Spring 6-9-2023

EPL Card Reader Capstone: The Strengths of Partner Programming from a Team Leader's Perspective

Zach Yost
Portland State University

EPL Card Reader Capstone: The Strengths of Partner Programming from a Team Leader's Perspective

by

Zachary Yost

An undergraduate honors thesis submitted in partial fulfillment of the

requirements for the degree of

Bachelor of Science

in

University Honors

and

Computer Science

Thesis Adviser

Bruce Irvin

Portland State University

2023

Abstract

This essay looks to reflect back upon the successes and failures of the EPL Card Reader capstone project, sponsored by Edward Ivory, head of Portland State University's Electronics Prototyping Lab. The EPL Card Reader's goal is to provide a means of tracking and updating student activity and training on the various machines in the lab. Using a local computer port to host this web app a lab administrator or manager is able to scan a student's access badge to review which machines they have been trained on as well as update that training status. The app also has a running file that logs when a user scans into the lab, providing valuable data for Mr. Ivory in the form of user metrics. All of this was achieved using the Flask framework which is a python library suite which enables database integration. I am able to look back at this work from the unique perspective of the project's Team Lead whose tasks consist of keeping in contact with the sponsor, the capstone adviser Bruce Irvin, keeping my team up to date with the project as a whole as well as with one another on their individual work. A key lesson I took from this project was the department's undervaluing, if not outright fear, of the concept of partner programming as a powerful learning tool.

Keywords: Framework, project management, database integration, capstone, pair programming, review

One of the most interesting proponents of this project is that it consists of mostly inexperienced members, including myself, the team lead. Most of the leads had little to no experience in project management, a fact that Bruce Irvin, capstone instructor, was quick to point out. Rather than looking for experience, Dr. Irvin emphasized a particular shared trait: “conscientiousness.” Armed with little more than the drive to do our research into the gritty details and complete tasks we marched on towards our first bi-weekly meetings into what it was to be a team lead.

A fear I had going into these initial meetings was my own lack of knowledge in particular sectors of the programming world. I had somehow made it through my degree without taking a proper full stack web development course. I was most comfortable with C++, the language at the heart of PSU’s upper division, competent in JavaScript and little else. One of our first meetings would embrace this heartily. We were told in no uncertain terms that some of us would work on projects that were unlike anything we had ever touched. Entire teams may end up with projects that few, if any members at all, would have the skillset to work on. We as team leads would have no experience in what it was to run a full team of developers and were fledgeling developers ourselves. Yet in the face of all this abject terror, there was some comfort to be found in its acknowledgement. To have forewarning into inevitable failures; to embrace them and learn. Now all we had to do was select our crew of ragtag developers to bring on this journey with us.

Developer interviews consisted of groups of three interviewees and three interviewers in-person, with Zoom as a last resort for those that needed it. Our capstone

group was so large that we had to split interviews in half and record one group's interviews for the other's to watch later. We would end up missing about half of one side's interviews with this system. I found myself looking for two things above all others: experience and competence. A handful of student developers were already working in industry and this I earmarked as top contenders. Next I'd list the most confident individuals that emphasized communication with their team, prioritizing students I recognized from previous classes with positive work ethic track records. When all was said and done, I was thrilled to have a team made up of individuals that were all in my top 10 candidates. Next came getting to know each other and researching which of the sponsored projects we'd prefer to work on as a team.

I assigned each member to research two projects and to leave notes for peer review in a github repository, a tool used to manage various versions of a software project to get people comfortable with using git in a team environment. We had a rating system to share how you felt about a specific project and listed them in order from high to low. It was during this research project that the department's structure had left a bit of a gap: web research. It would be foolish to say that most, if not all computer science (CS) students hadn't used the internet for something, whether it be as benign as verifying a specific program syntax or as nefarious as seeking answers to a homework problem. That being said, I am walking out of the department with some deeply ingrained fear of the web as it pertains to solving programming problems with the belief that answers provided online from the likes of Stack Exchange are unresearched, flawed, not to be trusted. But then how does one find answers to questions like "how do we build a program from scratch to manage a database of students, with local web

hosting integration, that also supports infrared card reader technology through a USB port?” Specific questions like this are hard to find answers to and certainly no class I’ve ever taken has provided tools for finding the answers. Initial web searches yielded plenty of potential frameworks to get started, but many were missing some key piece of the puzzle, such as USB integration or a way to communicate with the USB port. On a whim I dared that most dreaded of technology: ChatGPT. It was able to immediately suggest Flask, a Python framework (mind, a language I’ve no experience with) that supported direct database and object integration, the tools and support libraries to supplement it such as SQLAlchemy, and even a specified port library in pyserial to deal with the card reader technology. This exciting new technology which was explicitly forbidden in the past was able to provide me with answers in seconds that I was struggling to find in hours of research. While there are obvious scholarly ramifications of ChatGPT being unleashed upon the academic world, current industries are taking advantage of these tools and features. Jobs we’re likely to be applying for are fully supporting its integration. Why, then, are we not taught how to properly use such technology in a righteous and scholarly way? This is a thought I’d have at the back of my mind for the 6 months of capstone work to come.

After receiving our assignment of the EPL Card Reader project we were quick to get to work. After some debate of trying a new up-and-coming language, Rust, we eventually settled on the Flask Python framework to work in a language more people were familiar with, as well as being a more supported and simplified language with the knowledge that electronics students would likely be taking over maintenance in the future. After meeting with Mr. Ivory, we learned that the original hardware we were

expecting was now unavailable, and a simple alternative was being created in its place. Our team moved forward by designing the application to work with data input rather than with functional card reading until we'd receive the functioning hardware. We took the first several weeks to practice with this framework, each member learning a specific portion of the tool such as webpage integration, the serial reading library or the database initialization.

It didn't take long for cracks to start to appear with this more laid back "professional" environment simulation. Some members were working full time in industry as well as attending school full time; one lived outside the state, another worked overnight shifts, one worked mid shifts and would be hard pressed to make every meeting and another still was three or more hour's drive away from campus, making it difficult to get the card reading hardware we'd need to experiment with later. Simply put: this was still a class which would come second to people's personal work and family lives. It was my job to make sure everyone was still making a fair contribution to the project while also ensuring they were getting as much out of the capstone as possible. It was around this time that Mikayla, a fellow honor's student and teammate, mentioned pair or partner programming, wherein one person programs while the other watches, something easily done over the web with tools like Discord or Zoom. Unfortunately with such a scattering of time slot availability as a team we weren't able to coordinate this idea until the latter weeks of the second semester.

As members of the team became more comfortable with their respective assignments we began integrating our work together. We'd start with small simple examples such as creating a simple "student" concept in the database to demonstrate

we could enter information into the webpage and have it display out to the app as expected. Our workflow would be checking in every Monday to demo our respective progress and how our functionality worked. As we had the rare opportunity to create an application from scratch, communication with the sponsor was crucial. More straightforward clarifications such as differentiating an admin, manager and a student seemed obvious, but some things were taken for granted. For example, it hadn't occurred to me to ask what operating system we'd be launching from; as it turns out, the Flask framework will send multiple requests to the USB port at once. This didn't matter to Linux, which will simply go through each process one at a time. Mac and Windows, however, don't like multiple access requests to the same port and would crash or freeze the app. This led me to getting in contact with our Computer Action Team (CAT) to ask about installing a virtual machine on the EPL's kiosk to run the app, and clarify if the sponsor was okay with this route. A lesson in the importance of getting early stage demos into the hands of the sponsor, as Dr. Irvin would emphasize. If it wasn't for a demo with the sponsor and their Mac, this may have been a painful last minute report to write.

As we approached the end of the semester (we're still actively working on this project as I type this) we were able to have several instances of peer programming, as Mikayla was able to demonstrate her industry skills as she'd work through features and debugging or altering code while I or another would watch and help catch logic bugs or typos. After only several minutes I was blown away by the obvious benefits of this paired programming approach. Not only does it help the person writing the code keep track of things in their head to make sure it matches what's coming from their fingers,

but it provides the ability for the watcher to ask questions about a functionality or method of a function and confirm their own understanding of what's happening on their screen. When swapping places the viewer is able to solidify that understanding as they work through the code and have their teammate coach them through this function or that. This was a stark contrast to everything I've been told the previous two, three years: "don't share code." While I understand needing to learn the concepts yourself, are we not adults in higher education to be trusted? Some of my most impactful learning moments came from working through code with a classmate through the years, resulting in both them and me profusely commenting in our code "worked together with [...]" as though failing to do so would end in disciplinary action for plagiarism. If there was one thing I could redo for this capstone to the benefit of the team and the resulting product, it would be enforcing a partner programming approach to some degree. I can not overstate the value of this methodology enough, and can only hope the upper division program can embrace it in one way or another.

In these final weeks we're tying everything together, going back through our project board to make sure we haven't overlooked any core feature requests. I was surprised to learn that not everybody was testing their functionality in the same way, another process I had taken for granted. Some people were testing their functionalities in small self-contained modules, rather than running the app itself and testing within the app. This far in we had people that had never actually run the application, though their work was still able to be integrated and executed as expected. This situation, I believe, would have been made more clear sooner if I was able to implement pair programming! But I digress. We have our sponsor testing these final iterations for any last minute

changes, tweaks or missed functionalities and will be demoing the application with Dr. Irvin in the coming weeks before our final project presentation and product delivery to the sponsor. I believe keeping an open line of communication with the sponsor was crucial and can not stress enough my agreement that getting your product in the hands of a sponsor as soon as possible should be a top priority. The more realistic emulation of a work environment only confirmed my suspicions that being able to work side-by-side with someone else is an invaluable addition to the programming experience. If the goal is to prepare us for industry, why not offer us more realistic experiences? The capstone project itself is an excellent addition to my education and I'd love to see more group programming in the upper division.

References

Liechti, Chris (2020) pySerial (Version 2.7) [Python Library]

https://github.com/pyserial/pyserial/

Lord, David (2023) Flask-SQLAlchemy (Version 3.0.3) [SQLAlchemy Extension)

https://github.com/pallets-eco/flask-sqlalchemy/

Ronacher, Armin (2023) Flask (Version 2.3.2) [Web Framework]

https://github.com/pallets/flask