Project Experience Report: Growing Gamers

Table of Contents:

1.0 Initialization Experience

Getting started with this project was a daunting task. Deciding on an idea, researching, using new (to us) technologies, and organizing an effective plan were big challenges for a couple of guys who aren't necessarily big fans of paperwork and pen-pushing.

The main criteria that guided us when selecting an idea were scope, feasibility, and usefulness. We did not have a solid idea to pursue at the start of the semester and felt rushed to come to a decision. If we were to tackle this again we would spend more time brainstorming prior to the beginning of the project, if we had we likely would have decided on a different idea we felt we could accomplish more fully. Topics we were considering included robotic test equipment, image recognition in safety roles, and personal assistant apps. We rejected the possibility of any system requiring physical components (i.e. robotic test equipment) due to the current volatility of shipping and the wide scope involved in such a project. Image recognition is a field we have no experience with and so we shied away from it fearing we would struggle to meet deadlines. We ended up settling on personal assistant apps since we felt it was feasible for us to create a useful product in this field. Our first idea was to make a workout assistant that would scrub the web to find customized advice and exercises. After deliberation, we decided instead to incorporate a passion of ours into our project by focusing on gamers and we chose to work on a system to help players regulate their habits to improve their game-life balance.

Exploring technologies that existed in the niche we envisioned revealed that there is a distinct lack of tools to help players better balance their lives. It was encouraging to see that our concept is novel and in an uncluttered space.

Selecting technologies to realize our project was an interesting process, many technologies we wanted to use had unexpected challenges and barriers that we had to overcome. We needed a technology that would enable us to overlay over a game, display information to the user, and that could collect game data and events. Overwolf fit the bill, without it our options were slim. Overwolf offers an in-game event listener, an automatic trigger to launch our app alongside games, an ingame overlay, and a platform to release our product for deployment. Overwolf requires us to submit a project application to be white-listed and gain access to their developer tools. It was quite stressful knowing our alternative options that offered similar functionality were limited, if we did not get whitelisted we were in trouble. Luckily we were whitelisted and free to move on with our project. Overwolf also suffers from weak documentation and finicky best practice displayed by their sample app. We opted to build off the Overwolf sample and use Node.js, we had never used Node.js but we did have experience in html, css, and javascript so the learning curve was not overly steep in regards to the languages

we were using. Working with webpack, creating a multi-window app, and working with a manifest file were challenging aspects.

Developing our plan and creating our documentation came with its own set of challenges. First, we are not pen pushers, repetitive and sometimes redundant documentation gets old quickly. To counteract our reluctance we relied heavily on visually oriented documentation which allowed us to work in an organized chaos style where we were free to constantly brainstorm, our kanban board is a good example of this. We also planned our documentation for a single MVP at a time so we could dynamically choose what the next most important work would be. Second, we faced an issue where as we developed our documentation we would lose track of the scope and the intent of our project. When we caught ourselves veering off we would get into a team scrum and rationalize the work we were doing to help us get back on track. Lastly, due to the flexible philosophy we were operating under, changes arose throughout our project which lead to updates being needed in various documents.

2.0 Execution Experience

2.1 MVP 1 - Use Some Technologies

To complete the first major MVP we expected that we would have to; get the Overwolf app to launch and count the time the game has been active for. This would require us to install and figure out the Overwolf developer tools, then install and modify the example app to meet our needs.

Making use of the Overwolf developer tools was quite an easy process as was installing and using their sample app. This MVP was achieved relatively painlessly only requiring minor modification to the sample app, we stripped it down and implemented a static javascript timer that would count how long the app was active for (it did not truly collect game data). As we completed this MVP we focused on learning about the app's structure and how the window to window navigation worked.

If we were to tackle a similar problem again we would spend even more time trying to understand the functioning of the app while we had easier deliverables, this would have helped us moving forward.

2.2 MVP 2 - App sided functionality

The second MVP was to have a functioning client side app that could (1) collect game events through the Overwolf API and (2) display a message to the user based on the collected game events. To achieve this we expected to have to make adjustments to the first MVP’s data collection and display features, use the Overwolf API to collect in-game events, style the app with CSS, and add some client side logic to select what message to display.

This MVP challenged us. There was a fair bit of difficulty with beginning use of the overwolf API to collect in-game data and with using the Overwolf file system API, after a healthy dose of searching online and double that amount of experimentation we got it figured out. Learning to work with typescript was a small learning curve as well but being a sister language to javascript the pieces fell into place quickly. While little was accomplished from this work we did learn more about the way the app worked and woke up our long dormant knowledge of CSS and HTML.

It was now that our inexperience began to become apparent. We were unfamiliar with the way the app was structured, had no real experience working with typescript, and didn't understand the role node.js played in the whole thing. Instead of focusing on tackling even a simple MVP like we did, our time would likely have been better spent researching and learning more about the relevant technologies. A major thing we would do differently is avoid duplication of work. We both would focus on the same problem separately which resulted in duplicate work, instead we should have done more pair programming and delegated tasks better. Nonetheless we managed to create a minimally satisfying MVP.

2.X Overwolf Challenges

Unfortunately the issues surround our use of the Overwolf platform and API's are worthy of their own section. We expected the platform would be reasonably refined, our initial research led us to believe that the platform was very focused on being accessible and welcoming to new developers. We expected the API's would be useful, usable, and up to date. We expected strong documentation to lead customers through the use of their product.

What actually happened? The platform had poor beginner friendliness due to minimal documentation and introductory material. Example material was either extremely basic or overly specific. Examples provided were outdated and often deprecated. Documented features would often have duplicates, some listed as deprecated, some not listed but still nonfunctional. We lost many hours implementing and debugging deprecated features that were not listed as deprecated. bridging the gap in our knowledge of the program was difficult, the learning curve was harsh and support for newbies was virtually nonexistent. Following the Overwolf sample app required us to use webpack and an MVC architecture that was poorly documented. Understanding how webpack worked was challenging although with some research our understanding became reasonable. How the MVC architecture worked was highly challenging, figuring out how windows could be displayed and hidden and how to pass information between windows required a lot of time. Overwolf lists the use of typescript as best practice but webpack packs the app in javascript, this discrepancy meant that we could not make use of many node.js libraries we needed; Twilio was incompatible as was Filesystem i/o. This made the implementation of a server critical.

What we learned? Sometimes you have to use services you don't necessarily love. Despite Overwolfs shortcomings we needed it to be able to launch the app with the game and to have an easy to use platform to distribute our app over. If we had known how challenging working with Overwolfs API's would be we would have selected a different project.

2.3 AI

Once the second MVP was reached, we began looking into the implementation of AI to help provide relevant and useful messages to the user. We expected this would be a challenging thing to implement but expected we could make something work. We were looking at using python and the Riot Games API to build our model.

As it turned out, we were in over our heads. The model we developed to predict match outcomes underperformed and integration to the current Overwolf app proved to be a challenge that was sure to waste more time than it was worth. In the end we shelved the idea opting to work on more valuable features for the time being. Despite our failure, the technologies selected worked well and were easy to learn. The Riot games API was straightforward to use and python lent itself well to the tasks we were accomplishing.

The AI could have gone better if we had waited longer before working on it and it may still be in the books for future work. Our main application was not nearly developed enough to begin integrating the AI and we did not yet have a good sense of how the technologies we were using fit together. Our skill inventory did not include much experience with AI, more research and reaching out to more knowledgeable people could have helped us better succeed. Pushing off other work to pursue the AI was a waste of time, other much more important work was sidelined for something that ended up shelved.

2.4 Course Correction

Now that the AI idea was shelved, we decided as a team that we should modify the aim of our project. The new direction was to focus on young gamers and their parents and attempt to bridge the gap between them regarding gaming. It was expected that this correction would allow for us to retain our existing mission, retain our existing features, and open us up to more feasible and valuable future work.

This course correction worked well. We were able to implement a feature set that we could be proud of and that were within the bounds of our technical capabilities. Our mission remained the same, improving the wellbeing of gamers and their communities. We simply shrunk our focus.

2.5 MVP 3 - Server side initial setup

The third MVP required us to set up a server that could handle persistent data storage. This server was expected to allow us to implement future features more easily and in a more scalable manner. We expected the server setup to be challenging due to our limited experience.

While it was reasonably time consuming and there were a number of small issues to overcome as we worked through it, setting up the server went well and was not as challenging as expected. A lot of time was spent reading and researching the technologies we intended to use on the server (we learned our lesson about going in blind) and so the setup and configuration was not particularly challenging despite the work involved. It helped that the code and configuration of the server was fully our own work so we had a good understanding of how everything worked and fit together.

The server enabled us to create better and more valuable features more quickly than we expected. Something that would have been beneficial is to set up the server earlier, during the time we were working on the AI. At this stage we began delegating work better and naturally took the lead on different aspects of the project, Bryden led with the application logic while Shane led the back end. One way to improve our delegation of work would have been to increase the frequency of scrums to remain better informed about the other's work.

2.6 MVP 4 - Parent portal and SMS

With the server functional our next major milestone was to have a parent portal up and running. It was expected that this would take work on the server code as well as HTML/CSS work. We also expected to make use of the Twilio API for sending out our first SMS messages. We hoped to implement cell authentication for this MVP. Developing this MVP went mostly as expected.

First we modified the html and css of the app and created a popout menu with various options on it, this was our first MVP regarding the parent portal. After this we created a web page that lived on the server to host the parent portal so that parents can access the settings from anywhere. Using Twilio was easy and well documented, all we had to do was make an account, buy a phone number with our trial credits, and generate some codes. Calls to the API from the server were straightforward and easy to write. We did not get security features ironed out for this milestone but overall this MVP went smoothly.

Good communication and delegation of labor contributed to our success. Our technical skill and confidence improved wildly over the course of this project and it was now that we could see our improved understanding resulting in higher quality work.

2.7 MVP 5

This MVP was meant to improve the scalability of our project. We expected to have to migrate logic from the app to the server as well as modify the way data was

stored and used. It was also expected that we would have to implement some form of security for accounts. The workload was expected to be quite high but feasible.

Migration of logic was wrought with bugs and was labor intensive. Communication protocols had to be developed in order to provide the app with the correct data at the correct time, this was challenging due to the asynchronous nature of javascript. The more we did the faster we worked as awaits and promises became less and less mysterious to us. We implemented cell number authentication for the parent portal and as the account system for the app.

This MVP went smoothly.

3.0 End Result/ Reflection

All in all there were a number of expectations going into this project. We did not expect too great of technical challenges but rather expected to find challenges regarding implementation of features, not in using the underlying technologies. We expected to have more done sooner. We thought we would have enough manpower with just the two of us. And lastly we expected to succeed.

What actually happened overall challenged most of our expectations. Technical challenges were aplenty, the technologies we used caused most of our frustrations and caused the biggest challenges. Actually implementing logic and features was frankly quite easy, it was the technologies themselves that caused pain points with incompatibilities and deprecated code. Working on the AI component early crippled our schedule, we spent too much time working on a feature that didn't make the cut for project day and then had to play catch up for the remainder of the semester. We made it work with a two man team, surely a poor third member would have slowed us down but a competent third member would have been great. Managing documentation and code split between the two of us resulted in heavy responsibilities. A dedicated member for front end and documentation, a dedicated member for server code, and a dedicated member for app logic would have been ideal. Despite the challenges encountered one thing went precisely as expected, we persevered and succeeded.

Our experience taught us the importance of project planning, we had some experience with projects in other courses but never in such an organic and unsupervised manner. The kanban board we used was essential to our organization, version control was unbelievably useful, the importance of communication became more and more clear, and the value of research became obvious. The use of documentation to lead our implementation was excellent as well, by creating modified communication/structure diagrams we were able to harmoniously split work without duplication or compatibility issues. Also the creation of the after action report should have been done as we worked through the project, writing the whole thing at the end has proven challenging. We learned about our capabilities, as it turns out we are

capable and competent in the ethereal world of software. Softwares inherent lack of physical indicators of work done and success make it difficult to truly see our strengths and weaknesses. We learned the power of collaboration, not only through the way we brought each other up throughout the project but also in the way we struggled with only a skeleton crew. We learned that burnout is an ever present issue that must be mediated through effective scheduling and load splitting, after a certain amount of work productivity tanks and no matter how close a deadline might be you should probably take a break. Sometimes a break means twenty minute, sometimes two hours, and sometimes it means a day away from the keyboard. The importance of testing with an iterative MVP to MVP process seems to be highly beneficial, especially with multiple parties working on code that needs to work with each other. Clean coding practices were not respected as well as they could have been, particularly with first iteration code there are vast improvements that can be made. Lastly, we learned that without user input, without people focused design, it is difficult to choose a direction. When your product is designed to help people you need people to give you drive and help you understand what they need. Trying to make a product good for all gamers was too broad. By having a narrower focus and collecting specific feedback, the work we did was better and more meaningful.

3.2 Shane

There are a few things that stand out to me that were learned throughout this project. First, the scale of what I can reasonably accomplish. Second, The importance of trusting your team members. And third, how incredibly much I am not the customer.

The scale of what I can accomplish is a two way lesson. I learned about how much I can accomplish even with my limited knowledge and experience, but I also learned how hard it is to bite off more than I can chew. I can accomplish a lot, I am capable of learning damn near anything, I am capable of having vision and I am able to execute. On the other hand, I can be overzealous, I can set my sights too high and make it difficult to meet a deadline. There is a balance that I need to keep in mind when undertaking tasks. I will become more capable as I gain experience but I need to inventorize my skills and take that into consideration when undertaking work. If I lack a skill or set of knowledge I should strive to acquire it and do as much research into the topic as I can prior to tackling the problem, this would enable me to be more productive and move with more confidence.

The importance of trusting your team was huge for this project. You can’t do everything yourself even if you would like to and you have to trust the work of those who are making up for your shortcomings. Shortcomings come in the form of skill, time, and patience. They all require someone to fill in the gaps where you can’t. I could not have tackled this project alone especially given the time frame we had to complete it in and other life commitments we had to balance around it.

Lastly, from user testing and market research, I learned how far removed I am from the customer. Things I would have never thought of confused testers and some of the questionnaire responses we received showed priorities far from what we expected. My experience with technology and my worldview is far from average in the grand scheme of things. I have to remember my bias when doing work and give a great deal of importance to feedback from real users.

One more thought, writing AAR's as you work through the MVP's rather than writing the whole thing retrospectively would have been a good idea.

3.3 Bryden

Taking on this project with complete control to decide how our progress will look like and what directions we will go in, alongside Shane, was an eye opening experience. It was really fun to have this project become a work of love and implement features that were exciting to myself. Besides exploring the freedom to create as I wanted, this project also taught me some valuable lessons. The importance of organization, prioritization, and proper communication in a work environment, as well as the importance of keeping code clean and SOLID were taught to me.

Initially, keeping ourselves organized, and communicating was a lower priority than it should have been. This resulted in me mismanaging my time. I did not communicate with Shane exactly my plan to make progress on the project and let myself make assumptions about what he would do. So issues were made with us doing the same work or making designs that would not be compatible with each other. As we increased the frequency of our team meetings these issues were resolved. This additional communication made me feel more confident in the work I was doing and helped me direct myself better to work that would benefit the team the most.

An unexpected lesson I learned was about getting too caught up in the details. When implementing new features I found myself getting attached to them. I would want to add in more functionality to it to make it as great as it could be, create something more than what was needed. I'd be so focused on the small part of the project that is one single feature that I would forget about the rest that needed to be done. This kind of mentality helped me work longer and harder because it made me enjoy what I was doing, but learning how to step back and widen my vision was an important lesson that I learned.

As our project became increasing in size and complexity, good coding practice became more and more valuable to us. When our files become very large, organization of our functions and good comments help us keep up our productivity. Dead code or large, unnecessary comments also became more and more detrimental to our work when we were working on large files. Keeping our project clean from this was not very valuable to me at the beginning, but became very important to me very quickly.

One last lesson I learned was about our scrum meetings. Oftentimes we would meet with not alot new to show for all of our work. All of our progress was made behind the scenes. Something that I needed to accept was that we couldn't always have an exciting and interesting demo to show for all of our scrums. Rushing our work to have something to present was detrimental to our overall progress. Allowing myself to not think of scrums as some kind of deadline but instead a friendly meeting to talk about my progress was something I learned to do.