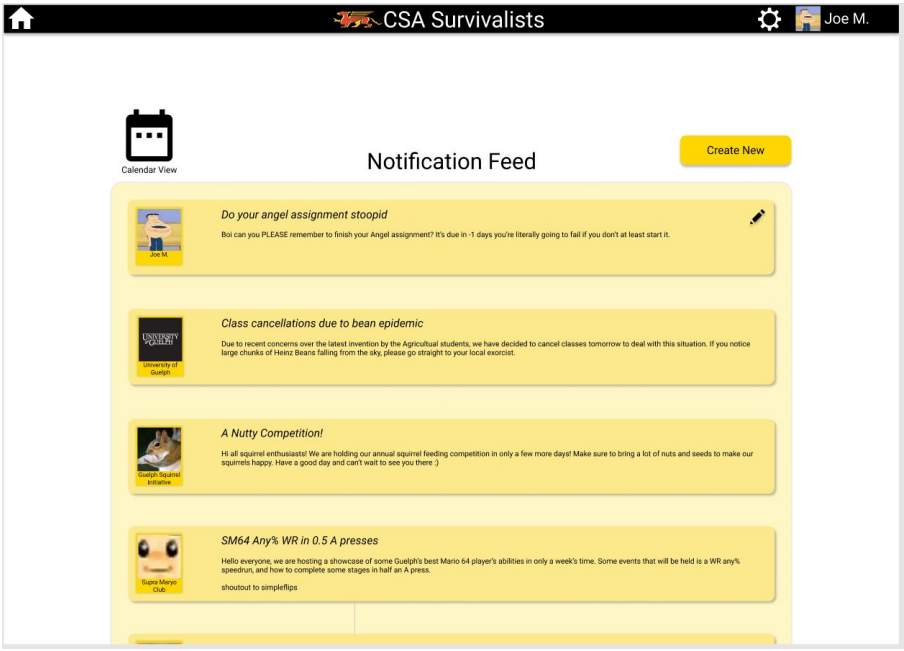


Figure 14: Notification Feed

This was done in response to many users not knowing where to click to edit a notification. To clarify, this is important because some notifications on their feed may not have been created by the user and thus not editable by them. For example, an emergency alert sent by the university would show up in the feed, but users would not have permission to edit it. This ambiguity is removed by adding an edit icon to show which notifications can be modified.

Section C: Implementation

Introduction

This section outlines the implementation phase. Unfortunately, as mentioned before, some events arose that heavily affected the amount of work that could be completed for this project. Just before this phase began, a global pandemic was announced and development halted immediately. The University of Guelph shut down for a week and all courses underwent extreme restructuring to accommodate virtual and remote learning for the remainder of the semester. Therefore, the requirements of this application changed dramatically. Due to this, the team was only able to piece together a high-fidelity prototype and a quick mockup of the login page in React. More information about what was completed will be detailed in the implementation subsection of this article.

Remote Work Strategies

With the design of the UI finalized, implementation could now begin; however, a development environment must first be created to allow the team to build the application. Due to COVID-19, all physical meetings were no longer possible due to safety concerns. Thus, everyone must be able to work remotely and consideration for each member's schedule and obligations must be taken.

The primary form of communication was through Slack, which the team had already been using for online communication throughout the semester. A Discord group was also created to hold video calls if necessary because Slack's free plan did not include screen sharing functionality. This course's GitLab group was used to host and store a repository of the application's code. The Wiki pages were also used to maintain this project's documentation. Utilizing these features allowed the team to share code and other files easily, speeding up development time. Since GitLab only allowed one person to edit a Wiki document at a time (or changes were lost and overwritten), the team decided to use an organized shared folder on Google Drive. This method allowed multiple members of the team to create and edit documents simultaneously. Afterwards, each formal document was rewritten in Markdown format using converters and by hand. Moreover, the team used Visual Studio Code (VS Code),

a lightweight text editor with many possible extensions, to fit development needs. One of the extensions used was to enable multiple members to simultaneously edit files on a host's computer. Using the Live Share extension allowed one person to push changes to the GitLab wiki after finalizing other member's contributions, thus improving workflow and increasing efficiency. Finally, Figma was used to create a high-fidelity (hi-fi) prototype, especially once it was determined that a lack of resources and time would not allow for full implementation of the application in some areas.

Transitioning to a solely remote work setting impacted the group in numerous ways. First of all, switching from face-to-face communication in labs and lectures to remote meetings led to overhead in the beginning, especially because all big decisions were carefully examined in the labs. Future decisions had to be made in an online environment where the benefits of face-to-face communication were lost. Furthermore, working solely online can be difficult due to scheduling conflicts. With the changes due to COVID-19, each member's schedule was affected and contingency plans to produce a product by the end of the semester became necessary. However, the largest impact was how the University of Guelph reacted to COVID-19, where class cancellations and restructuring affected implementation plans drastically. As mentioned earlier, only a high-fidelity version of our paper prototype was possible in the new scope. However, this prototype is missing an important feature, the campus map. As a result, only two of the three core features were actually implemented in some manner, as we decided earlier the map would be coded and not mocked in the hi-fi prototype. Unfortunately, this severely impacted the final application, which was missing a campus map.

Throughout this whole process, the team learned valuable lessons. For instance, no matter how much time is spent on planning, something unexpected can still happen. Nobody realistically predicted a severe global pandemic to occur during development, and this caused the largest impact on the product. Secondly, coordinating between multiple people can be difficult, especially when every team member has commitments outside of school. Scheduling meetings and time to work together as a team was a challenge. On the other hand, working remotely was a valuable learning opportunity because it was a new environment to get used to. Finally, communicating remotely will never be as efficient as physically being in the room with your team members because information sent and received through text lacks the subtleties of communication. Body language and tone of voice conveys additional information

that provides context to words and leads to less misunderstanding. Additionally, messages are usually slower because no one is constantly checking for notifications, which meant that time-sensitive decisions may potentially go unanswered.

Development Environment

After creating a new strategy to accommodate remote work and communication, the development environment could be finalized. The system administrator for the team, Dixant Patel, was responsible for this task. Since he was most knowledgeable in the group about front-end frameworks, he volunteered and was also deemed best-suited for this task. After setting up a Figma account for creating the hi-fi prototype, Node.js was chosen as the back-end, and React as the front-end. A few team members were familiar with React, so it was selected as the optimal way to put together an aesthetically-pleasing UI within the short time available. Once React was installed, the command `create-react-app` would generate the skeleton of the program and development could begin immediately without worrying about the application structure.

As mentioned before, the team planned to use an external API to handle the campus map; however, a lack of resources disrupted this plan. The external API Leaflet (*see Appendix C1*) was chosen because it focused on the creation of mobile-friendly maps. This API would have made the creation of a mobile application easier in the future. Only the longitude and latitude of the campus were necessary to create the map, then waypoints containing embedded text about campus locations could be added very easily. For icons, Styled Icons (*see Appendix C2*) would be used with the React application to include high quality and intuitive iconography that would improve the usability of the interface. By importing premade icons, development time was reduced since the team did not have to allocate resources to create new assets.

Reflecting back, the team felt that this environment was a little difficult to work with. Although all of the team was familiar with JavaScript and Node.js from previous courses, not all members were proficient in React. For these members, this development environment presented a learning curve that made it difficult to contribute code. As a result, these members ended up working primarily on documentation or prototypes. Whereas members familiar with

React were very confident that they could complete coding tasks. For this reason, the number of members assigned to actual code development was not ideal and further contributed to the lack of fully implemented features. Regardless of factors outside of the team's control, the application would have taken longer to implement because members unfamiliar with this framework would first need to learn before actually starting to work. In contrast, the high-fidelity prototype was comparably much easier to design and all members were able to contribute immediately without difficulties.

In order to retrieve the code and run it on a local machine, a few prerequisites must be met beforehand:

1. Ensure Node.js and Git are installed
2. Navigate to the group GitLab page
3. Click the "Clone" button and copy the link under "Clone with HTTPS"
4. Open a local terminal and type `git clone <copied link>`
5. Open the directory that was cloned and navigate to the "csasurvivalist" folder
6. Open a terminal in that folder and type `npm install`
7. Once installation finishes, type `npm start` to run the application

Technical Design

Now that the development environment was set up and a basic skeleton application was created, technical design for the application had to be finalized. Obviously, this would be based on the prototypes and usability tests performed earlier in the semester; however, planning the control flow for the application was still necessary. Thus, a basic flowchart based on past activities was developed for certain areas of the application. These flowcharts outline a user's interaction with the application at a high level.

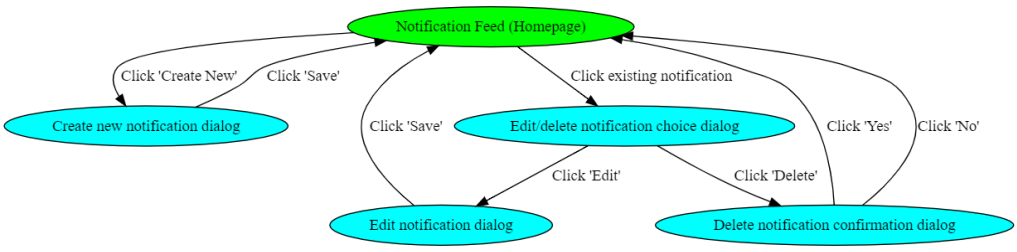


Figure 15: Basic Notification Flowchart

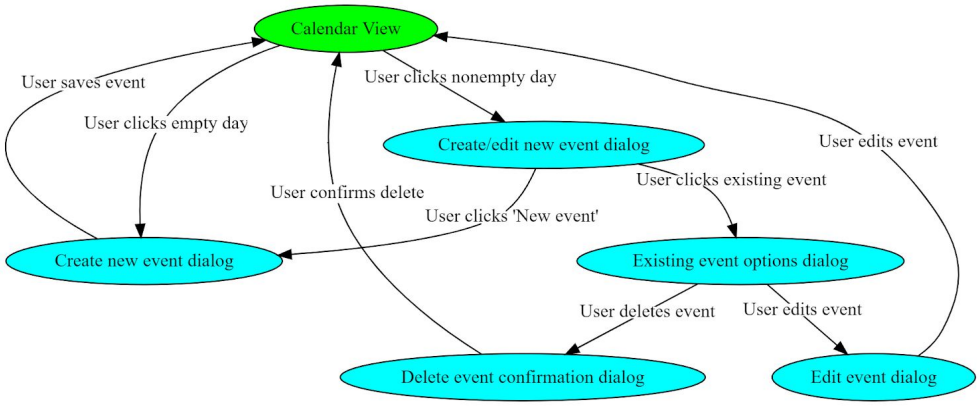


Figure 16: Calendar View Flowchart

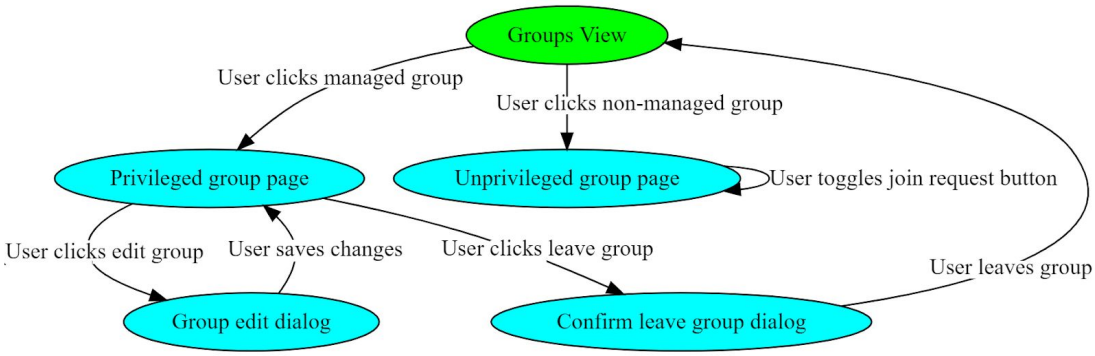


Figure 17: Groups View Flowchart

Based on these flowcharts, an in-depth data-flow diagram was composed to contrast the previous flowcharts, so this is a low-level diagram that also integrates how data storage occurs within the application. Since this was developed in React, it did not make sense to create classes to model data. Instead, each “screen” in React is a separate entity that the user interacts with, which is called a component. For example, the Map screen is a component that the user interacts with through actions like zooming and placing waypoints. A waypoint itself is not a class because it is simply updating the state of the Map component. The elements that are components have been labelled accordingly in the data-flow diagram. In addition, there are containers that manage the program state.

Furthermore, certain elements may require instances of a JavaScript object, but are not necessarily a class. For example, a Student “object” may require elements, such as a first and last name, but do not possess methods or inherit from a base object. In essence, these are a dictionary of key-value pairs that store information in a collection. These objects only contain

default JavaScript data types and no user-defined objects. For clarification, an Event object in the data-flow diagram only contains DateTime and String types in their dictionary, no additional objects. This is important to note because there is no hierarchy of data, which is something not typically seen in a standard object-oriented (OO) application. Below is the full data-flow diagram that illustrates every facet of user interaction with the application (*see Figure 18*). Note that unless specified, everything in the diagram is a component.

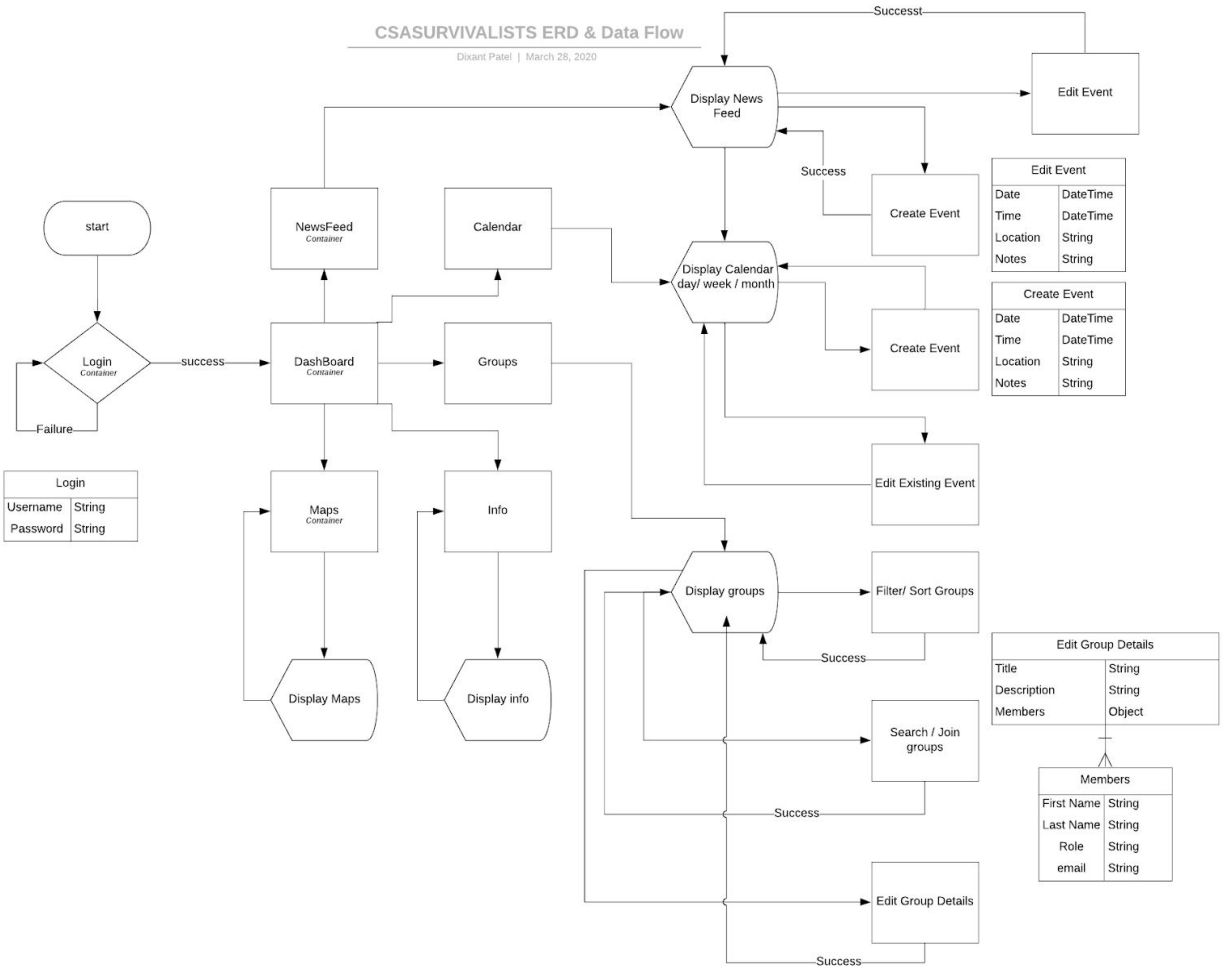


Figure 18: Data-Flow Diagram

During the design of the application, the team encountered some technical issues. To begin, one of the requirements was to integrate the application with Single Sign On (SSO), the University of Guelph's method of linking official student accounts with other applications like CourseLink. Initially, the team wanted to implement this, but a key issue was discovered. In order to actually use SSO, permission from the university was required. Due to this fact, there

was no way to actually implement it; however, a few solutions were brought up. At first, the University's SSO test page was used to simulate the login process; however, this solution was not successful because it could not redirect back to the main application once the fake login was completed. Therefore, the team eventually created our own dummy-login for testing purposes until we receive permission to use the official SSO. To continue, another problem was figuring out how to use Leaflet in React because it was an external API that nobody was familiar with, so some preliminary research was required. Also, the team was not certain how this external API would function in this environment. Unfortunately, this issue was not resolved because the map was ultimately never included in the final product due to the COVID-19 pandemic. Lastly, there was some difficulty implementing Material-UI. The idea was to use some images in the UI to enhance clarity and visual appeal, but importing the images turned out to be too troublesome, so BootStrap was used instead. This decision was a good compromise because the team was more familiar with this framework.

Time Estimation and Schedule

Throughout the course of the project, time was tracked on each task. Note that no concrete plan was established with regards to the completion order of tasks. The general rule was to follow the highest priority requirements and start with ones related to the three core features. In addition, building features shown in usability tests were given higher priority because more information was obtained about them. This meant that tasks were not explicitly scheduled, and our team was unable to tell if our general schedule was completely accurate. The team felt this form of development was productive, allowing quick iteration to build features at our own pace within our limited time. In order to track the allocated time, two burndown charts were generated to outline task completion. The burndown chart featured below tracked the tasks required for checkpoint 5 (*see Figure 19*), using the original time frame of deadlines (March 2 to March 27). This burndown chart was used to help visualize progress and to ensure that milestones and final deadlines were met. Some points detailed here include preparing the final presentation, building a working demo, and testing. Note that due to COVID-19, teams were unable to present any demo, so this is reflected in our burndown chart by having one task remaining to be completed.

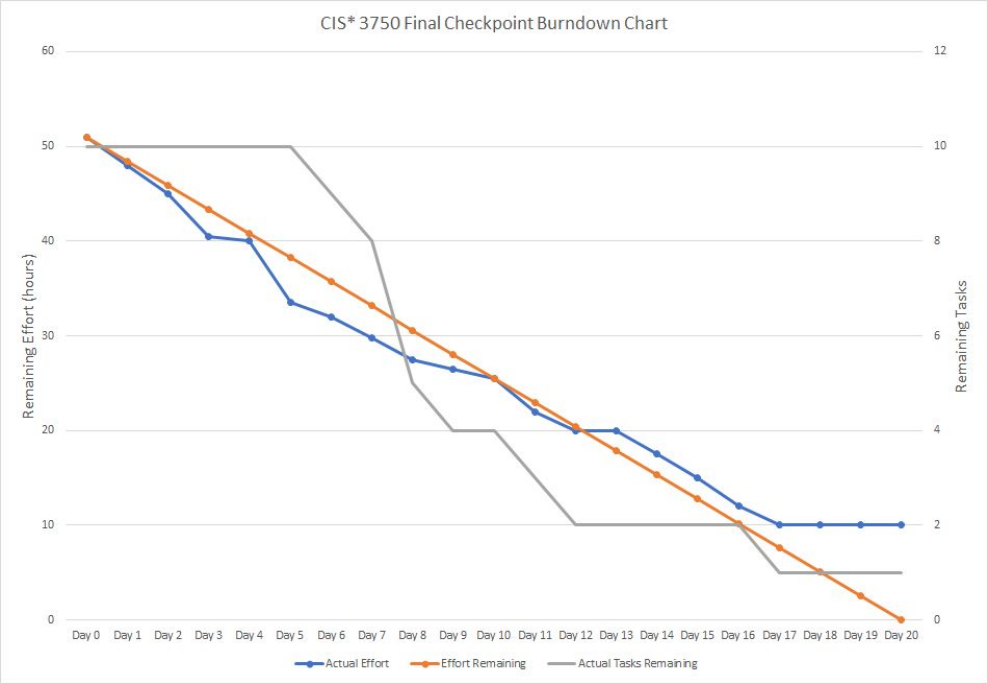


Figure 19: Checkpoint 5 Burndown Chart

In addition, a second burndown chart (*see Figure 20*) was created to track all of the application's features. Anything coded would be included in this chart, but not anything that was prototyped.

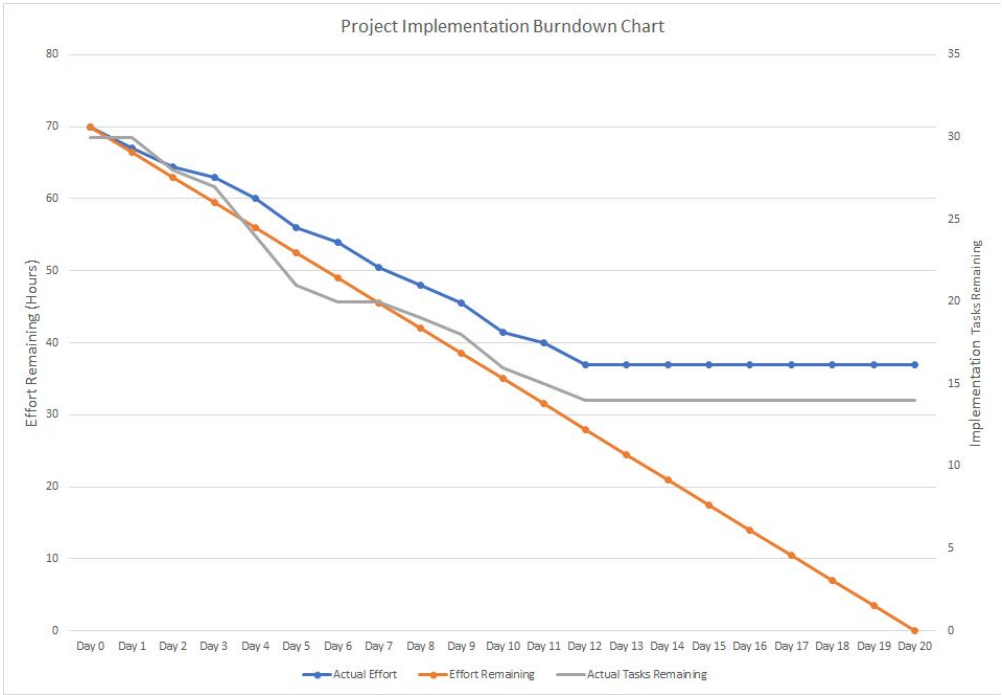


Figure 20: Feature Implementation Burndown chart

It is important to note that on day 12 of the burndown chart, work on all features ceased due to the University of Guelph restructuring courses around COVID-19, which caused focus and effort to shift elsewhere. Since the client modified submission requirements accordingly, the team was forced to accept that certain features would not be implemented. Therefore, efforts were refocused to create this final document and a mockup of a usable interface. As a result, the campus map unfortunately had no implementation. The other two core features were created as part of the high-fidelity Figma mockup, which is shown in the next section.

Implementation

At this point in time, development was interrupted because COVID-19 caused the university to shut down. Just as coding started, plans had to be revised, code that was completed at this point was saved, and the entire team shifted focus to the hi-fi mockup. In terms of functional code in the final product, a very basic login screen that took a user to the main home screen was completed. Moreover, this login is temporary until permission is received to integrate the application with the official SSO. Unfortunately, the home screen was not finished even though it served as the hub that allowed the user to access all of the application's features. It would have been similar to the home screen of the paper prototype, so a user would be able to access the campus map (implemented with Leaflet), the notification feed, calendar, groups, and a general information page. Once again, it is important to mention that work was forcefully halted under very short notice.

In terms of the prototype, both use cases from usability testing were fully implemented. The use case for the notification feed and the use case for the groups page. For this prototype, it was assumed that the logged in user had content moderator permissions for a few groups. The prototype was split into two sections based on these two use cases. A home page was not created in this hi-fi mockup because the plan was to demonstrate it along with the code. Unfortunately, this did not happen so the final prototype does not possess a home page.

First, the notification feed was designed. Everything looks relatively similar to the paper prototype and all the buttons are functional. The creation of a notification also looks similar to the prototype, just revamped to include UoG themes (*see Figure 21*).

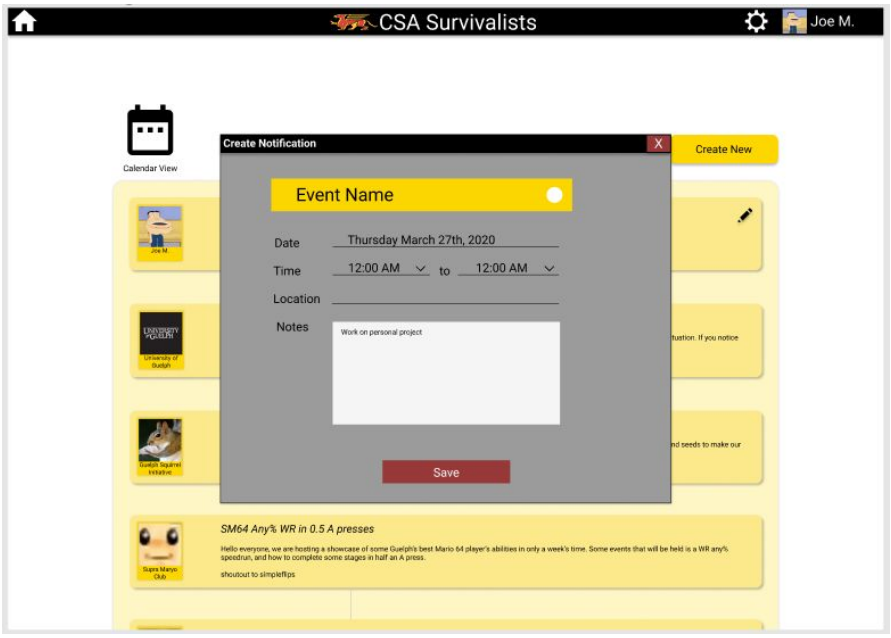


Figure 21: Creating a Notification in the Notification Feed

Next, the calendar was designed (*see Figure 22*). A weekly view of the calendar was also included (*see Appendix C3*). When a user clicks on a cell in the calendar, an event creator will pop-up just like in the notification feed (*see Appendix C3*). If a user selects a day, a pop-up will appear allowing them to either modify an existing event or create a new one.

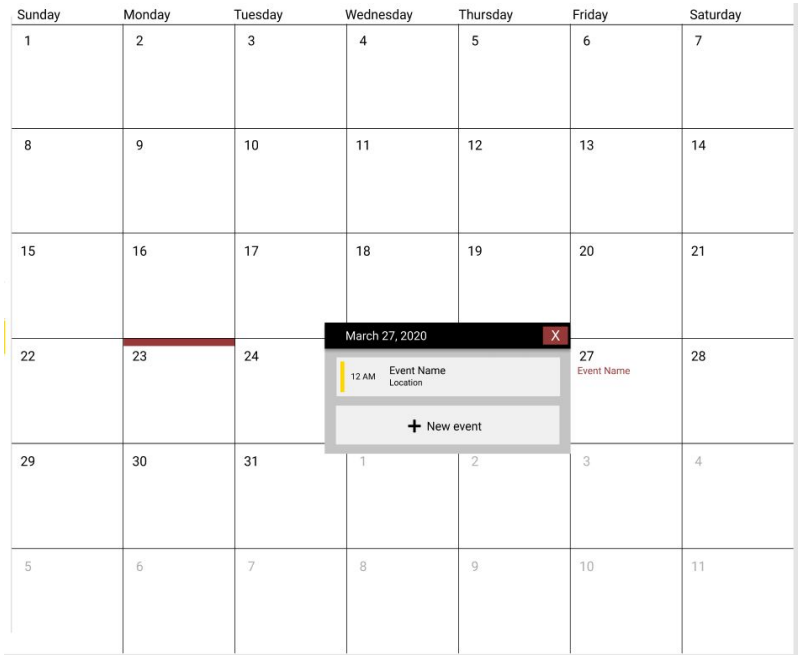
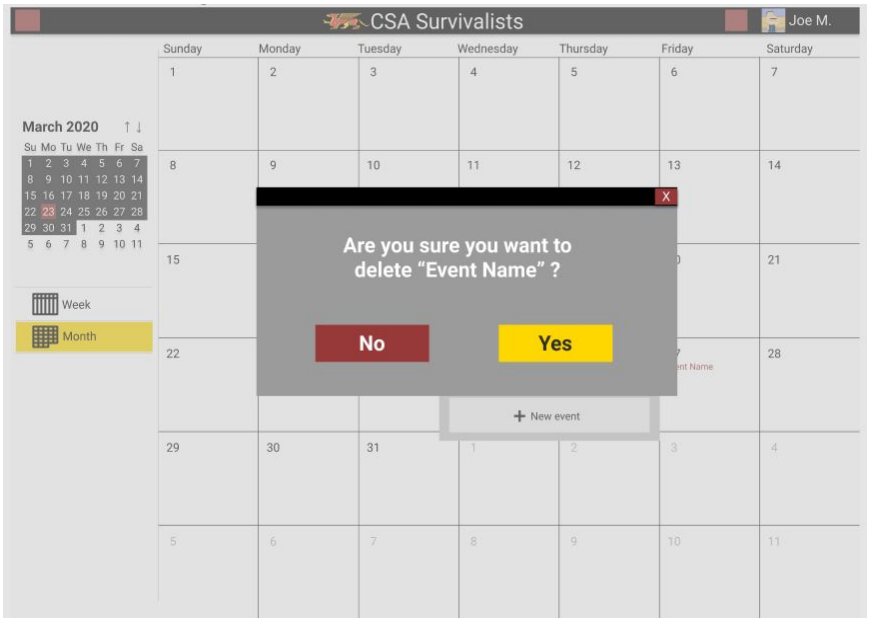
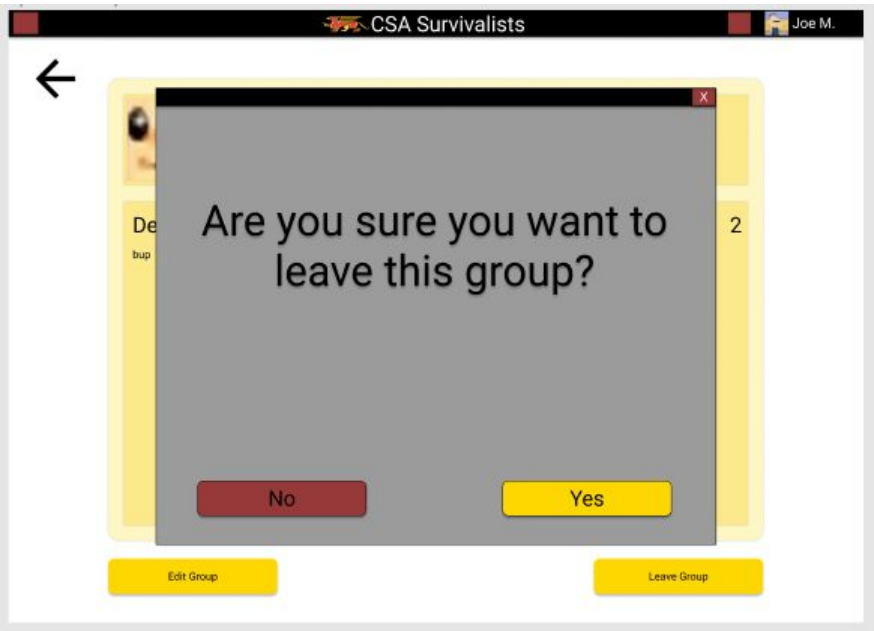


Figure 22: Selecting a Day in the Calendar and Choosing to Edit Existing or Create a New Event

If the user ever does something that may be destructive in nature, a confirmation box is presented (*see Figure 23*) to help prevent mistakes and reduce user error. This can be seen when a user modifies a notification, tries to delete a notification, or if they want to leave a group.





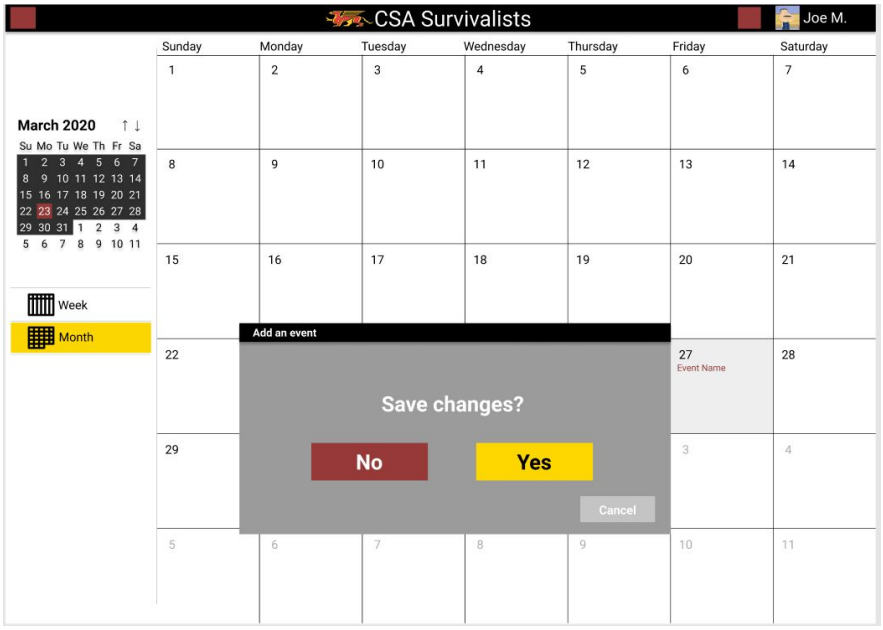


Figure 23: Three Examples of User Confirmation

The groups page of the final interface is also very similar to the paper prototype. An expand/collapse button (upwards arrow) was added to allow the user to organize the interface (*see Figure 24*). The results of clicking one of these buttons can be seen in *Appendix C3*.

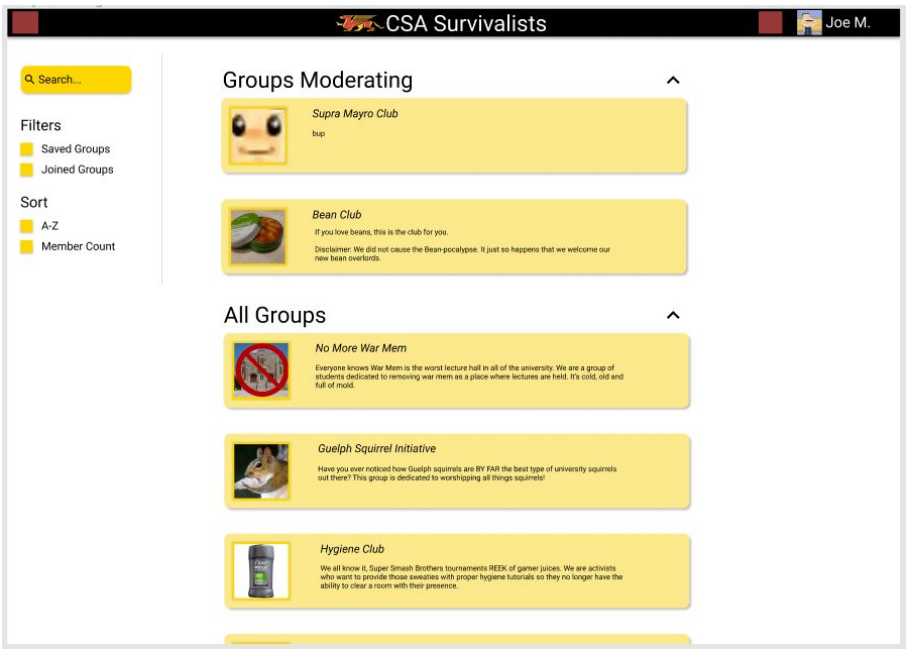


Figure 24: Groups Page

There are some features that Figma would not be able to implement, like detailed search menus. This became a problem because this group page contained a search box. Therefore, only very basic filters to simulate a sorted list of groups were mocked (*see Figure 25*).

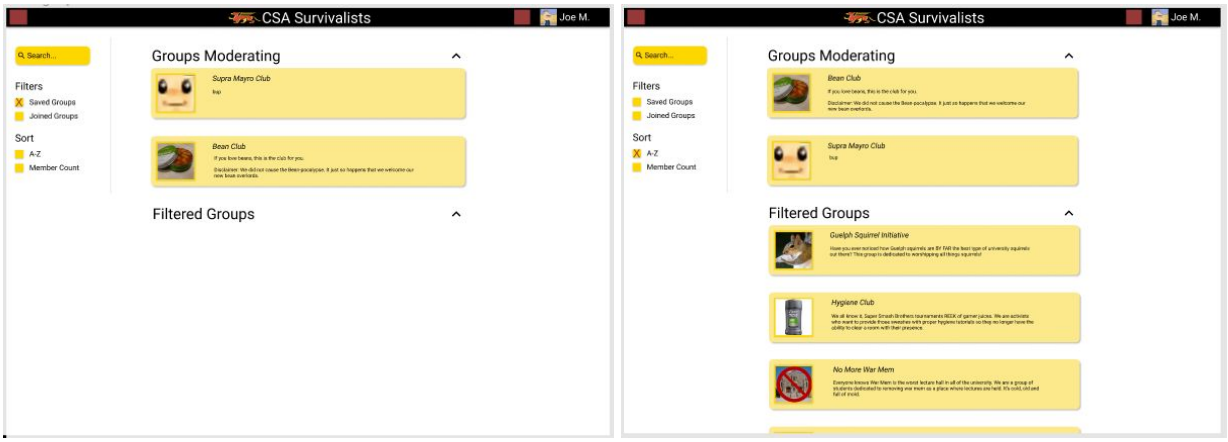


Figure 25: Examples of Different Simple Filters for Groups

Finally, when a user clicks on a specific group, they will be directed to the group's information page (*refer back to Figure 13*). If they are a moderator, they will have the option to edit the page (*see Appendix C3*).

Testing

Since there was not much code implemented, a formal testing strategy was not created for this application. However, the high-fidelity prototype was tested thoroughly at all stages of its development. As this was heavily based on the paper prototype, most of the usability testing was done during that phase. Thus, only simpler aspects such as colour schemes and links between pages, required testing to ensure that they functioned correctly. In order to verify design decisions, internal testing was conducted. For example, many different colour combinations were explored and team members' feedback was gathered. After the consideration of our feedback, it was presented to friends and family for an unbiased perspective.

Testing on the application's code involved basic debugging to ensure that it functioned correctly. In other words, making sure that any forms in the UI were properly submitted, that a user could enter text properly, and that buttons redirected as designed. If the code did not run after adding a feature, a bug fix was pushed to have the code work again with the new addition. Sometimes the application would run but do something unexpected, so debugging was necessary to determine the source of the issue. No major issues with the code were uncovered besides insignificant bugs and tiny mistakes that were easily fixed.

Once the high-fidelity prototype was finished, every member of the team tested it by going into Figma's "presentation mode". Here, each member pretended to be a user and recreated the usability tests that were conducted during the paper prototyping stage. If anyone was unable to complete the use case, or if a link was broken, it would be recorded as an issue that needed to be fixed. Since some feedback from the prototyping session was incorporated, those aspects were also tested to ensure that these new features were working as intended. Similar to the code tests, besides a few minor fixes and broken links, nothing of significance was uncovered in the high-fidelity prototype tests.

Conclusions

Ultimately, the entire team has gained valuable experience as a result of going through the Software Development Life Cycle (SDLC) for the CSA Survivalist application. It is unfortunate that the opportunity to implement more features was disrupted. Upon reflection, the team agreed that the development for code should have started earlier. There are a few takeaways that everyone received from this experience.

Everyone learned that developing an application from start to finish at the request of a client requires very clear communication between the client and the development team. By increasing the amount of communication, the process can become much smoother without any painful inconveniences from miscommunication. In addition, it can mitigate possible issues that can arise from simply not understanding what a client is asking for.

Moreover, the team learned that the process of creating an application does not start with building code, it begins with careful planning and brainstorming. It was unexpected that the majority of deliverables caused us to spend months of planning and testing before any

implementation was even discussed. By collecting thoughts and forming cohesive ideas that reflect the desires of the client and potential users, a better solution can be designed. However, it is vital that a balance must be found between planning and implementation. Devoting too many resources to brainstorming and planning can impede the creation of the application.

Out of everything that we learned, the most important lesson was that it's impossible to predict what may happen during development. Nobody predicted that a global pandemic would shut down the entire university and prevent groups from finishing their projects properly. It is important to recognize that things will not go according to plan. The best way to prepare is to accept that they will happen regardless and to be adaptable. Hopefully, by outlining these experiences from creating this application, those reading can gain an appreciation for the process of software development.

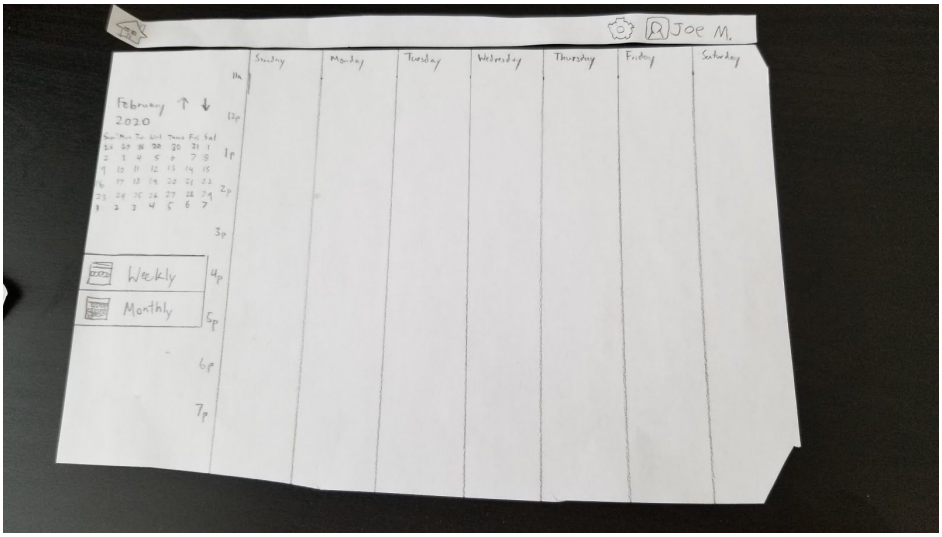
Section D: Appendices

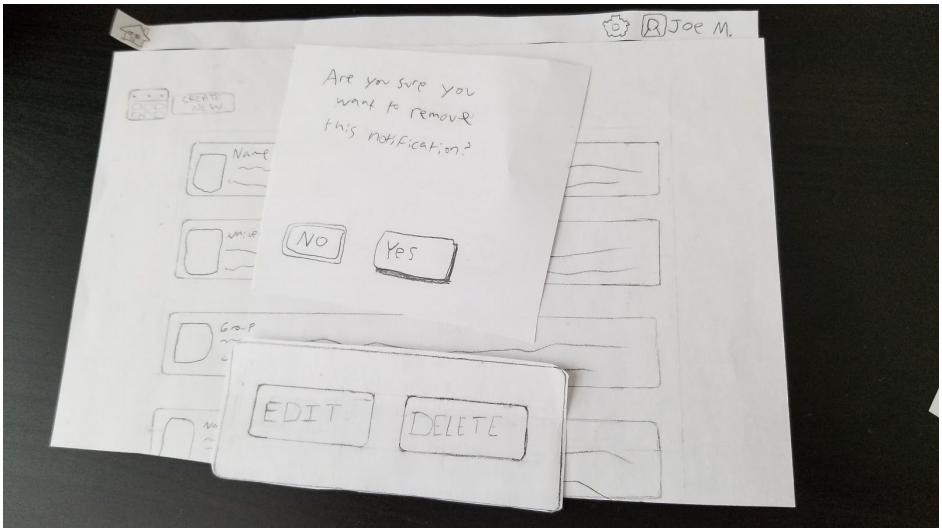
Appendix A

1. Link to wiki article containing personal biographies of each member of the group
    - Each member's personal biography can be found here.
2. User stories related to a campus map
    - **Bob Jones**
        - As an undergraduate student, I want to be able to find classrooms on the campus map so that I can get to classes
    - **Agatha Thornbrough**
        - As a student admitted into university, I want to know where the parking lots are, and which ones I can access so that I have no difficulties making it to campus on moving day
    - **Dirr T. Dan**
        - As an undergraduate student, I want to know about each building's responsibilities and services so that I know where to go and who to go to when I need information
    - **Peter Picky**
        - As a student looking to apply to university, I want to learn about the different housing options for first-year students so that I know if I will feel welcome living on-campus
3. User stories related to campus life or clubs available
    - **Lianna Booth**
        - As an undergraduate student, I want to know about which clubs are offered at the university so that I can be involved in extracurricular activities to unwind after classes
    - **Casper Shad**
        - As a student pursuing higher education, I want a guide on the many different clubs that the school offers, especially those geared toward mature students so that I can socialize with like-minded graduate students
    - **Peter Picky**

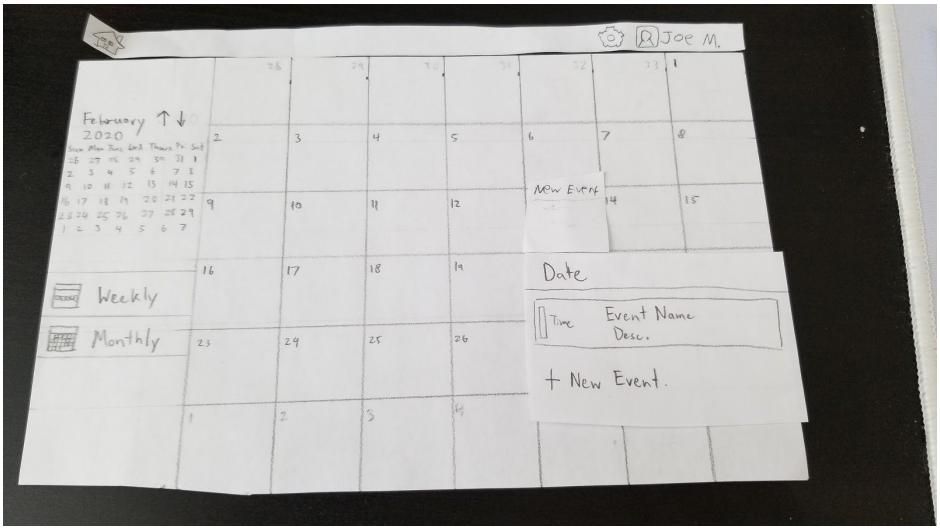
        - As a student looking to apply to university, I want to learn about the student life of the school I might apply to so that I know if I'm a good fit for this community
4. A detailed overview of Use Case 1
    - The primary actor for this use case is a student who is logged in to the system. In this use case, the student will create a new notification to remind them about an event that is occurring at a date and time. This use case has the precondition that the user must be logged in so they have access to their newsfeed and calendar. To perform the use case, the user will go to the home screen then navigate to their personalized newsfeed. From there, the user will click on a create notification button and fill out all the required fields. They will then hit a save button which will store their new notification in their newsfeed. Alternate methods can be taken such as creating the notification from the calendar.
    - Link to the full use case can be found here.
5. A detailed overview of Use Case 2
    - This use case is about a user who is trying to manage information about a group that they moderate. The primary actor in this use case is a content moderator who has permissions to manage a group's information. Before this use case can occur, they must be logged in to the system and authenticated as a content moderator for at least one group. We are also assuming that the group already exists in the system and doesn't need to be created beforehand. To complete this use case the moderator must navigate to their group that they manage via the groups page and click an edit button. They will then modify aspects of the group then press a save button to commit those changes. An alternate way for the moderator to go to the group page is by searching for it in a search bar.
    - Link to the full use case can be found here.
6. Adalrich Schröder user story
    - As an international student, I want to be able to have text translated to a language I'm comfortable with so that I can have a better understanding of what I'm reading
7. Requirements Spreadsheet
    - To view the requirements document, a download link can be viewed here.

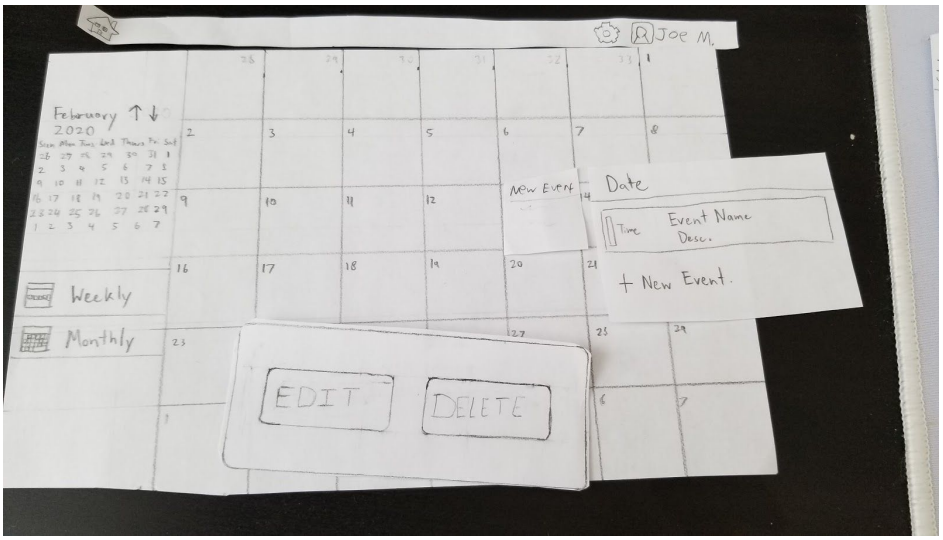
Appendix B

1. Additional images of the paper prototype
   - 

     - Weekly View of the Calendar
   - 

     - Confirmation for Deleting a Notification

- 

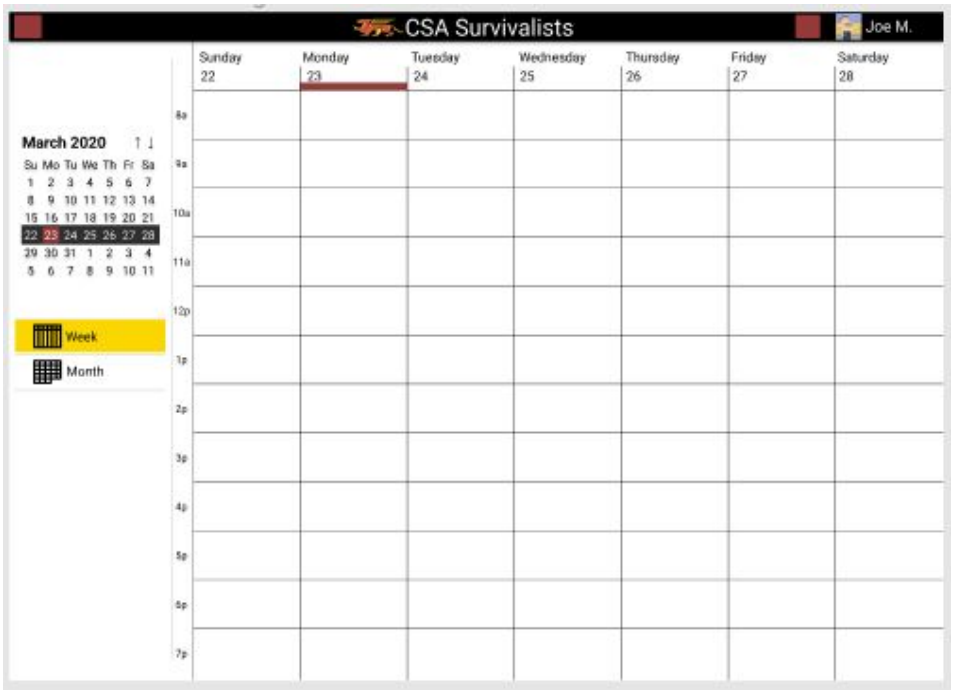
    - Clicking on a Notification That Already Exists
- 

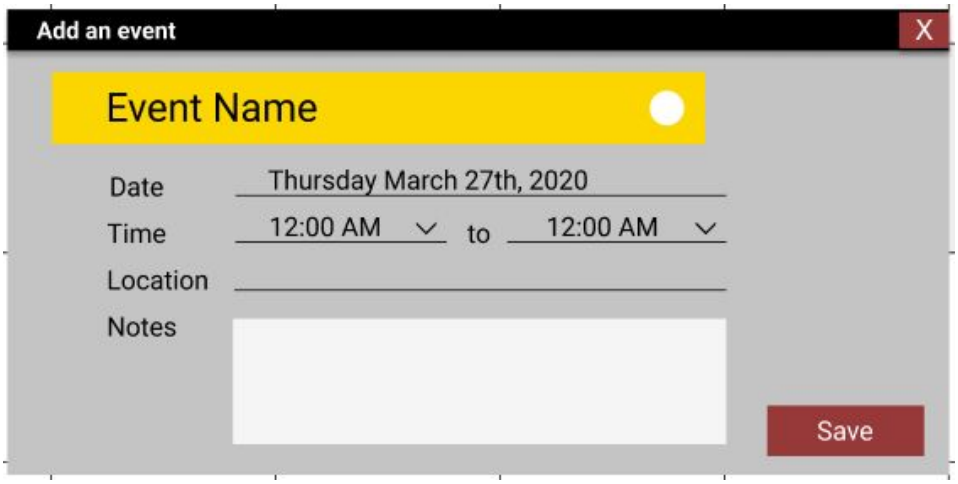
    - User Choosing What to do After Clicking on a Specific Event

Appendix C

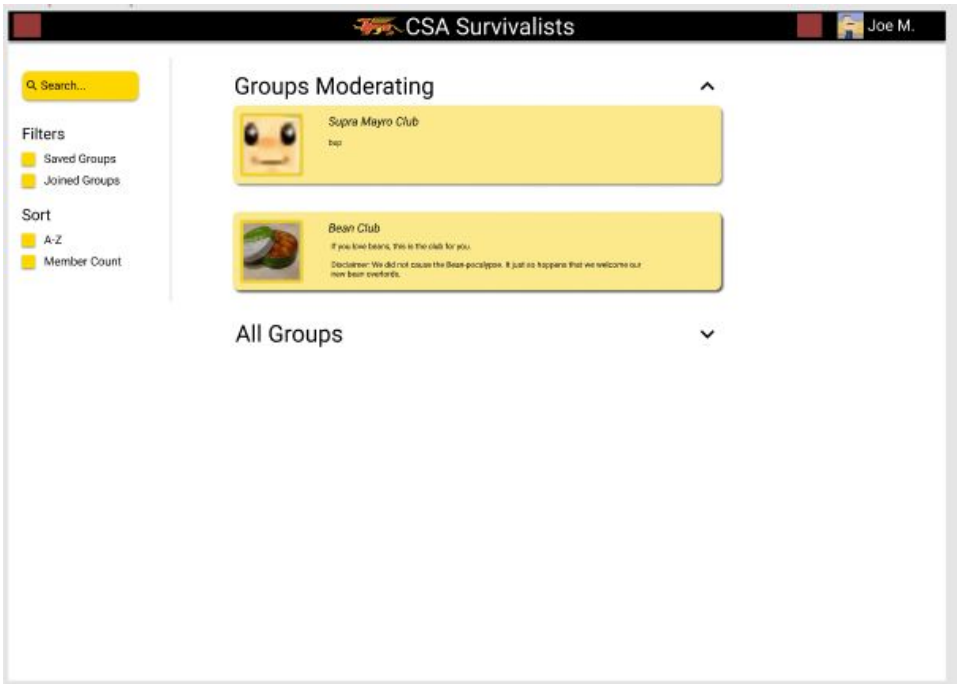
1. Leaflet: an external map API
    - Leaflet is an open-source Javascript library for creating mobile-friendly maps.
    - A link to Leaflet can be found here.
    - A link to the Leaflet documentation can be found here.
2. Styled Icons: a way to import intuitive icons into a Javascript application
    - Styled Icons is a large collection of icons that are easily imported into Javascript code and displayed in React apps.

- A link to the website can be found here.

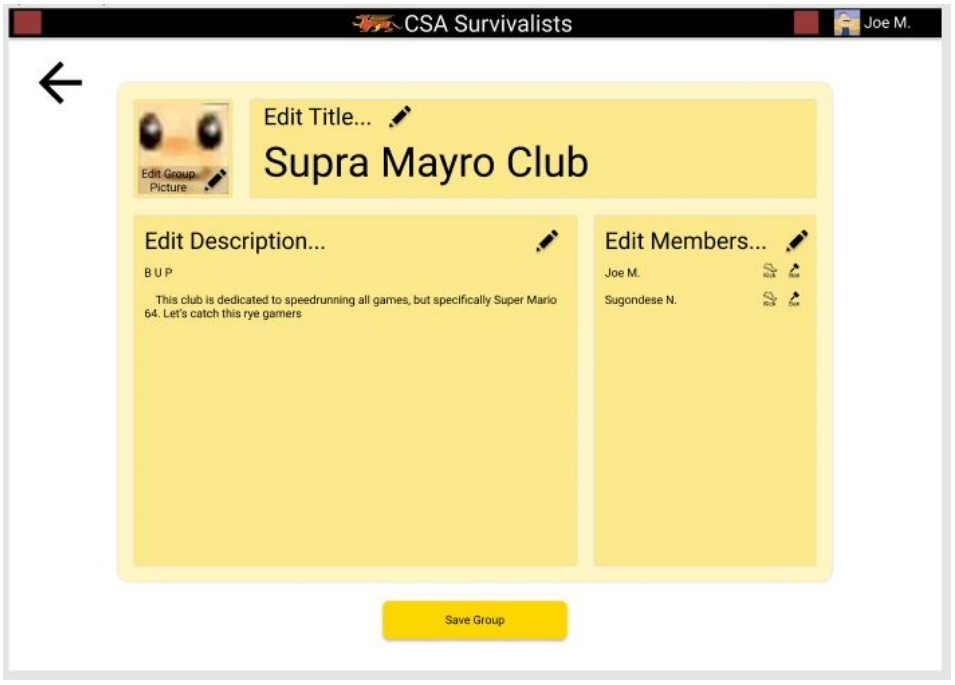
3. High-fidelity prototype additional photos
   - 

     - Calendar Weekly View
   - 

     - Calendar Create Notification



- Collapsing a Category on the Groups Page



- Moderator View of Editing a Group Info Page