**Alex Peterson – Z1912480 – 5/1/23**

**CSCI 390 Internship Experience Essay**

My internship with ALDI and the Customer Interaction team was not my first foray into the corporate world, but it was my first ever internship- let alone an IT internship. I have worked in a variety of different capacities in my life: retail, mechanical repair, warehousing, trading cards, woodworking, territory management. I have worn many different hats, but my internship with ALDI was the first to truly give me an appreciation for the amount of time, energy, and organization it takes to successfully deploy and support new technology product.

During the summer of 2022, a fellow student relayed his frustrations to me with interviewing for software development positions within companies, whether they be dedicated software companies or an IT branch of a business. He lamented to me of his perceived catch-22 of the university computer science student. He said to me, “I will be graduating in the fall, and I am interviewing for software development roles. Hiring managers turn me down and tell me they want a candidate with more experience, but the only way to get that experience is to get hired for a development role.” With this roadblock in both of our minds, we set out to find a way to escape this vicious cycle of lack of experience: get hired for internships. An internship would give us a safe and nurturing environment to get our feet wet in the corporate world, let us explore different areas of IT and development, and give us a chance to gain valuable experience that would be put to use for future interviews. I wasn’t concerned with whether I would snag a dedicated software development internship, or simply one related to IT. My main criteria would be that I could learn more about a company’s tech stack, how they handle IT, how their development cycle worked, and insert myself into meetings with developers to learn more about their work.

NIU held a job fair in September in the Convocation Center. I met with ALDI recruiters on campus and had some great conversations with their human resources representatives, some of which even seemed to have a technical background, which made the fact-finding process much more fruitful. My experience with prior job fairs and meetings with recruiters had been fairly lackluster overall. ALDI’s recruitment team were friendly, kind, and interested in hearing about my story, experience, and what I wanted to get out of an internship.

I began my internship the week of Thanksgiving 2022. I cannot speak on behalf of all interns, but I would exercise caution with Fall/Winter internships. I cannot speak truth for every industry, but for an international grocery business like ALDI, Fall/Winter is a fairly dead time in terms of projects as well as it is peak time for vacation and key people being out of office. I say exercise caution only in a partial sense, as I self-identify with a classic case of “American Capitalist Work Ethic Syndrome”. When I am new, things are slow, and there isn’t an abundance of deliverables to work on, I start to get anxious that I am not performing in the eyes of management. Despite my anxieties, this slow period was a blessing in disguise, as it gave me more time to get trained and onboarded without the distractions of project deadlines.

I joined ALDI’s Customer Interaction (CI) team within ALDI National IT (NIT) at their campus in Aurora, IL. As a brief aside: my manager, Irene Shiu, is on my short list of leaders that I feel genuinely appreciate and care about their personnel. Irene is tenacious, optimistic, and seeks to elevate others to the potential that she sees in them. From a 30,000 foot view, the NIT CI team is responsible for providing support, creating the “how to” knowledge base for said support, help manage relationships within ALDI and external IT vendors, as well as be liaisons for IT projects and ALDI business counterparts (marketing, buying, etc). As of 2020, ALDI has been working on a project called ACI: Accelerated Customer Interaction.

ACI is made up of a massive collection of different features, technology, and products. In short, ACI’s goal is threefold:

* Upgrade ALDI’s online grocery presence from a static catalog site to a full-fledged, transactional, e-commerce platform.
* Make the website available on desktop, mobile, and a dedicated mobile app.
* Offer pickup and delivery services and reduce the reliance on external partners like Instacart for support.

As a project, ACI still is hard for me to wrap my mind around. It is a project with active development and a massive scope. My role on the Customer Interaction team in relation to supporting ACI comprised of a few things: regression and end to end UAT testing, providing support to the business counterparts who are also responsible for ACI’s marketing, and attending daily/weekly standups with domestic and international teams within ALDI regarding ACI’s status.

Before going into detail on how I spent the bulk of my time supporting ACI, I also had the opportunity and freedom to branch out and explore different areas of the company to learn more about how ALDI works. I spent time with the Systems team who are responsible for managing servers for ALDI corporate and their grocery stores. ALDI’s Systems team uses various virtual machine software to manage their servers nationally as well as manage the vast array of databases that hold ALDI data. I also had the opportunity to embed myself with the two main development teams within ALDI- National Development and ADS. I have spent the most time with ADS and had the privilege to participate in their daily standup activities to discuss their project status, participate in code reviews, discuss story grooming, and participate in writing both sequence and activity diagrams.

My time with National Development consisted of learning about several different in-house developed projects for applications that are used internally by the business. By far the most interesting project, and the one I spent the most time learning about, was the TOT program. TOT is an application that is used for planning trucking delivery routes. TOT takes into account a list of stores in a given region that need replenishment, the nearest warehouses, the contents of pallets a store may need, the different trucking companies that ALDI partners with, and the truck type needed (non-perishables, dry goods, freezer). TOT takes all of this information, and by using a radar-like algorithm similar to one developed by the US Navy, it maps out the most efficient trucking routes- prioritizing sending the fewest number of trucks to as many stores as possible. For instance, 3 stores are within a 100-mile radius of a given warehouse. Given that each store’s order and buying plan is already accounted for by the forecasting and buying teams, TOT just needs to know how many pallets and of what type need to get to each store. After entering store locations, quantity of pallets, and type of product being shipped into the delivery compilation functions, TOT will generate routes for the most efficient one-way trip to accomplish the delivery for all three stores using a single truck. TOT is a standalone web-app with an HTML and Node JS front-end and using Microsoft C# for the backend logic. It interfaces with ALDI’s forecasting and buying software API’s to receive information on the store orders and their pallets. Despite being a fairly green and aspiring software developer, I was really impressed at how easily understandable the code for the project was. Given that TOT was already fully deployed and in use, I didn’t have much insight into National Development’s deployment model, but I was made aware that they used an agile method with two week sprints.

As I mentioned before, the codebase for TOT was easy to follow and incredibly well documented and avoided obtuse naming conventions. Coming into a professional environment, my expectations were that the code I’d have the privilege to see would be complex, unwieldly, and arcane. Much to my surprise, this was quite the opposite case with TOT. I spent time with Josh, one of the developers on the NatDev team, to talk through TOT and its future improvements. I was excited that I was able to follow the entire full-stack logic of the project without getting lost. Being my first experience with a professional development team, it was exciting to see a project written with both performance and readability in mind for future supporters of the application. Within my experience with the university curriculum, there have been plenty of times that given how fast web-based tech stacks are changing, I have felt unprepared to work around professional development. I was happy and relieved to see that the focus on fundamentals/CS foundational principles at NIU armed me with enough skills to hold my own when discussing ALDI’s applications and be able to ask great questions to learn more.

I had no prior C# experience before, but I did have a semester’s worth of experience from taking Dr. Freedman’s Java development course last Fall. The project made great use of many of the principles that I learned early on in the computer science curriculum: encapsulation, polymorphism, and abstraction. Similar to Java, with C# being so class and object oriented, the TOT code made great use of dependency inversion and dependency injection. TOT abstracts out the different components of a delivery (location, route, truck, contents) into different individual components/classes. After the algorithm compiled the information, it would create an instance of a Route object to be consumed as the final product.

The ADS development team consisted of about five developers and their manager. Considering my inexperience with professional software design, I wasn’t sure what to expect in terms of team size. For a large, global company like ALDI, I assumed groups would be larger than or bigger in scale. In hindsight, despite the importance and potential sales draw of the SNAP module project, it was entirely a backend project that was a plugin to an existing system, so it’s not outlandish that it would have a relatively small team composition.

My time with ADS was primarily spent embedded with the developers who were working on SNAP payment module for ACI. The SNAP module is a piece of middleware to allow customers on the new.aldi.us site to pay with their government issued SNAP / EBT benefits card rather than with a credit card. SNAP was (and is still) a very exciting project to participate it given how much is at stake for ALDI- millions of sales dollars are currently being missed given that ALDI currently doesn’t support this payment on the ACI site. While I haven’t had the pleasure to directly develop, I have had the privilege to work with two of the developers on code review, planning, and diagramming. ALDI’s ADS team uses an Agile model and works within two week sprints on their features. Participating in daily standup meetings has been an enlightening, albeit stressful, window into professional development and the pace of play within Agile- things are moving fast and we are ready to pivot when need be.

The SNAP module has been the most interesting piece of development I got to interact with at ALDI. Unlike most of the programming projects I have done within NIU that were self-contained programs or full-stack projects, the SNAP module is a plugin that interfaces with the ACI site and the government’s SNAP/EBT benefits payment system. Given that SNAP is entirely a plugin to an existing web-app, ACI, SNAP consists of backend logic and is written entirely in PHP. This was a huge boon for me- I had had plenty of PHP experience in CSCI 466 and 467 in writing the backend for two different web applications. Although they were using some PHP features which I was unfamiliar with, along with much of the application abstracted out into many different classes, my experience from my NIU coursework allowed me to follow along during code review with (relative) ease. The SNAP module, as well as the bulk of the ordering logic for ACI, is built on top of the Spryker Commerce OS Platform. Spryker sells itself as an operating system, and while this is *technically* true, it behaves more as a sandbox for you to develop your e-commerce store within. Similar to how a game engine like Unreal or Unity offers developers tools and plugins to create the logic for their game while the studio creates its own elements in Blender or Maya, Spryker consists of packages and modules for you to arrange your business data and handle orders, while allowing you to create separate modules like this SNAP module to interface over different layers via API calls. In short, since the ALDI payment processor, Fiserv, doesn’t handle SNAP/EBT transactions, and there are thousands of ALDI customers who already use their SNAP benefits in store, this necessitated the creation of the SNAP module for ACI orders. The SNAP developers also used GitLab, much to my chagrin. I was excited to see them using this as their primary version control software, as I had experience learning how to use Git and GitHub within NIU since Prof. Winan’s made it mandatory for software submission for CSCI 340.

To contrast my time with my own group projects at NIU, the group of developers within ADS worked remarkably well together while working in a primarily siloed basis. Each developer would own a piece or feature from within Jira for that sprint and come into each daily standup with their current status on the feature, the amount of time they felt they would need to finish, any concerns or questions they wished to raise, and any calls for help as needed. I was impressed at the ability of the developers to have the agreed upon vision to be able to work separately but adhere to the group’s design. I believe a large part of this was due to the extensive documentation and diagrams that they would create and host on Atlassian’s Confluence software. ALDI uses Confluence as a knowledge repository and documentation source for all of its IT projects. Having a single repository of information like Confluence was incredibly useful given how large of a company ALDI is, in addition to having developers in the US, Europe, and Asia working across different time-zones be able to have a shared reference that could be maintained by all.

In my experience with group programming projects with at NIU, the groups would typically not have a shared vision, design documentation reference, or a clear understanding of the division of roles. Generally speaking, I was typically the most outspoken and outgoing member of the group, so generally I had to come up with the design vision while also wearing the “scrum master” hat. ALDI’s ADS developers didn’t seem to have these issues and showed interest in helping one another make progress and meet the deadlines; there was a sincere camaraderie I had not seen before working on software. I was surprised at ALDI’s interpretation of Agile and how streamlined it was. I spent time learning about different development methods in CSCI 467, the bulk of it being on waterfall and agile. In my time in that course as well as in my personal research, I have spent time learning about agile methods and terminology. There are a lot of different ways to work in an “agile” fashion, and strictly in my opinion, many of these tactics appear dubious at best in terms of increasing focus and improving the outcome of the project. ALDI’s approach to agile development is very different from my learnings in CSCI 467- by barely implementing agile in the first place. Make no mistake, this is not criticism. In my research on how other companies implement agile, time can be often wasted in meetings and kanban board planning before even writing a single line of code. It is more efficient and “agile” to have individuals create working code and deliver that iteration to the business for feedback, rather than meticulously plan ahead of time to write it one time. Software is not like hardware or infrastructure, every component is malleable and you can make sweeping changes in only a few days, sometimes just a few hours. ALDI may be one of my only exposures to agile development, but the minimal implementation reads as highly efficient.

Additionally, similar to my experience with my manager for NIT CI, the ADS development manager, Adam Suhayda, displayed an eager interest to help the team with any issues at hand. In standup meetings where the topic related to upper management raising concerns about progress on SNAP or changing requirements, he frequently voiced intent to defend his team and speak up for them upon their behalf. This management style of going to bat for your team and leading from the front was refreshing and a departure from what I have experienced in past companies.

Moving away from spending time with the ADS development team on SNAP, as mentioned above, my assigned role was with Customer Interaction team working on the Accelerated Customer Interaction project. Similar to my surprise with the ADS development team size, and considering how large ACI is in scope, the NIT CI team is also small in size, eight individuals including our manager and myself. My first few months of onboarding at ALDI were spent learning the different applications that make up the ACI project as well as the software to support it. The following is by no means an exhaustive list, but here are the applications I spent the most time using: ServiceNow, Adobe Analytics, New Relic, Jira, Confluence, and Medalia.

ServiceNow (SNOW) is the incident and request ticketing system that ALDI uses globally. As an outsider, I take tech support or issue resolution for granted- I now know it is no simple feat to implement both the process and procedure for fast resolution in a large company. ServiceNow consists of a front end interface for any user to create a ticket containing their ALDI issues. From a feature on the website not working appropriately, to needing a new computer monitor in the office, it all runs through SNOW. ALDI has a team called the Global Service Desk (GSD) who act as level one support and field every single ticket. GSD is a massive body of labor and are the first responders to every single incident or request made in SNOW. Part of my time spent with the NIT CI team using SNOW was to monitor our ticketing queue and offer solutions to ACI issues, as well as be responsible for escalating incidents up to the appropriate fixers. My technical background within NIU helped greatly here. There were many time that I would become the liaison for an issue and leverage my soft skills to be able to relay technical issues to non-technical business users. ServiceNow is a behemoth of an application and has experts within ALDI that are simply responsible for maintaining the process and logic to keep the app running. After all, how do you provide support if your support app isn’t working?

In terms of monitoring and analytics software, I had the privilege to learn and use Adobe Analytics and New Relic. The most interesting part of learning how these different entities within ALDI would use them in such different ways. A team like marketing, for instance, would use Adobe Analytics to make business decisions based on customer journey trends, search query information, frequent purchases, and using tracking pixels to other sites. Since I am within NITCI and not concerned with business trends, I could look at Adobe Analytics to spot user engagement and dropout rates, combine that with site health monitoring data from New Relic, such as 400 and 500 errors from that day, and combine that data to find potential issues on within the site. Where Adobe Analytics is more marketing and user behavior focused, New Relic provides site health and traffic monitoring. I was most excited by New Relic and the insights it offered on user dropout rates as well as the aforementioned 400/500 error reporting. In a given day, if the customer service teem received feedback from users about payment issues, I could deep dive into New Relic and take a look at the relevant payment pages on the app. If we were showing a spike in 500 errors, we could potentially have a serverside issue within ALDI or a payment processor. If there was a small spike in 400 errors, it could simply just mean we had a large number of users with expired card information or bad credentials.

My enthusiasm for using these analytics tools was also made manifest in my internship capstone project that I had the privilege to present to a (virtual) room of ALDI directors, managers, and a VP. Similar to this essay, I collected my experience into a slide presentation along with things I felt the company could improve upon. As much as I enjoyed being a part of the UAT testing team, many of the test cases are either redundant or repeats of tests from a prior week. Ideally if a feature received a bugfix, and you are writing software in using SOLID principles, it shouldn’t require retests of other currently working features. My personal peeves aside, within my presentation, I suggested to ALDI that test automation software like Selenium and Cucumber could be useful. While it would front-load more labor onto the development teams to use and implement these testing suites, this could save substantial amounts of time (money) on redundant UAT testing for future features, and would help limit the manual testing to only the key features that may need a more human touch, like QA exploratory testing. I took the time to learn how to program using the Selenium web driver to create test automation. I created a test case using IntelliJ and the Chrome WebDriver to show to the audience an automation case involving a user logging in using their Google social account and how the automated case would track each step and could account for things like clicking, hovering, typing input, and navigating to new windows or external websites from new.aldi.us. In addition to time savings test automation could introduce, I encouraged the ALDI developers to work with the teams responsible for monitoring New Relic and Adobe Analytics. I spent time using this software and pointed out that the insights we can see within these tools could help us identify typical (and atypical) user behavior and write more comprehensive test automation scripts.

Jira and Confluence are the Atlassian ­­­products used by ALDI for their software issue tracking (not to be confused with incident tracking within ServiceNow) and their knowledge repository (the other being knowledge base articles that live in SNOW regarding SNOW processes). I was especially excited to work in Jira and Confluence, as they are extremely popular products across companies that use any kind of software. My time in Jira was mostly using it for bug tracking as well as sprint and testing management. Part of my role in participating in the 2-week sprint process was within a team of UAT testers testing the QA instance of the new website. I would perform exploratory testing on my own by intentionally trying to break and abuse the site, inspecting around the site source code for any vulnerability that might be user facing, as well as meticulously following user journeys to simulate how a typical customer would interface with the application. I was happy to have experience with web-based applications prior to ALDI, as my perspective in writing them was useful to have the insights and the eye for weaknesses and being able to speculate how a certain bug could be manifesting before having to raise the issue to the developers. Any issues I would find within the new.aldi.us site or any undesired behavior I would raise with a bug ticket and work with developers to get resolved, ideally in time for the next sprint in two weeks. Sometimes these bugs wouldn’t get resolved, or hotfixes would introduce new problems. My experience working within a QA team showed me how important and useful a team of inquisitive QA testers is to an IT project. The delta between a bad tester and a QA team member who is invested in the project and knows the history of past issues is massive. If I were a full-time developer within ALDI and the quality team came to me saying “all green”, I would be very concerned. One doesn’t simply write software that is bug-free. Valuable QA testers understand the nature of the failure points within the application and can offer nuanced feedback on a sprint. Offering feedback of “this is broken” is useless, but it’s far more productive to be able to articulate “case A works, case B works, case C is inconsistent, and I can introduce a case D that you didn’t account for in your journey that causes undefined behavior!”

ALDI’s use of Jira, Confluence, and their other development focused applications were interesting to me as a new intern. I had assumed that all development and design was in house, either within the US or Europe, primarily Germany (ALDI home base). However, I was surprised to learn about how many different systems and development was handled by external entities. ALDI uses several different external development teams as well as a few external teams to provide consumer-facing customer service. With regards to Jira and Confluence, all the external entities and users interface with ALDI through these apps. Jira and Confluence are used as a town square of sorts for anyone and everyone who is touching an ALDI application. This is not only important in terms of external developers, but also other businesses that ALDI partners with. While ALDI is working on their own pickup and delivery solutions, they partner with Instacart and recently DoorDash to offer customers more choices on how to shop. Just these two delivery partners introduce a massive amount of extra complexity into the ACI project. Managing the relationship between these two entities and ALDI is heavily leveraged by using Jira. One solution that ALDI is bringing to their stores is their own proprietary pickup application to be used by store associates. Instead of using a smartphone and the Instacart or DoorDash apps, ALDI will be rolling out a combination scanner and smartphone device with its own application for in-store pickup. The picking application software project is a subset of ACI and result in the ability to keep more customers within the ALDI ecosystem, rather than handing them off to a delivery company to handle that transaction. I personally spent time working with the picking app on UAT testing as well as physically testing it within a store in a normal use case.

Despite this internship not being solely about development, learning how a large company like ALDI manages their software projects and IT infrastructure gave me valuable knowledge that I couldn’t have learned within the NIU CS curriculum. This experience improved my development skills by retraining my brain to think more holistically about the entire lifecycle of software. Writing a code and deploying it is one thing, but creating something that is sustainable with a plan for future support is an entirely different venture.

Getting immersed in this more supportive role provided me a lot more insight into working in a professional setting on a software project. Within my role on the NIT CI team supporting ACI, my team filled an incredibly important role that I often neglected to consider within a global software project. Given that the focus of my NIU CS education is about development and writing software, I had not paid much mind to the other roles that would make up a software project other than just the developers who write code. This may be obvious to some, I was surprised to find out that there is simply not enough bandwidth for business counterparts (marketing, buying, et cetera) and development to interface and coordinate on projects. Another party is needed to liaison between the business’s use case and demand requirements, the developers changing implementations, while also being a force to provide ongoing support and maintain the knowledge base for an application. I had experience in the corporate world prior to ALDI, but my experience was mostly alone and not working within groups. My experience providing support for and working on ACI has shown me that without the ability to communicate across channels, across time-zones, and across teams, you could have the most talented developers in the world and still to the table with an unfinished and broken product. A cohesive collection of different teams who can stay in their lane, while also cross-collaborating, will surely improve the odds of delivering a quality product.

(note to self: go look at confluence and jira and grab some more information about SNAP that is dev/code related. Talk about gitlab, php ide, cool PHP features, etc. Talk about the diagram showing the module and its layers , aci -> spryker-> govt)