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

Technologies

1) Amazon Web Services (AWS)

Our original plan for making our web application public was to use an AWS EC2 Ubuntu instance as a server for our application. We initially chose this route because it was recommended by other students who had used it previously. However, we had a lot of trouble as first time users and put a lot of time into getting our application running on our instance, but AWS proved to be too complex for our time frame and getting the instance running was taking too long to set up and was severely holding back the progress of the rest of our project. We had purchased a domain name from Registrar, set a static ip for our instance, got our Ubuntu instance running, set up security groups, and more and still we could not get our application running on AWS. On top of all of those problems, since we were first time users we did not know about AWS credits so it was also costing us money to run these services. We decided to abandon the AWS route and instead opened ports on Kegan's local computer to host both the web application as well as the Express API we built. Our recommendation for future groups is to make sure to leave a good amount of time for your group to refresh or learn technologies especially if they are new to you to ensure that you

have sufficient time to work through difficulties and bugs that may occur when working with new technologies.

2) Vue.js

We decided to implement a MEVN stack for our capstone project which includes MySQL, Express, Vue,js, and Node.js. This was an entirely new stack for both of us and our experience with it was very positive. Vue.js has a small learning curve in the beginning as you have to learn Vue.js specific commands, but after you learn those specific directives and commands, the rest is pure Javascript, HTML, and CSS. Vue.js acts as a container that combines all of the Javascript, HTML, and CSS in one application and uses hot-reloads as well as a MVC structure which allows us to dynamically see our changes without having to recompile our application every time a change is made. We implemented all of our front-end application work, all HTTP requests to our Express API, and all of our back-end Javascript functions within our Vue.js application. We highly recommend anyone looking to learn a new front-end language to give Vue.js a try. It is a modern language that has many useful built-in functionalities such as an MVC structure and fits into more modern tech stacks like the MEVN stack that we implemented.

3) Express/axios

We are using an Express server and Node.js to host our backend RESTful API that connects to our MySQL database. In our experience, express was very easy to learn and allows us to quickly make new endpoints to make queries to our database that allow us to retrieve, update, or insert new data into our database when we need to. We are using axios within our Vue.js application to make HTTP requests to our express API which has also been very easy to learn. Making GET and POST requests from our Vue.js application are simple one-line axios requests to a specific endpoint in our RESTful API and Axios handles the requests for us. All we need to do is tell it which endpoint to contact and what to do with the data being retrieved. We recommend learning how to create a RESTful API because it is a nice modern way to access databases and this approach is being used by many top companies such as google, Amazon, and Twitter.

4) XAMPP

XAMPP is a software that provides us with a mariaDB instance as well as an Apache server. This Maria DB database is where we are storing and retrieving all of our data for our application from. We chose to use a SQL database because we are very familiar with it from our experience gained in school and also

because many of our tables are related to each other and we benefit greatly by having a relational database to work with. This is a deviation from the industry standard MEVN tech stack as MongoDB is usually used as a database in combination with our other technologies, but we found no problems integrating MySQL into the stack instead of MongoDB.

Customer Acquisition/Data gathering

1) Insurance Companies

Our original goal was to secure data from at least 2 insurance companies so that we had a bare minimum to compare prices. However, insurance companies are not very forthcoming with information about their pricing and we were only able to find online quotes. Because of this problem, we decided to implement a new approach of gathering data from the users themselves. They are able to post blog posts about their own experiences as well as give real prices of their packages. We can then use this data to provide multiple price estimates both from the online quotes we found as well as what the users provide.

This implementation suited our needs well because it helped build a community of practice for people to go to for help with choosing insurance as well as solving our lack of data problem.

2) User Forum

We created a user forum as a way for users to give feedback and share their experiences with different insurance companies as a way for us to gain more data as well as create a community of practice that people can contribute to. Users can enter their unique insurance plan price gathered in real life to help us provide a more accurate estimate to future users. The combination of user supplied quotes and forum posts allows other users to not only make better decisions when choosing their own insurance plans, but also generates discussions and a sense of community within our application. This was not included in our original design, but we feel that it was an excellent addition to our application and dramatically increased the value of our application from a user's standpoint. We recommend trying to include the general users of your application in some way because they are usually an excellent source for data and can give real feedback on your applications design and bugs as well which is very helpful.

Progress Tools/Techniques

1) User Story Mapping

One of our earliest organization techniques that we used was user story mapping. This was an ideal starting point because it let us put ourselves into the shoes of the users and try to picture what they would want. However, with the lack of real users available to us, we found that this technique was not nearly sufficient in terms of collecting user requirements or keeping us organized and on track of our tasks. For this reason, We created a kanban board to supplement our story map as well as personas and low-fi/high-fi mockups.

2) Kanban Board

In our experience using a Kanban board worked extremely well for us. Kanban boards are extremely malleable as tasks can be added or removed as they change and having a constant visual of the tasks we need to get done kept us on track very well. Kanban boards are also extremely easy to set up and maintain and can be implemented on paper, whiteboards, online, etc.

3) Personas

We created multiple personas to try and describe and understand different user groups and what they are all looking for in our application. We thought that using personas gave us a better idea of what to create within our application and kept us focused on what really mattered instead of getting sidetracked on less important features or flashy looks.

4) Lo-fi/Hi-fi prototyping

We created multiple Lo-fi and Hi-fi prototypes to plan our designs for our application. In our experience creating these prototypes is always a good idea as it allows very quick alterations and changes with very little time or effort spent. We changed our final application design more than 20 times throughout the engineering process due to changing focuses and certain limitations, but were made relatively easily and smooth due to the use of prototypes. In our experience we recommend to use the planning tools that work best for you. Everyone has different work styles and different needs when planning an engineering project, so we feel that no one approach or tool is better than the other. Oftentimes the best tool of choice is based on the type of project or the type of people working on that project.

Conclusion

Our Capstone project had many peaks and valleys like any engineering project should. 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.