Final Report: The Process of Creating schedule-me-up

Comp 4350 - Winter 2020

Bonnie Tang tangb34@myumanitoba.ca

Brenna Epp eppb@myumanitoba.ca

Daryl Fung lerhxind@myumanitoba.ca

Earl Placido placidej@myumanitoba.ca

Emmanuella Osinaike osinaike@myumanitoba.ca

Jennifer Seo seoj@myumanitoba.ca

Kevin Hoang hoangk@myumanitoba.ca

Manasseh Banda bandam@myumanitoba.ca

Ovietobore (Toby) Oghre oghreo@myumanitoba.ca

Renz Cabusas cabusasr@myumanitoba.ca

Introduction

The purpose of this project was to create an application that makes scheduling meetings less time consuming. We created schedule-me-up, a web and Android application, that students, work groups, or anyone with a meeting to schedule can use to find the time that works for the most people within a group.

Our motivation came from our own struggles to find a time to all meet up as a group. We built this software in hopes that scheduling meetings will become easier for others. Although there are other scheduling apps available, these do not ultimately select or even list optimal meeting times. In these applications, it is up to the group to review all possible meeting times and select a final meeting time. In contrast, schedule-me-up provides the most optimal time for all group members to meet up. Our application also alleviates the time burden from the organizer - it distributes the responsibility among all team members by allowing each group member to fill in their own availability and having the application calculate the optimal meeting times.

Evaluate Success Criteria

In our initial proposal, we had the following success criterias:

1) Decrease the overall time spent on determining a meeting time by at least 50%
2) Decreasing an individual's time spent on inputting their availability by at least 50%

For our first meet up, it took our group 20 minutes to find a meet up time that worked for the majority of our members. Individuals had to input their availability in excel and someone had to count how many individuals were available at each slot. With the use of schedule-me-up, we conducted a test to see how long it would take all our team members to login, join a

group, input three availability times, and decide on a time. It took us less than 5 minutes to complete this test which is well beyond our 50% goal; that is to say, we achieved our success criteria.

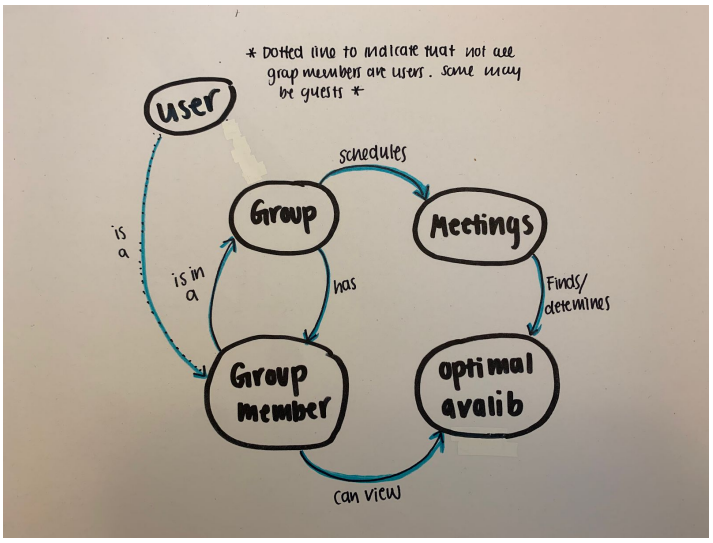
Defining Our MVP: Lessons Learned

Our project mostly lines up to what we envisioned - an app that made scheduling meetings less time consuming for students and work groups. However, our MVP, as defined by our user stories, was beyond the scope of our ability to accomplish within the time span given. We were not able to complete the following high-priority user story: *As a user, I want to be able to "login" as a guest when inputting my availability for a group*. In hindsight, we realize that implementing the login as a guest should have been one of the earlier stories we worked on. Allowing users to avoid the hassle of logging in has the potential to save time and increase the motivation for users to use our product.

If we were able to re-allocate our time, we would have preferred to have implemented the login as a user over the following user story: *As a user, I want to be able to create an account to keep track of my different groups.* We believe this feature should have been a lower priority and that the guest login would have aligned with your vision more closely.

In terms of whether to implement user login or guest login first, we believe there were many benefits that came with implementing the user login first. For instance, it helped us visualize how the models can relate to each other. It allowed us to consider the “happy path” and implement more straight forward functionality. This was beneficial in allowing us to familiarize ourselves to the project and the technologies related to it. Had we started off with

guest login, it would have been significantly more complex and our data model may require significant changes.



Data Model

For instance, there was the issue of how to deal with the inputted availability of guests. We decided that we wanted guests to be able to view their inputted availability even after they've closed and re-opened the application. We also determined that we did not want users to be able to manipulate other guests' information, but on the other hand, we still wanted guests to be able to update their own availability if needed. As a result, implementing guest login first would have been more complex than implementing the user login.

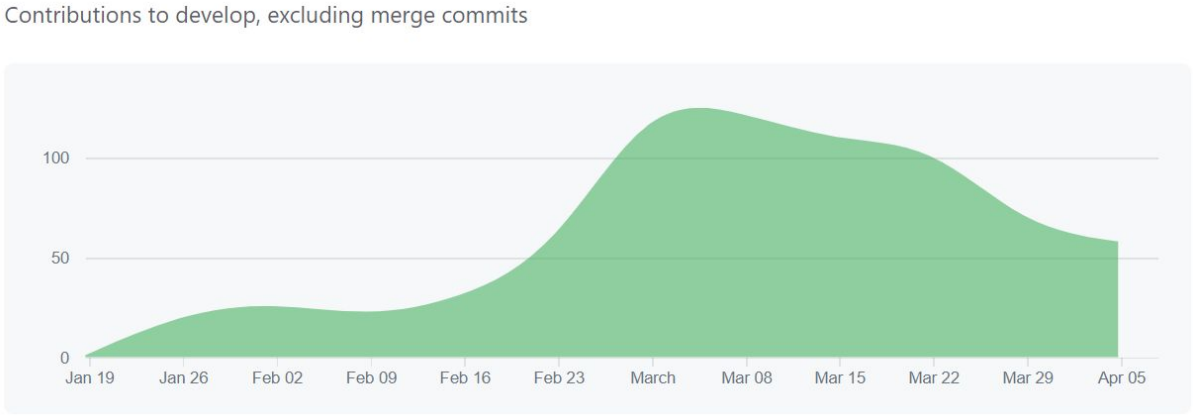
In hindsight, we also believe that allowing guests to login without any credentials may have been too complicated. Instead, we would have loosen the constraints and made users input at least a unique email address so that we could persist the guest information and allow the guest to modify their availability times without interfering with other guest information.

Moving Away From Native Android Development

We decided not to use native android development for three main reasons: visual consistency, language consistency, and to remove uncertainty. For visual consistency, we wanted to select React Native because we knew it shared styling libraries to React. Therefore, it would allow our web application and mobile application to look similar. In terms of language consistency, we thought it would be easier to have both our web application and mobile application in the same language. Finally, we chose to switch to React Native because many of our team members were either unfamiliar or had little experience using Android Studio. Since there were a few members who were knowledgeable in React Native, we thought this switch would be more beneficial to the speed of our development.

Testing Decision

Tests were not significantly focused on until the later part of our development process. Our group had a slow start to development and as a result, did not have a lot of features to test until later on. This was evident in our development curve.



Development Curve

Once our group began ramping up on development, most of our attention remained on implementation and not testing. We did not choose to do this deliberately. Instead, we got caught up in features implementation, and the excitement of development overshadowed the need to test. Due to the time limitation to finalize the implementation to contain at least a working MVP based on the user stories that we have, we also created more technical debt by focusing most of our attention on feature implementations rather than testing. As a result, our tests were pushed to the end. In retrospective, we would have liked to have focused on testing sooner. This may have helped us identify issues, determine development strategy, and much more. We also understand that leaving tests to the end implies that some areas may not have been tested as thoroughly. We might not be able to come up with more edge cases to test on the features that we have developed as well if we leave tests to the end because the features might not be as fresh as they were when we first implemented them.

Development Process: What Went Well

Our communication and collaboration as a team was great. People were willing to help each other out and everyone was willing to help; everyone on the team was very open and agreeable. Our teamwork was also great, everyone was friendly and communicative. As time went on we also completed more user stories. We got our client and server setup fairly early.

Development Process: What Did Not Go So Well

We did not pace the project as well as we could have. It would have been more ideal to have more development done at the start of the project. We believe that the slow startup of development was due to confusion on what individuals should do. Many team members did not start on development until a task was assigned to them. Individuals were unsure whether

they should work on mobile, web, server, etc. If we had decided on a development strategy sooner, things might have gone differently.

There was also a steep learning curve. Setting up environments and fixing all the errors that occur in the setup sometimes took a whole day of development or more, even with help. The initial learning of several languages, frameworks and libraries led to making simple mistakes and/or taking a lot of time on simple tasks.

Development Process: Steps we Took

We set up a “stand-up” channel in Slack so everyone could share what they worked on for that week, what they would be working on, and if they had any blockers. This allowed us to have continuous communication and feedback which we found extremely helpful.

We also decided to make smaller pull requests. Smaller PRs made it easier to review and understand code, which helped acquaint everyone to different areas of the code base. Smaller PRs also updated everyone sooner on the work that is being done.

What we Learned

We should have made more effort to invest time into peer programming sessions earlier on. In particular, it would have been helpful for project and CI/CD setup. We saw it as a waste of time or tasks that would delay development, as a lot of us had no experience in project/backend setup areas. Unfortunately, the decision to leave these responsibilities to only a couple of people led to untapped developer time as devs waited for setup to be done; a lot of us did not feel invested in the project early on. Once we started peer sessions we

learned that although some developer time is not maximized in the short run, it works out better in the longer run.

We should have attempted to define a smaller MVP/earliest usable/testable product. With this, it would have been easier to define smaller, more meaningful, and more concrete tasks to each person in the group. It might have also increased our investment in the project and provided a sense of accomplishment and progress throughout. Defining a smaller MVP could have also provided us a chance to establish earlier and more concrete deadlines to measure our progress and motivate those of us who like working towards a deadline.

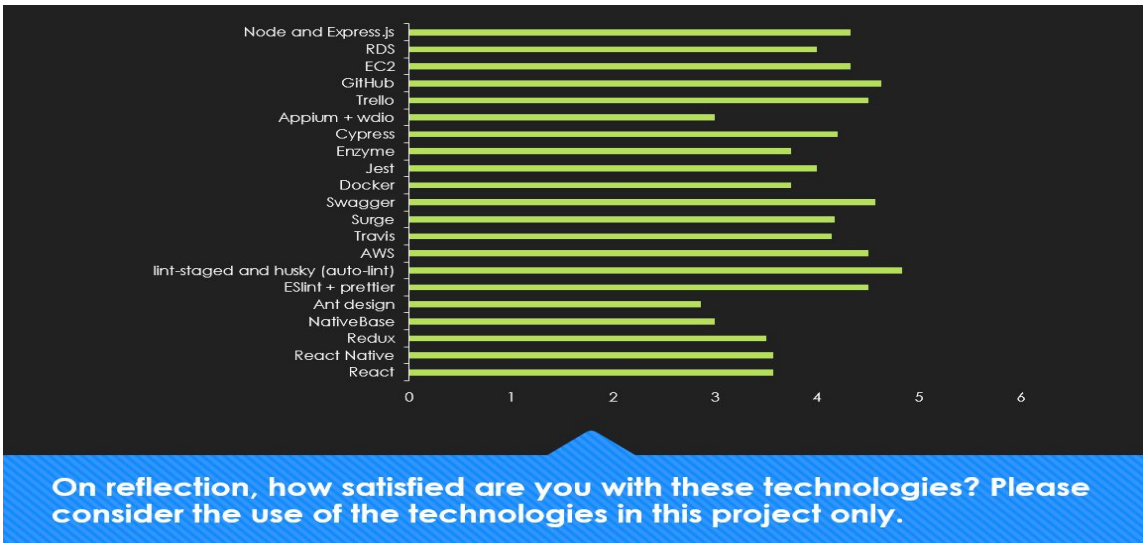
Thoughts About Technologies Used

Throughout our project, there was much discussion about the technologies we were using and if they were the best tools for the job. We formalized this discussion at the end of our project through a survey which measured how satisfactory we found the technologies we used.

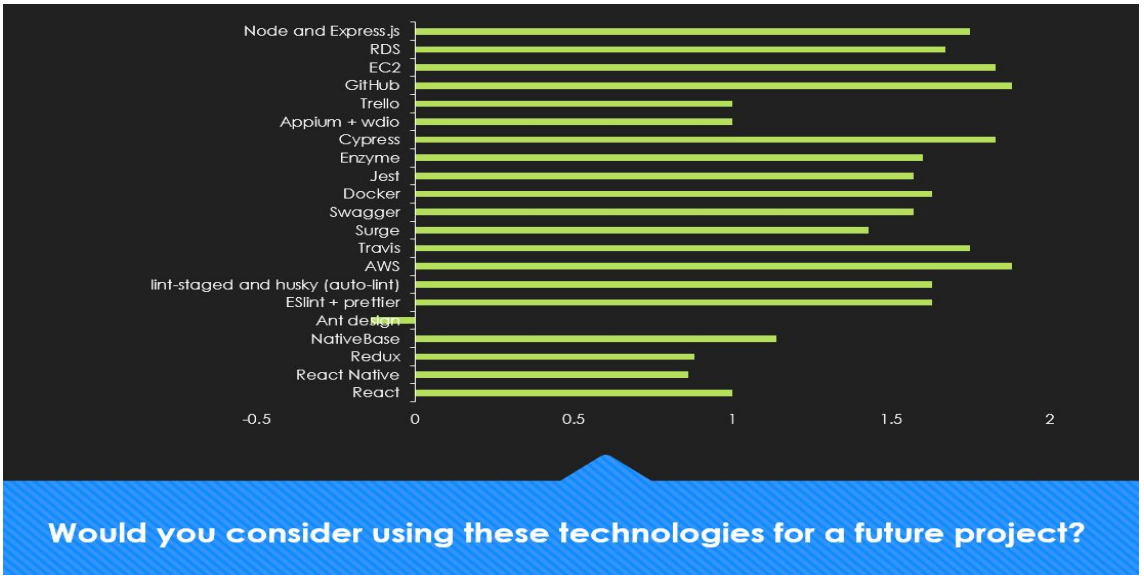
About halfway through our development, the web UI team discussed moving away from ant design because we were frustrated with its grid system and lack of responsiveness. We had to compare the benefits vs the cost of switching from ant design at that point in our project. We ultimately decided to take on the "technical debt" (as several on the web UI team thought ant design was) of continuing to use it, because we felt we might not be able to accomplish our goal in time otherwise. To continue using ant design then, we had to fix some of our pages, which were not responsive. We weighed the time it would take us to make these pages responsive vs. switching to a new framework and decided to try the first. For future projects,

however, we would definitely consider using something a lot more responsive and which has a better grid system.

We also considered switching from Trello to a different task management system. Trello did not have all the functionality we wanted and our group also did not use it as much as we would have liked. We did not end up switching however, because we felt that GitHub along with our weekly meetings provided us with enough communication and resources to complete our tasks.



Technologies Used Survey Graph 1



Technologies Used Survey Graph 2

Difficulties in Development: A Personal Perspective - Anonymous

The purpose of this perspective is to share my experiences during this project and talk about some processes I followed. I will also talk about setbacks and issues I had. Partaking in this project was a rollercoaster ride for me. I had highs and lows; sometimes I was so happy and proud with myself and sometimes I felt like a failure. Working in a team of 10 people is also new for me. It can be intimidating when it seems everyone around you is smarter and ahead of you.

One of my weak points is research and finding information for myself from the web. I would say I'm a fast learner if I'm taught or told what to do, but when I must go online to find information to learn I struggle. That was the most challenging area for me as I started to work on the project. I have less than minimal knowledge about JavaScript, and I never used React or React Native before, so trying to think of solutions to solve an issue was very overwhelming as I did not even know the basics to start working with. Usually when you get some little errors you know what it means or what to do to get rid of it, but usually it took me a whole day or more to get rid of those errors as most times I had to wait for a group member to take a look at it.

Even though I had issues learning the languages and frameworks I would definitely use the technologies we used again. They were efficient and could do what we needed them to do. For future projects instead of waiting to do something I know and trying to learn the technologies and languages on my own, I will work closely with my team members and observe what they do. I think I would grasp things much faster by observing and asking those who know what they are doing questions.

Reluctance in Leadership: A Personal Perspective - Anonymous

When our group was first formed, we identified the need to select a project manager. We knew that the group was far too large to simply rely on development without a manager. As a result, individuals were nominated by other team members to become a project manager. I was one of the nominees but immediately vocalized my desire to avoid being a manager. In the past, I had been the team lead on many group projects. As a result, the idea of being a leader fatigued me. I wanted a new perspective and hoped that this group project would be the perfect opportunity. As someone who hopes to get into project management one day, I learned a lot from not being a team lead on this project.

The most significant lesson I learned was that just because you're not a manager, it does not mean that you do not need to be aware of other parts of the project. When you're not a group leader, sometimes it's difficult to have a grasp on all the components of the project. As a result, functionalities can overlap and confusion can occur. As a developer, your duty is to not only code, but to tell others what you are coding while being mindful of what others are doing as well. It's easy to pigeonhole yourself and focus on just the parts that pertain to you, but the best development happens when ideas are shared and understood. For instance, I did not find out that someone else was having a similar server error that I was until I had spent several days on the same issue.

Conclusion

In conclusion, our overall vision of this project was accomplished. We made some mistakes which we realized later on through our retrospectives and regular communications, but we

were able to improve from them. We gained an invaluable experience that will help us in our future projects and career. We learned the hard lesson and consequences of overestimating our goals. Our advice to others would be to not overshoot their capabilities and to define smaller, iterative MVPs. It can be difficult to estimate development speed successfully - this could possibly be alleviated by defining the absolute smallest iteration. Additionally, finishing several of these can lead to more sense of accomplishment and ownership amongst developers.