After Action Review

URStreamSight

Raymond Knorr, Avery Cameron, Noah Rowbotham

Version: 1 2021-04-09

Table of Contents

What Was Expected

Our Purpose & Objectives

This project came about as an opportunity to build a software solution for the company Prairie Robotics, so ensuring the success of that solution immediately inserted itself as one of our primary goals. A specific way of putting that goal would be to create a method of storing and displaying recycling quality information on an individual household basis, and to create infrastructure for municipalities to take this information and make educated decisions about how and where to spend their resources to improve recycling habits. To put it simply, we were trying to clean up curbside recycling at the source of the problem: the actual people recycling.

Besides that primary goal, we had several other objectives. We also wanted to document the entire process of building the project to ensure that we were making the correct decisions along the way, as well as to have a record of what we completed to pass on to Prairie Robotics at the end. We wanted to learn industry standard practices to use them in the project, benefit PR through their usage, and benefit us through our new skills. Above all, we wanted to work on a problem that is real-world applicable, useful, and important.

Stakeholders & Project Allies

We identified that the target audience for our project included the municipality workers who would use our app; Prairie robotics, and specifically Sam Dietrich who was our main contact there; and Tim Maciag, our Capstone instructor and one of our mentors along the way. The project was to be built for all of these people in different ways.

The people we envisioned to be directly responsible for the success of the project are the three group members Avery, Noah and Ray. Second to that, we thought that Tim, Sam from PR, and Dr. Yow would have indirect effects on the project by giving us guidance or other help.

Our Planned Timeline

When we started planning for StreamSight we knew we wanted to build a strong foundation and understanding of our project before we started development. We gave ourselves two months solely to design, discuss, and document our project. Then, once we have the understanding, we could use the last two months to elaborate on our target MVP and build an action plan for development. By mid-December, we wanted to be writing code shortly after completing our finals.

We intentionally gave ourselves a break over the holidays to rejuvenate our brains, and then planned for full development to start by early January. Our first MVP for the API/frontend web application was slated for early February. Then following that, the back end and machine learning should be at an MVP by mid to late February. Finally, we wanted everything completed by mid March, except for machine learning which we gave ourselves time to improve right up until April.

Facilitators & Barriers

We believed that low quality data or a lack thereof would be our most imposing barrier to success. We planned extensively for what we would do late December/early January if there wasn't enough data to start training a model. This flowed into a further concern of the amount of time and effort it would take to annotate all of the data once Prairie Robotics had collected it. Aside from machine learning concerns, we were also very concerned that our application would violate privacy law or be unethical. We were having multiple meetings to try to avoid running into such issues.

What we believed to be our facilitations to success was the access to AWS and credits for implementing our back end, hosting our app, and training a model using SageMaker. We were also confident that by integrating with Prairie Robotics' current platform, we would have access to lots of good code to review and learn from. These references would then help facilitate our development substantially.

Our Intended Results

At the onset of our project, we had a good idea of the types of outcomes that we wished to create. These involved some sort of filter for separating out low quality images of recycling, a score that would evaluate the quality of a bin tip's recycling, an API for facilitating uploading of images from the truck to the cloud, and a front end with various metrics and a map page that visualized the recycling scores on a neighborhood by neighborhood basis. We knew that our general outcome had to be a solution that could be reintegrated easily into Prairie Robotics existing software suite.

The products that these outcomes would involve would be a front end web application for the municipality worker to view metrics of contamination data, a machine learning model to identify contaminants in the bed of a truck, and an API connected to a database.

What Actually Happened

Project Purpose & Objectives

After completing the project, our list of objectives largely remained the same. We believe that we did create a well-made, scalable software solution for Prairie Robotics that would be able to integrate into their existing tools, but we also did not have the foresight to predict that we would enter into a Mitacs agreement to help compensate our efforts in building an app for a company. We also learned how to apply personal skill within a business environment, have professional weekly communication, and balance a multitude of different views about the future direction of our app. The main goal of creating a scalable solution for Prairie Robotics was also augmented to include creating a set of files that would be capable of recreating the entire cloud infrastructure from scratch, dynamically. The creation of the bin detection model was also an added objective when other methods were not available.

Stakeholders & Project Allies

The audience for our app largely remained the same over the course of development. The focus on building an app that would be understandable and usable by municipality workers is still our primary concern, while having it be useful by Prairie Robotics, and understandable by Tim being separate concerns.

In terms of people helping with our development, we found out that Dr. El-Darieby would be our primary mentor. It was his idea that we should set up a Mitacs application to allow us to be compensated for our work, since we were not going to receive any of the intellectual property. The Mitacs application and process gave us some insights into where we should take the machine learning, and one of Dr. El-Darieby's graduate students, George Daoud, gave us some advice for our machine learning as well. We received far less support from Dr. Yow than we anticipated, but this is purely because he didn't end up being one of our mentors. Sam and Prairie Robotics gave us a lot of support, and another PR employee, Matthew Bariault, helped with the contaminant classification and bin detection pipelines. Sam even organized to have a code review done of our API from an outside contractor. This gave us some good insights into potential improvements that we could make in terms of code organization, security, and functionality.

Our Actual Timeline

We are quite happy that we followed our initial schedule quite accurately. We did have to shift some deadlines back and forth, but overall it went exceedingly well. The changes that we did make, we now understand why and looking back it is fascinating to observe our project timeline with "wiser" eyes.

We initially stated that our design and documentation would be complete by late October. By October, we hadn't even completed lo-fi or hi-fi prototypes and were genuinely concerned how far back we had fallen. We buckled down, worked really hard through November, and by the end of the month had some great results. We were finally at a point where we felt the "why" of our project was well defined. We also had completed a lot of empathy mapping and prototyping to really zone in on our target audience and design concerns.

Development started in December as we had hoped, it could have been sooner but was delayed by our Mitacs application. Our involvement with Mitacs and applying for a grant significantly increased our workload. The application process essentially consumed a whole week of late November/December which pushed back a lot of our other projects. While this was a lot of work, it was a good problem to have.

Starting off in the next term, we worked a lot on development. A huge early success was Ray completing our web app front end before the end of January. This was way ahead of schedule and freed him up to work a lot more on the API. We mostly stayed on schedule except for a bit of a delay training a contaminant classification model. We had lots of data, but were waiting to get it annotated by SuperAnnotate. We didn't train our first model version till early

March, almost a month behind schedule. In the end, we managed to put in extra work and had StreamSight functionally complete by our second project bazaar. We spent the remainder of the term updating documentation, working on our presentation, and testing selected parts of our code thoroughly.

Facilitators & Barriers

We had many new facilitators introduced during the span of our project. Through Prairie Robotics help, we had our API reviewed by an industry professional and got so much good feedback. We were also awarded a research grant by Mitacs which allowed Prairie Robotics to invest $10 000 into data annotations from SuperAnnotate. This helped immensely with the preparation of data for model training. We also got tons of support from Prairie Robotics machine learning technician Matthew and from George Daoud, one of Dr. El-Darieby's PhD. students.

Unlike we had predicted, image quality and the collection of data barely impacted our project. The same goes for privacy, which ended up being a relatively simple problem to address. The privacy regarding recycling has had a supreme court case where they ruled recycled material and the collection and inspection of it is not a violation of privacy. We di however have some development issues that took substantial effort from the three of us to resolve. First, we had plenty of issues completing SSIO and getting it to run remotely on a truck operating North of Saskatoon. This delayed data collection and halted completion of the back end portion of StreamSight. We also had issues getting the built-in GPS on the Aaeon onboard computer to interface with SSIO. Nearly a week of work was spent just getting GPS operational.

On the API side of things, deploying the API as part of Prairie Robotics monorepo didn't work out and has to be deployed from its own repository. Although, the front end successfully deployed and is integrated with Prairie Robotics. The API also initially had really slow response times and improving took some analysis.

Project Results

Our actual project results do not differ too much from our intended results, with a few exceptions. We created a front-end that is understandable and usable by a municipality analyst, created an API with a database connection that is used by that front-end, and created a contaminant classifier that is able to find contaminants from pictures of recycling in a truck bed. As well as these, we also developed documents about potential third-party integrations for Prairie Robotics, setup documentation for initializing and connecting GPS to the onboard computer, created a bin detector pipeline for parsing the video stream coming from the truck, created a CI/CD pipeline for quality assurance and deployment of our code, wrote tests for our API and onboard computer, and created lambdas capable of reading and writing messages to/from an SQS queue on AWS.

Team Achievements

Our Steps To Success

There were various activities and steps that contributed to the overall success of the project. The weekly touchpoint meetings with Prairie Robotics helped to maintain a clear vision of what the next task was and helped to avoid scope creep. Delaying some of the completion of our design and documentation when narrowing down our why and exploring the possible directions of the project also helped us to be more focused and involved. Taking feedback from our scrums, vlogs, and the project bazaar seriously helped us to continually improve and address concerns and keep our project on track. The formulation of a better action plan after the first project bazaar was extremely helpful. The incorporation of user testing to improve our frontend and having Prairie Robotics provide client feedback was extremely valuable.

The Highlights

The highlight of this project for us is the finished product but also the steps that were involved in reaching it. The Bin detector model achieving 99% accuracy and allowing us to easily take in images of recycling from each house was extremely valuable. Having completed this early helped us to receive more recycling data. The implementation of our CI/CD pipeline on GitHub was helpful in keeping our deployed apps up to date and saved us a lot of time in the long run. Having the tests run on our pipeline was going to be a larger job but we found a solution that provides fast results and easy setup. Testing of the API and achieving 95% coverage and having all tests pass is great for the confidence in our API and ensuring its reliability. As well, having linting is a highlight due to it providing us with code that matches Prairie Robotics standards and is more maintainable which was one of our goals. The Heat map in the frontend was a great addition and helps visualize the data clearly and provide important information to the municipalities.

Future Improvements

What Could Have Been Better

Throughout the project, we found ourselves learning a lot about how we could streamline all aspects of the project. In particular, we identified a few areas that we wished we had spent more time or thought. We believe that we didn’t spend enough time on our “Why” in the early part of the project. People generally don’t care as much about the complex problem that we are solving. We also could have negotiated our partnership with Prairie Robotics better. We were going to do the same amount of work for no compensation before Dr El-Darieby persuaded us otherwise. Specifically in regards to testing, we could’ve started testing SSIO earlier in development to make sure that it was functioning acceptably. Finally, we believe that we should have had basic versions of our frontend and API hosted on AWS much earlier in the process. We built large apps and tried to integrate after, which led to a lot more bugs and other integration pains than we would have liked. Our API is still not hosted on AWS as part of the monorepo, although we found a workaround.

What Is Going to Be Better

In future projects, we will work harder to define project goals, purpose and the ‘why’ earlier to ensure the vision of our projects. We will also carefully evaluate advice and recommendations from 3rd parties while staying true to our clients needs and the overall goals of the projects. As well, more focus will be put on staying agile and getting feedback quickly and continuously. We will try to have an MVP together rather than trying to perfect all the components separately.

Advice for Future Groups

Our advice for future groups would be to communicate with stakeholders and ensure that they understand your project so that they can provide feedback and help spot errors early.

Having weekly meetings is great, even if not much has been accomplished. They provide important information and updates and help ensure everyone is on the same page.

Document all stages of the project and ensure that your project starting documents provide a good foundation. Code documentation will help you catch bugs, issues, inconsistencies and help other team members understand various decisions.

Work regularly on the project even if it is only a little bit during busy weeks. This let’s you continue to understand your project and if you run into any bugs or problems you can think about them and research them during free time. This also helps you to maintain your schedule and achieve the goals set for your project.

Having a strong CI/CD process is great to support quality assurance and easier deployment. Setting up GitHub Actions provides you with free CI/CD right in your repository and many powerful open source checks. Adding in linting helps to make sure quality of code is high and reduces technical debt rather than if you added later and had to fix a large number of inconsistencies and styling issues. Testing is great as it provides more reliable code and if you are deploying in your CI/CD pipeline, the code is less likely to have bugs. Write tests early and often and look at different test coverage metrics to see how progress is going. Using the continuous deployment feature means you don’t have to take time to setup your website or your API and add additional configuration as with manual deployment. It saves a lot of time in the long run and if you are testing your code before-hand then the deployed code will be more reliable. If you are doing deployment, it can help to have a testing environment for your frontend for end-to-end testing and then deploy to production. Overall CI/CD pipelines are great for code quality and improving deployment times and freeing up time to develop.

Closing Remarks

Our Actionable Recommendations

Project Definition and Communication

As soon as you think you have a solid idea for capstone, immediately work on documenting it. We found the templates that Tim provided a great start. Fill all of them out as best as you can. The more you write about your idea, the more you think about it, and the more issues you find early on. Then, take everything you think you know about your project and try to talk about it with someone. You should be able to clearly inform that person on your idea. They should be able to understand your project and its purpose. Get them to give you feedback, and if that feedback doesn't make sense to you that may mean they don't understand your project like you do. Repeat the process of documenting, defining, and communicating your project and you will greatly improve its quality and make presenting it far easier.

Value Your Work, Take Feedback Seriously & Prioritize the Client

Value your work and your ideas no matter how small they seem to you. If you are working with a company, ensure that you are getting some form of credit and compensation for your efforts. Even if you do not think it is a big deal to do the work for free, still push to get compensation because it is only fair to you to be treated like your work has value. Also, listen and take everyone's criticism and feedback seriously. We had a lot of people want us to go in all sorts of different directions, and they all had great ideas. Eventually, we had to decide what was best for the clients and how to best meet their needs with respect to the feedback that we had received from stakeholders. By prioritizing feedback from the client and ensuring that we were receiving feedback from them was helpful in ensuring the direction of the project stayed on course. Prioritizing the client and continually working to improve is important for the success of the project.

Fail Fast

Trust in the agile method and fail fast. We waited too long to assemble an MVP because we saw flaws. We thought if we just spent one more week on the API or one more week finishing SSIO that the MVP would be better and that would be a good thing. Instead, we should have had something usable, no matter how flawed, completed early so that we could get a "feel" for our product and get additional criticism early. You can't just communicate regularly and work in sprints and call it agile development. You have to commit to building "drafts" of a product and accepting the fact that it is going to suck for a while before it is any good. We were trying to do agile, but we didn't follow through.

Document, Document, Document

We constantly wrote everything down. We documented our ideas, we made checklists, we filled out all the standard documentation, we wrote down the feedback we got from our user testing. Documentation is probably the most boring part of the project, but it is an important investment into the success of your project. Documentation is an investment because it usually isn't useful right away. Its usefulness comes up months later when it reminds you of important ideas, helps explain something that you forgot how it worked, or helps you reflect and develop new ideas.