Care Compare Capstone Experience Report

The purpose of this report is to highlight the successes and failures encountered during the engineering process as well as an overview of the technologies and tools we used to organize and create our capstone project. Our original goal was to create a web application that allows people to compare insurance quotes from multiple insurance companies side by side using data from the insurers. Due to the lack of access to data from insurers, our final product took a community-focused approach by allowing users to compare to other users' quotes as well as share their personal experiences within a community of practice. We were able to accomplish that community-focused approach and the tools and processes discussed below will highlight both the good and bad experiences of our capstone project.

Made by: Kegan Lavoy, Daris Lychuk
Capstone of April, 2020

Table of Contents

Early Stages and Planning

We used the entire first semester of our Capstone project to organize, plan, and design the processes and frameworks that we wanted to use to create our Capstone project. We used a waterfall/agile hybrid engineering methodology where we progressed in a linear pattern, but iterated each of our steps to improve our design. This approach worked well for us because we were able to focus on one task at a time, but still get feedback and suggestions from advisors and other students so that we could iterate and improve on our designs.

For project roadmap strategies and requirements gathering we used user story maps, a kanban board, and user personas as ways to plan out our Capstone project and stay on track throughout the process. User story mapping was a good starting point for us in terms of gathering requirements from users and planning our tasks accordingly, but we found that this method fell short in terms of keeping us organized throughout the project. Since user story mapping wasn't sufficient enough for us on its own, we created user personas as a way to take different perspectives on our problem and determine what different groups of people might be looking for from our application. This worked well for us because we were able to ask friends and family about their personal experiences with finding insurance which allowed us to build a better application for all types of people.

This is Jon. Jon may look a little old, but he actually just retired at the age of 65, and is looking for an insurance plan to keep him and his wife covered. Jon lives at home in Regina, Saskatchewan, where he has lived his whole life. He currently lives with just his wife Karen, as his kids have grown old, and moved out. While Jon was in the work field, his company had a very good health, dental, vision, and prescription drug coverage plan. He noticed as he was getting older that he was relying on the plan more and more. Jon is not the most tech savvy guy and would prefer if there was an easier method to finding the best plan suited for him without having to leave the comfort of his home.

Figure 1. Persona for older man

This is Aaron. Aaron moved to Regina, Saskatchewan from Australia in order to study abroad for his degree and one day find a job he loves. He is currently 22 years old and just finished his last year of Computer Science at the University of Regina. Aaron just got hired at FarmTech, and even though he loves the job, they unfortunately do not offer much when it comes to insurance coverage. He used to rely on the University's student plan, but now has to find his own. Aaron also just moved into a small apartment near Uplands, and still takes the bus whenever he needs to go anywhere. Due to Aaron being a more tech savvy guy, and not owning a car currently, he would prefer if there was some kind of application that he could use to look up the best insurance plan to add additional coverage on top of what his work is already offering.

Figure 2. Persona of student

We created a Kanban board to help us organize and separate work into manageable chunks as well as provide ourselves with a visual of our progression throughout the semester. We personally loved the Kanban board method because it is extremely malleable as tasks can be removed or added based on changes within the project design which worked well with our agile approach and it provides a good visual representation of your progress and remaining tasks. Kanban boards are also very easy to set up and can be implemented on paper, whiteboards, online, etc.

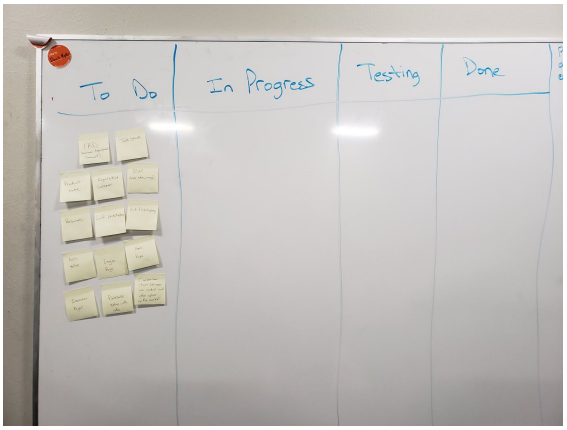

Figure 3. Kanban board start

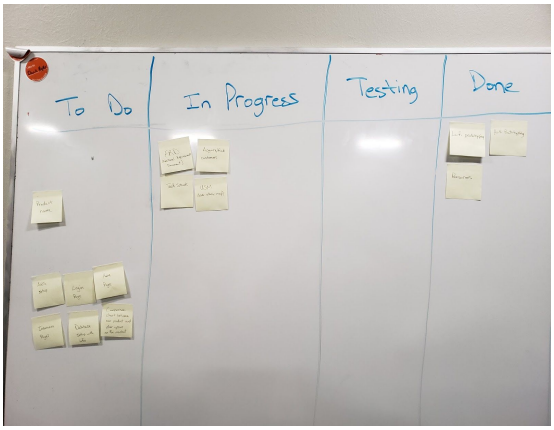

Figure 4. Kanban board iteration 1

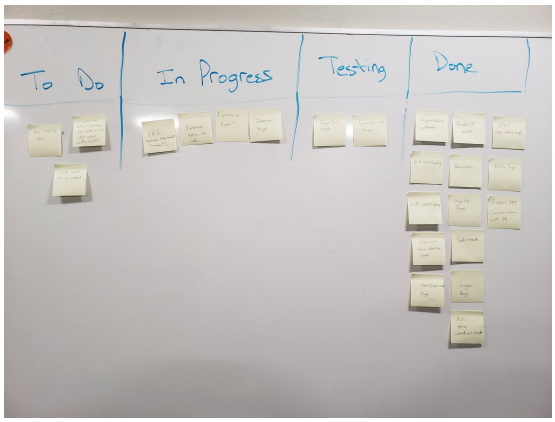

Figure 5. Kanban board iteration 2

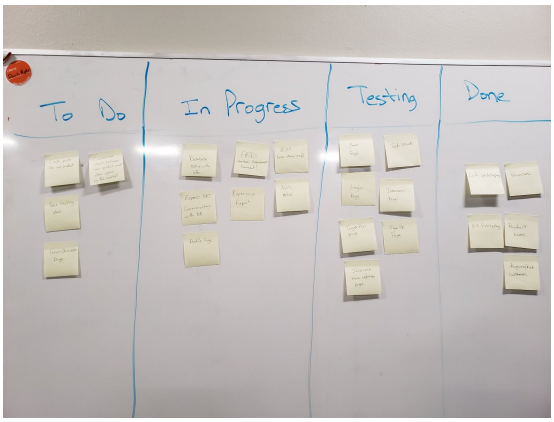

Figure 6. Kanban board iteration 3

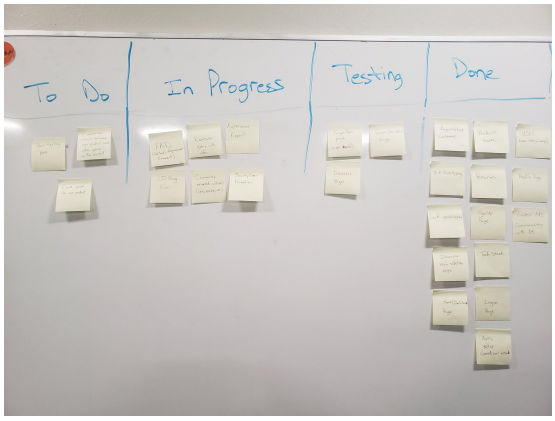

Figure 7. Kanban board iteration 4

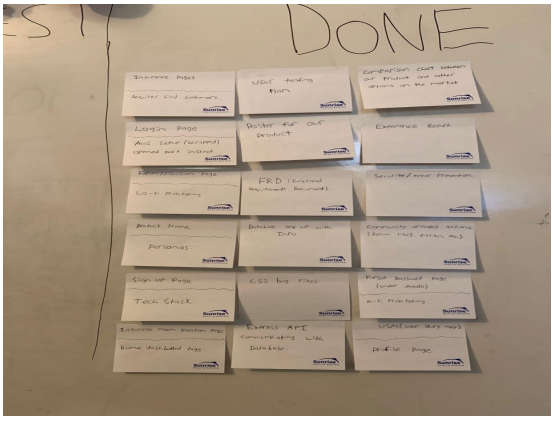

Figure 8. Kanban board final

When designing our interface for our web application, we made use of lo-fi and hi-fi mockups using a whiteboard and adobe XD respectively. We highly recommend creating prototypes when designing any interface because they allow change to your designs to be fast and require much less effort. In our experience we changed designs more than 20 times due to tech restrictions, requirements being added or removed, or new innovations coming into our minds in the middle of our designing process. Our initial designs are nowhere near what our final product looks like or operates and we think that was a valuable lesson learned in this whole process. As you can see from the development pictures below, our initial design is completely different from our final design and the initial design doesn't even include some of the elements that came into focus further into the Capstone process.

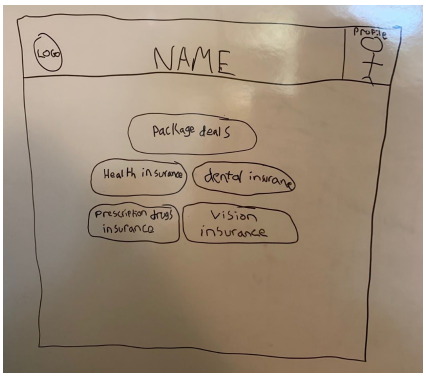

Figure 9. Initial Lo-fi dashboard

Figure 10. Hi-fi dashboard

Figure 11. Final dashboard

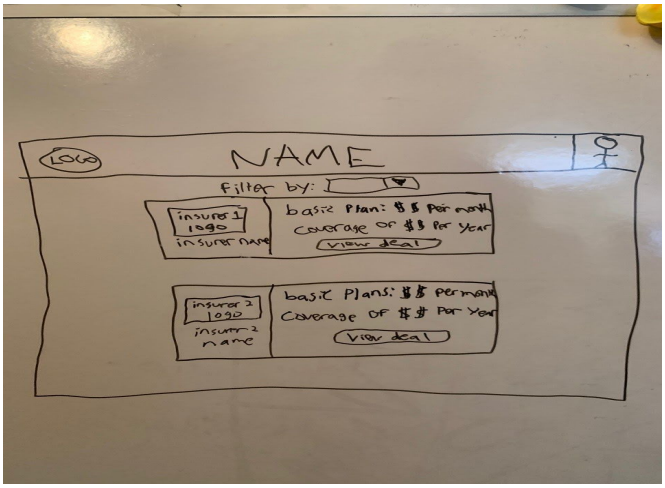

Figure 12. Initial Lo-fi Insurance Quote

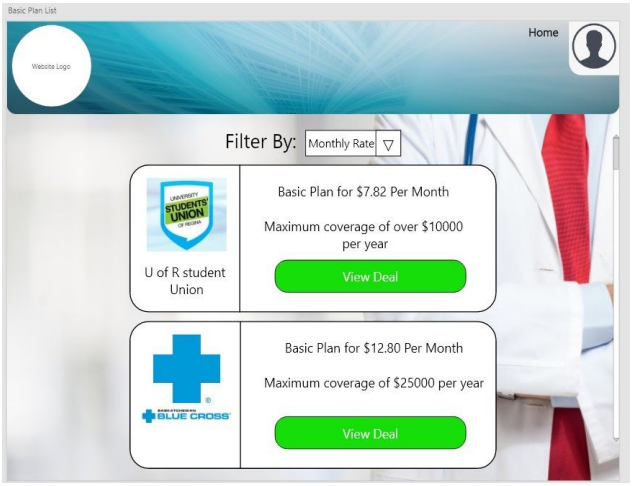

Figure 13. Hi-fi Insurance Quote

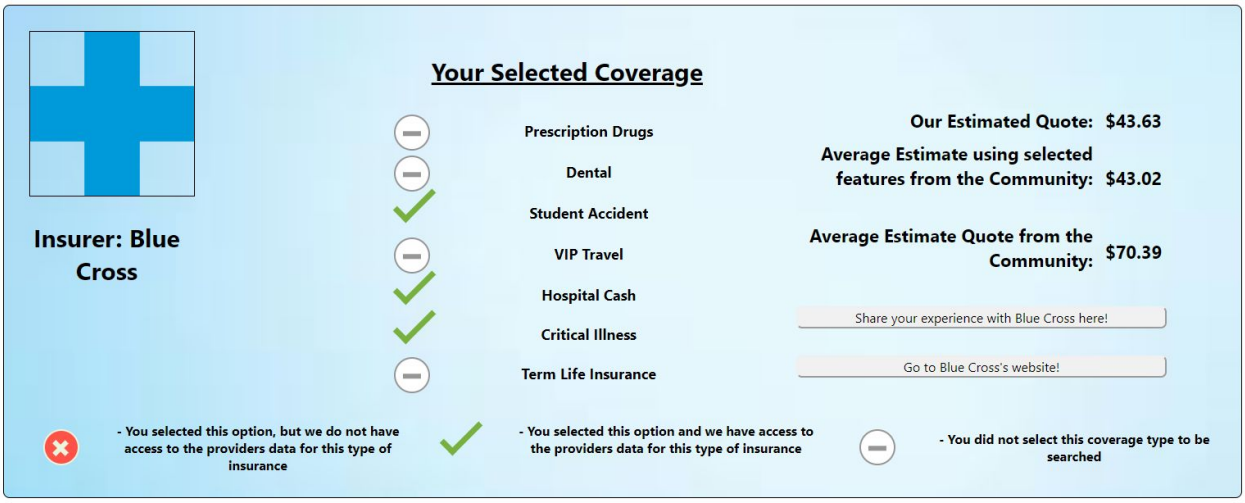

Figure 14. Final Product Insurance Quote

The final decision made during our planning stage was our tech stack. We decided to use a modified version of the MEVN tech stack where we used MySQL, Express, Vue.js, and Node.js. We decided to use Vue.js and Node.js frameworks for our front and back end because we were interested in exploring a new framework and language we had never experienced before. Vue.js and React are also gaining a lot of traction in the industry which gave us extra motivation to learn something that may become industry standard in the future. We replaced the commonly used MongoDB with MySQL because we have related data within our database between tables and we also have experience with MySQL through school which made it a better choice for our Capstone.

We really enjoyed working with this tech stack because it includes many “quality of life” features (examples include: MVC design pattern used directly within Vue.js, “hot-reloads” used in development that apply changes to code within the application without having to reload website or recompile code, uses single page application framework which eliminates page loading, etc.) within it and has a very natural modular structure at its core which helps with providing clean, reusable, modular code. We created communication and data diagrams for our application to model how our site navigated and interacted with both itself and other applications.Our communication diagram was simple and didn’t change much throughout the process, but our data diagram was modified and adjusted based on the changes made to the system. However, due to Vue.js’ modular structure making use of components and routes, new components could be added to the diagram very easily without affecting any other components in the application.

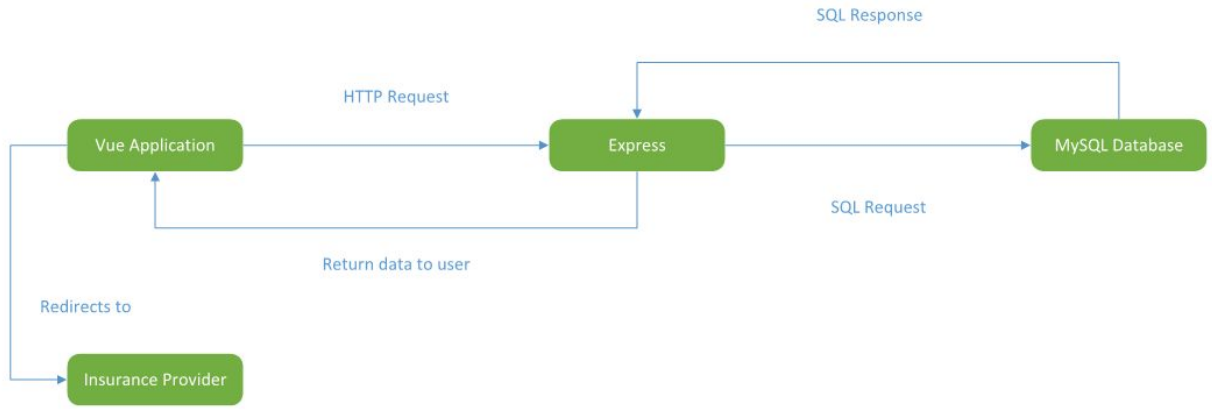

Figure 15. Communication diagram for our web application

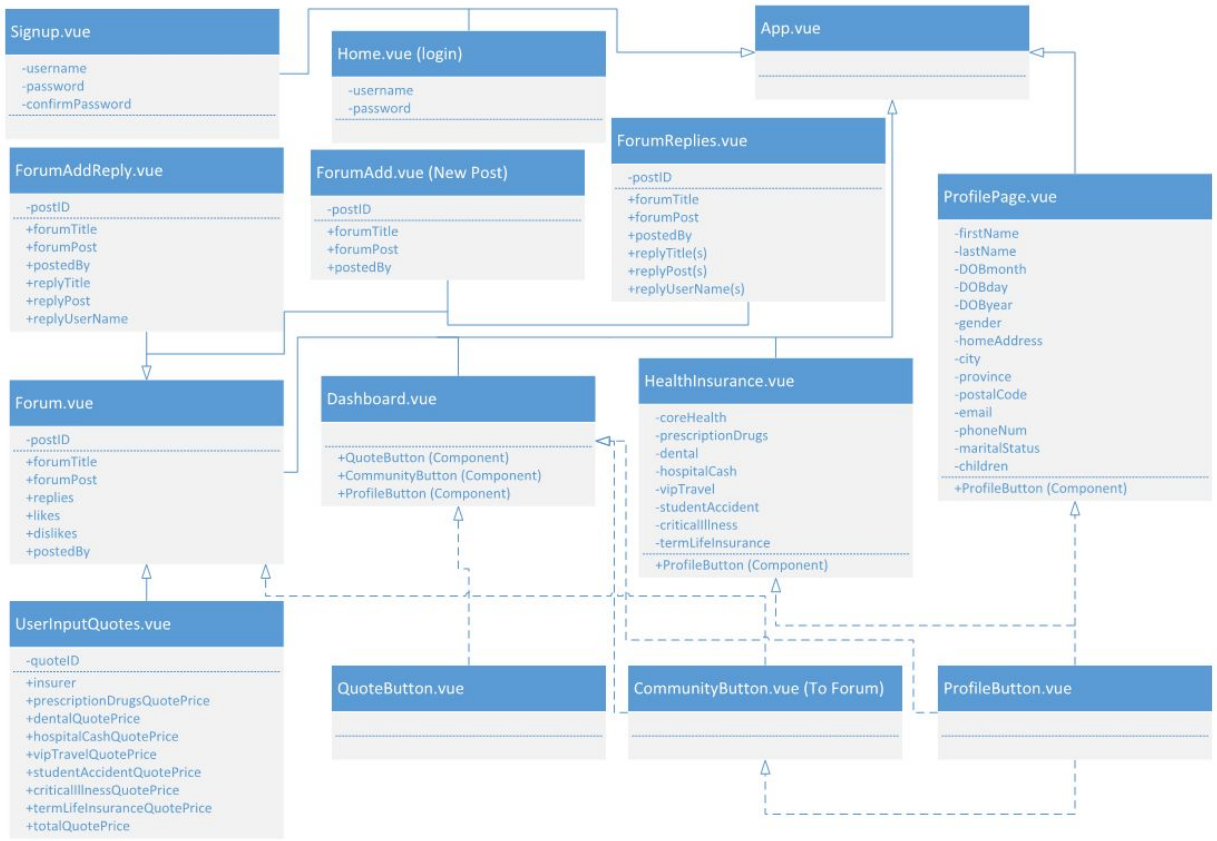

Figure 16. Data diagram showing how different components (ex. buttons) can be added without affecting other components (modular code)

Developing Stage

The developing stage was where many of our hurdles experienced in our Capstone project occured. One of our first major hurdles was insurance data acquisition. By nature, most insurance companies keep their specific insurance pricing confidential so that their competitors have a harder time competing for customers. We discovered that at best insurance companies provided an estimate quote builder and at worst only provided a meeting with an advisor which made it very difficult for us to get reliable data. Essentially, we failed at obtaining insurance data directly from insurers. This problem led us to create a new community-based approach by attempting to crowd-source the data from our users themselves. This led to us redesigning our application to include an open forum where users could communicate about their own insurance experiences with different providers as well as share their real quote data with us which in turn allowed us to provide a better quote estimate for other users. In our opinion, this change from insurer focused data to user focused data was an excellent addition to our application and dramatically enhanced our application because users now have a sense of belonging to a community and can contribute to community discussions.

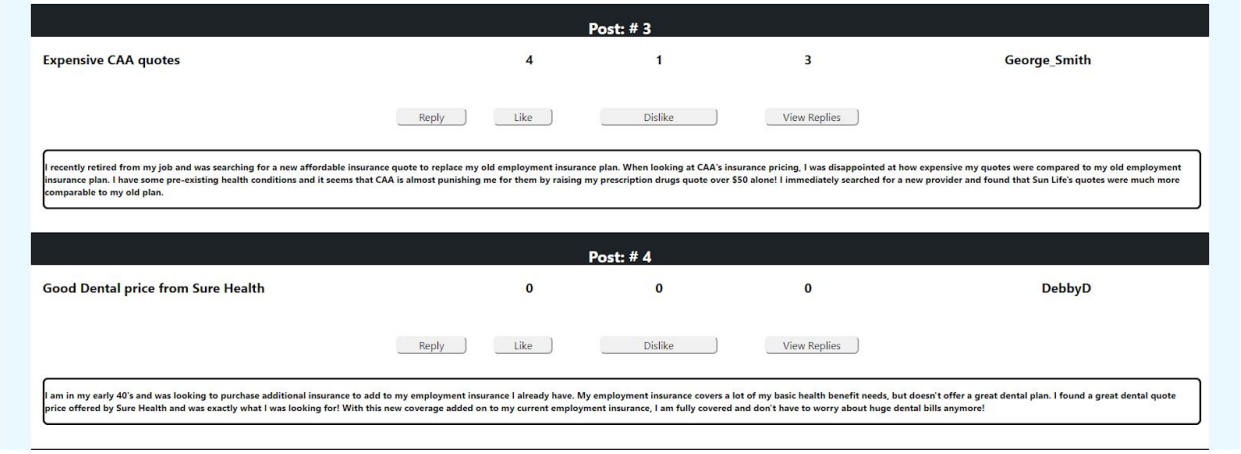

Figure 17. Example of new user forum implemented for community based approach

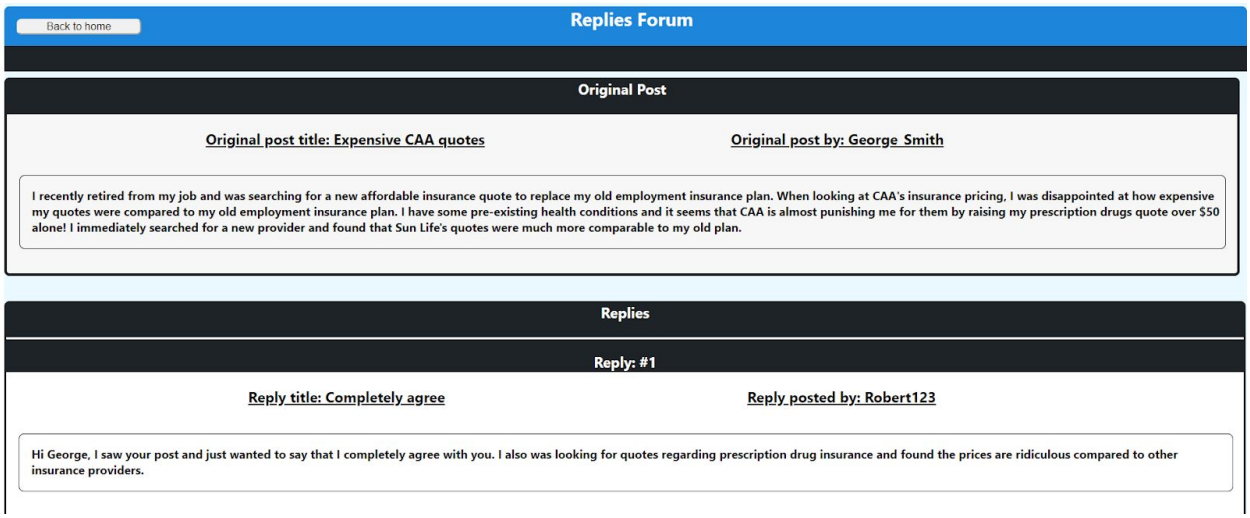

Figure 18. Example of Reply functionality for user forum

When it came to hosting our web application, we initially envisioned using Amazon Web Services (AWS) to host our website. We wanted to use an AWS EC2 Ubuntu instance to host our server, but neither of us had used AWS previously and when we dived into the software it proved to be more complex than originally expected. We executed steps recommended by other students who had used AWS previously such as purchasing a domain name from Registrar, setting a static IP for our instance, running our Ubuntu instance, setting up security groups, and more yet we were still unable to host our application successfully on AWS. At this point nearly three weeks had been put into trying to get AWS to work successfully which was severely holding back the progress of the rest of the project, so we abandoned the AWS route and instead opened up ports on Kegan's home computer to host both the web application and Express RESTful API that we built. On top of everything else, since we were new to AWS and did not know about AWS credits, we were also paying to use all of these services. Our advice to future students is if you are involving new technologies that you are unfamiliar with, make sure to start exploring them very early to see if the tool is right for you. Unexpected problems are always bound to occur and having as much time as possible to work through difficulties is always helpful.

After making the decision to step away from AWS, our progress dramatically increased within our project. We were able to set up our MySQL database using XAMPP software that provides a MariaDB instance as well as an Apache server. Soon after we also had our Express RESTful API operating fully.

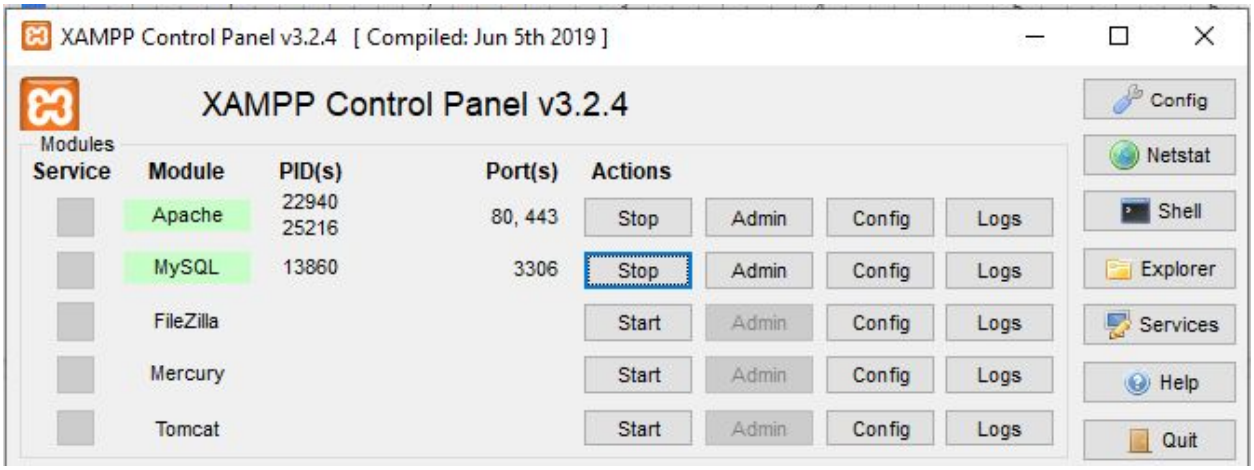

Figure 19. XAMPP software supplying Apache and MySQL services

```
JS index.js > ...
const express = require('express');
const mysql = require('mysql');
const cors = require('cors');
const app = express();



//creates connection to local database
const db = mysql.createConnection({

    host     : 'localhost',
    user     : 'root',
    password : '',
    database : 'CareCompare'


})

db.connect((err) => {

    if(err){
        throw err;
    }

    console.log('MySQL Connected...');

})

app.use(cors());
app.use(express.json());

app.listen(3000, () => console.log(`Example app listening on port ${3000}!`));
```

Figure 20. Base Express API Code

Coding/Testing Phase

Once all of our frameworks were in place, we dived right into the code and began creating our application. Vue.js combines HTML, CSS, and Javascript together in Vue files to create individual pages and components. The unique part about Vue.js is that it uses elements called v-directives that have special functions within Vue. These directives allow the creation of very useful and cool dynamic elements within applications and was something we really enjoyed when working with Vue. When developing our application, we followed some coding conventions that we found important such as keeping a consistent indentation scheme, commenting all functions and complex logics, and using descriptive variable names.

```
<!--This is a div that makes use of the v-for directive which repeats the div an amount of times equal to the number of posts retrieved from the database.
    Each post has a unique id and index assigned to it so that they are distiguishable from one another. Each one of these divs provide an array o f options to
    the users such as replying to the post, liking the post, disliking the post, viewing all replies made by other users, and seeing who the post was made by.-->
<div class="center-community" id="center-community-tabs">
  <div v-bind:key="post.post_id" v-bind:index="index" v-for="(post, index) in posts">
      <div id="postHead"><h2 class="postNumbering">Post: # {{index + 1}}</h2></div>
      <div id="topicHeading"><h3 class="forum-H1">{{post.post_title}}</h3></div>
      <div id="replyButton"><button type="button" class="button" @click="$router.push(`/ForumAddReply/${user_id}/${post.post_id}/${post.post_replies}`)">Reply</button></div>
      <div id="numLikesHeading"><h3 class="forum-H3">{{post.post_likes}}</h3></div>
      <div id="numDislikesHeading"><h3 class="forum-H3">{{post.post_dislikes}}</h3></div>
      <div id="numRepliesHeading"><h3 class="forum-H3">{{post.post_replies}}</h3></div>
      <div id="lastPostNameHeading"><h3 class="forum-H3">{{post.username}}</h3></div>
      <button type="button" id="numLikesButton" class="button" @click="checkLike(post.post_id, user_id, post.post_likes)">Like</button>
      <button type="button" id="numDislikesButton" class="button" @click="checkDislike(post.post_id, user_id, post.post_dislikes)">Dislike</button>
      <button type="button" id="viewRepliesButton" class="button" @click="$router.push(`/ForumReplies/${user_id}/${post.post_id}`)">View Replies</button>
      <div id="forumPost"><h3 class="forum-body">{{post.post_body}}</h3></div>

  </div>
</div>
```

Figure 21. Example of v-bind and v-for directives/indentation scheme (Forum.vue, line 32-47)

```
/*This is the start of the calculations for the user quotes. It once again checks the user inputs.
  If the user selected the type of insurance to be searched, each table in the database that was retrieved
  is looped through. If the price in a row of the table is 0, that means that the user didn't have that
  insurance included in their user submitted quote and isn't processed. If the price is anything other than
  0, it is added to a temporary average and a counter is updated by 1 so that an average can be calculated later.
  All of the individual insurance type's averages are then added together at the end to give the final price.*/
```

Figure 22. Example of a good comment describing a complex part of a function (HealthInsurance.vue, line 622-627)

```
input: {
    firstName: "",
    lastName: "",
    DOBmonth: "",
    DOBday: "",
    DOByear: "",
    gender: "",
    homeAddress: "",
    city: "",
    province: "",
    postalCode: "",
    email: "",
    phoneNum: "",
    maritalStatus: "",
    children: ""

},

user_id: 0,
invalidMonth: false,
validMonth: true,
invalidDay: false,
validDay: true,
invalidYear: false,
validYear: true,
invalidSave: true,
validSave: false,
```

Figure 23. Example of descriptive variable names for profile info and error checking (ProfilePage.vue line 71-97)

As we created new sections and pages of our web application, we applied unit testing on the functions by using the Vue Developer Tools Chrome extension that allowed us to view in real time the type, value, structure of both the individual pages and variables. We tested our API functions by both debugging the Express app itself as well as viewing the responses of the API requests after being executed with the Vue Developer Tools and we were able to work out our bugs and receive correct and dynamic results. We also tested our CSS code to make sure that it was dynamic on multiple screen sizes and browsers by using an online tool found here: http://whatismyscreenresolution.net/multi-screen-test. We tested on all standard screen sizes between 10 inches and 27 inches, and on the Mozilla Firefox and Google Chrome browsers.

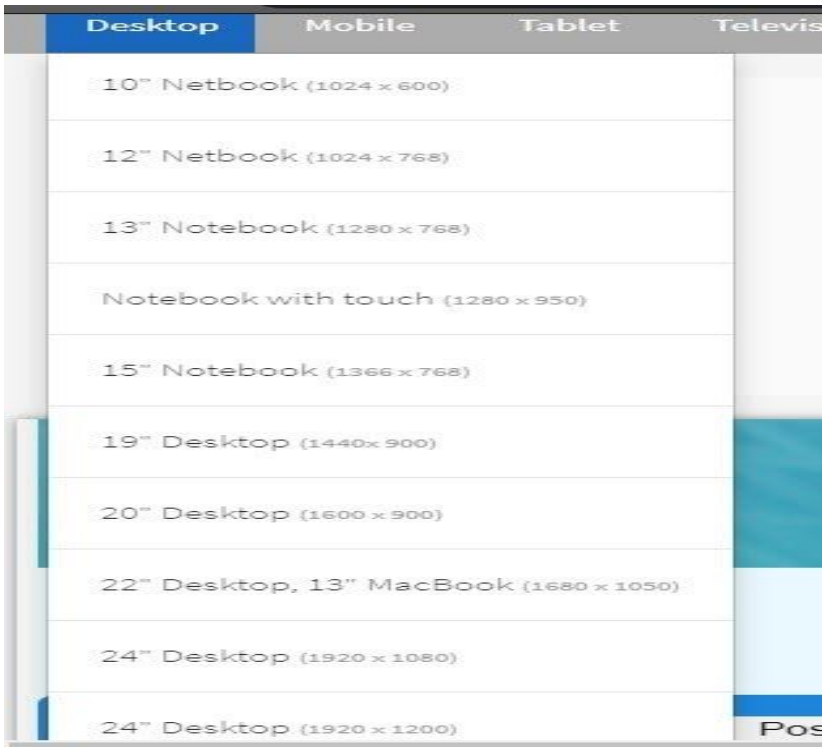

Figure 24. Different Screen sizes tested using whatismyscreenresolution.net

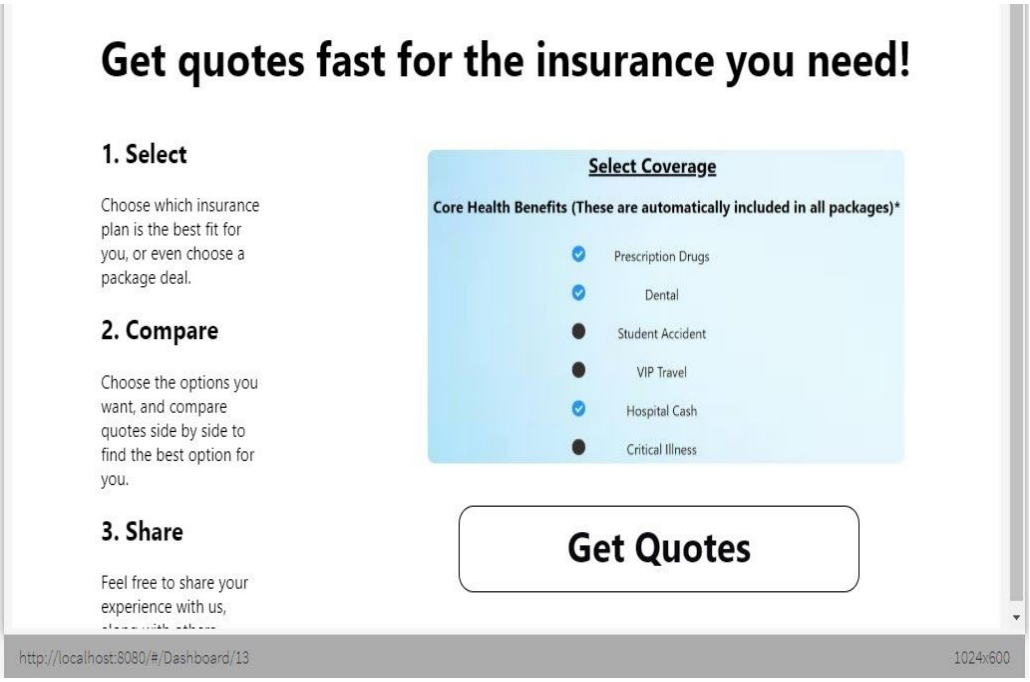

Figure 25. Dynamically modified Dashboard for 10 inch screen

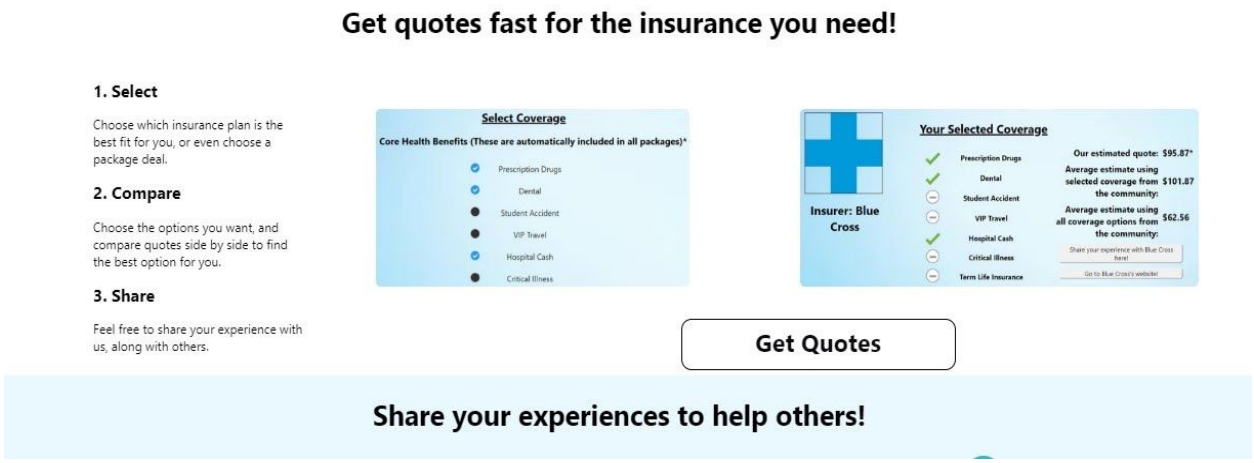

Figure 26. Dynamically modified Dashboard for 24 inch screen

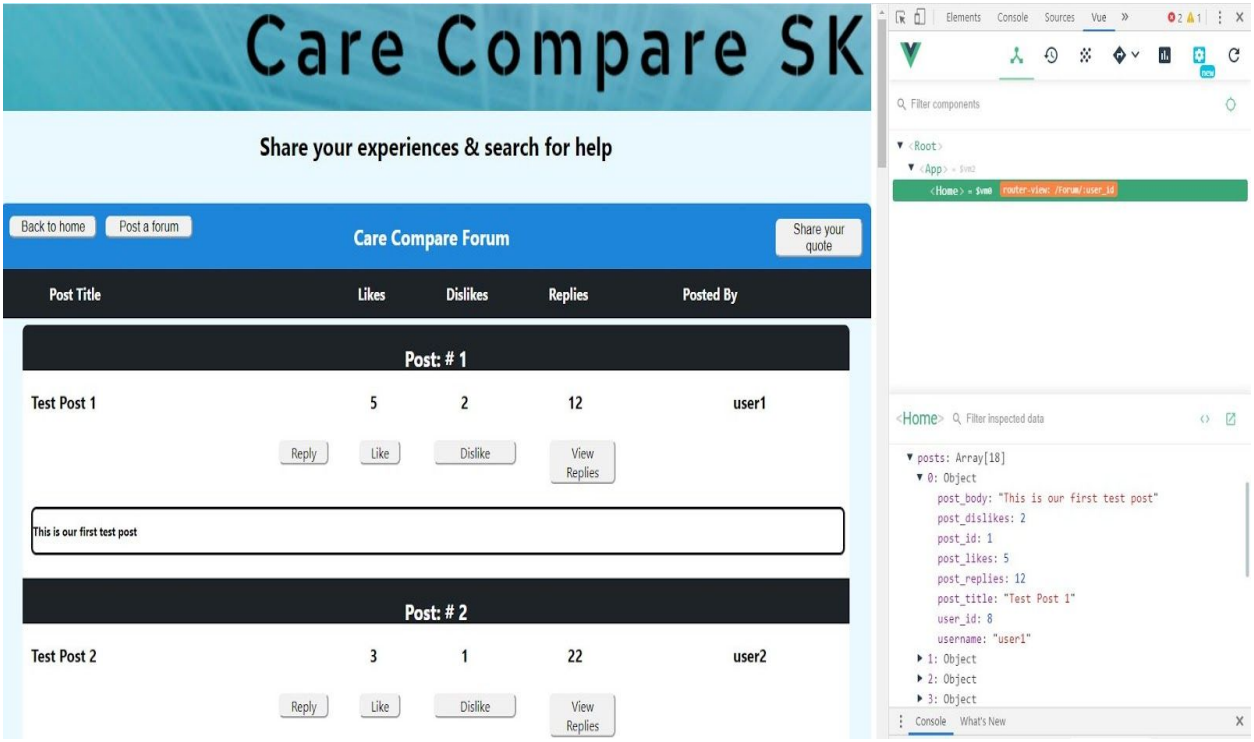

Figure 27. Example of Vue Developer Tools displaying data structures and values

Conclusion

Our Capstone project had many peaks and valleys like any engineering project should and will. We learned many valuable lessons about the engineering process such as planning, design, programming, testing, redesigning, and more which in our opinion is the main point of the Capstone project. Experiencing the entire engineering process as well as learning a new tech stack was so valuable to us and the knowledge gained from it will no doubt be invaluable to us in the future. We want to thank all of our advisors and people who helped us out along the way for making this experience as great as it was. Our final product that we created is very different from what we first imagined, but to us that makes it even better because we got to experience the true nature of engineering problems which are constantly changing and evolving.