**A Fintech Experience as a Software Developer Co-op at CIBC**

Henry So

Science 3391

Technical Report

Faculty of Science, Department of Computer Science, Western University

April 17 2024

## Table of Contents

[Table of Contents 2](#_h42seh966fy2)

[Overview 3](#_8bhi62x20p6x)

[**A Fintech Experience as a Software Developer Co-op at CIBC 4**](#_dzeezb9ez9op)

[New Beginnings 4](#_pwmbwsacbfk3)

[Future Career Goals and Necessary Professionalism Caveats Set Prior 4](#_rcu67zz6rq1h)

[Day-to-day Duties, Projects / Tasks I was involved with 5](#_ww1fz2i74nal)

[Working with Difficult People 6](#_z6q5o8y9muc2)

[Engineering Culture 8](#_3yyy9hd9fdjo)

[Mentorship, Overall Pace 11](#_x6d8rrr84vbh)

[My Future Career Ambitions 12](#_f2yko94i0lk0)

[Personal Recommendations and Final Remarks 12](#_932p544uy0yu)

[Gallery 15](#_jgazlrbdjp27)

### 

### 

### **Overview**

This internship report outlines a comprehensive reflection of my role as a Software Developer Co-op at a Big Five bank in Canada. The report goes over the goals I had prior to my first day, day-to-day duties, project contributions, knowledge and skills that I gained, as well as my overall personal experience, my retrospective perspective, and recommendations derived from my 8 month internship experience at CIBC (Canadian Imperial Bank of Commerce).

### 

### 

### 

### 

### 

### 

# A Fintech Experience as a Software Developer Co-op at CIBC

## New Beginnings

This was my first ever technical job, and I was glad to be given the opportunity to showcase my skills and prove my value as a competent developer / engineer. On my first day enroute to the office, I found it pretty intimidating navigating my way around downtown, but once I found my way to the 50-floor blue-tinted glass skyscraper at 81 Bay Street, I felt a profound feeling when I realised that I would be working here on the 22nd floor for the next 8 months. I was both excited and nervous to meet my new peers, co-workers, my manager, but nonetheless ready to face the challenges.

## Future Career Goals and Necessary Professionalism Caveats Set Prior

One of my career goals is to work for the largest tech companies (think Meta, Google, Apple, Airbnb, Uber, Stripe, Databricks) in the biggest tech hubs in the world, namely New York City, Toronto, Seattle, and/or San Francisco. Ideally, I’d like to become the most complete software engineer that I could be to maximize my chances at landing that big-time position. I believe that everyone has limitless potential, so I wanted to realize my own potential and leverage my internship opportunity to create a solid foundation for my future. Some of the caveats that I set for myself prior to the start of my internship were to hold myself to the highest standard of integrity, approach things with an open-mind, ask good questions strictly on the topics I had trouble with, and ask for help when I would get blocked (give it at least three tries first!). But also be a team-player and help others within my team. Show up everyday ready to work, while also maintaining a positive, patient, humble attitude. Be professional, show initiative and give employers reasons why I should be considered for a full-time role through my work.

## Day-to-day Duties, Projects / Tasks I was involved with

As a Software Developer Co-op at CIBC, I was fortunate to be working in a hybrid role, being in the office every Thursday, and working from home the other four days. As a developer, I would be responsible for handling stories assigned to me on Jira, a task management platform. Stories are essentially issues, fixes, or new product features to be added to the platform. At the beginning of my term, it took me several weeks to onboard, which included; getting access to the codebase, Confluence (the documentation hub), and Jira. Afterwards, I was assigned to some minor issues to ramp up my Salesforce knowledge and to familiarise myself with the large codebase. I would be coding in Apex, a programming language based on Java. In November, there was an opportunity to work on a new project to get rid of null interfaces and improve error / exception handling. I volunteered myself to work on this project with the goal to centralize and provide methods that made logging exceptions easier, such as to uphold a standard, to maximize code re-use and rid of unwanted values that were in the Exception Log. It would also assist the Application Health Check team to obtain more accurate statistics on particular errors. I worked on the technical design and implementation of this project. In my final weeks, I took on some issues to assist a tech lead of a pod where the primary focus on the pod was to take on a wide scope of technical issues on the platform; essentially helping where help is needed. Most recently, I worked on a bunch of test classes that failed when there was an attempt to deploy to a test environment. So I made sure that those tests would pass with my changes.

## Working with Difficult People

In terms of working in group projects, I’ve had minimal negative experiences working with peopIe. The only negative experiences I’ve had were joining groups blindly in hackathons and group projects where the professors assigned the group. But in most of the Computer Science group projects I was a part of, I was able to work with my friends who also were as academically-focused and career ambitious as I was, which is something I love; I especially love working with like-minded people who have the lets-get-it-done-well mindset and don’t excuse poor implementations. I had a poor experience with a manager that was overseeing the Exception Logging framework project that I was working on alongside another co-op. I felt that under her management, the project team stretched a project that should have been a couple months into a six month project. Additionally, I felt that my opinions were looked down upon due to my position as an Intern/Co-op.

At the beginning of the Exception Logging project, I proposed a simple solution that would meet the requirements of the problem. However, the Sr. Manager was not receptive to the solution, wanting a more dynamic, robust, scalable solution (mapping based on exception description). I didn’t want to protest, so I said sure, worked on the solution with another co-op and brought back a complex solution that would have some limitations (mapping only allows for 1:1; cannot have multiple interfaces based on description without it being extremely complicated, and there was another issue that I can’t recall) with the approach. I let her know about these limitations, and she asked my seniors / mentors if the solution was fine. They had their own tasks that they were busy with so they just went with whatever she said with minimal protest. Throughout the development of this new project, she would change the requirements biweekly, going back on what she said in previous meetings whenever I presented the solution. This is a little frustrating, but it’s fine, new projects always need to be refined. Looking back, one of the things I should have done was to have meeting notes and document her requests to avoid any miscommunication, and to refer to it if she wanted things to be done differently than previously asked.

Fast forward a couple months, it is late February, and the solution is being pushed into the codebase for a feature release (monthly code improvements to the platform). After all tech leads look at it, they all approve except one. The tech lead brings up the same limitations of the solution that I let the manager know beforehand, and proposes a redesign since it was super complicated (I agreed!). She then proposes the same solution that I proposed initially with the string appending instead of the current implementation, only this time the manager wholeheartedly agrees without disagreement. I was just glad that someone else brought this up, but was slightly annoyed that my voice wasn’t heard when I was the one to say it. I quickly implemented the changes, tested them, and finished my task, moving away from the project afterwards, and now, spending my last weeks at CIBC working on tasks assigned from the tech lead. I felt that I had wasted a lot of months that I could have spent working on other tasks, learning other things, specifically more Salesforce specific tasks, as the exception logging project was more of a back-end engineering task, since it assisted with the processing of data associated with exceptions. This experience dissuaded me from considering working with Salesforce as a long-term gig.

I did bring this up to my hiring manager, another Sr. Manager about my want for additional involvement to learn and dive into something else as to increase my exposure, and I was assigned a topic to learn more about and explore, but I would have liked to have been assigned tasks on Jira (task management tool that lots of companies use). I would have stayed working on the Exception Logging project as well as handle additional tasks. However, he maintained that my primary focus was the Exception Logging project. My seniors did try to advocate for me to move away from the project, but he was firm on his stance. I believe in learning through a focused, open-ended discovery through projects/tasks. By the end of the term, I had only learned close to a handful of things involving Salesforce. I would have liked to received more meaningful work and tasks that challenged me to learn new things, as it would have incentivized me to learn more about Salesforce on my own time, since I always want to hold a standard of excellence in terms of my work. I know that I am capable of much more, but I felt as if I was a low-level grunt assigned to do all the work that no one else wanted to do. I wanted more of a challenge instead of a stroll in the park.

I realize that there is a risk involved with giving a co-op/intern more challenging tasks, especially if it has a high impact on the end-user / application, as well as giving a co-op too heavy of a workload, but I was open to doing more things. Additionally, I would have liked to gain some respect from my peers that I was capable of doing things. I felt that my situation was handled poorly, and although I received a return/extension offer for the summer, I ended up declining in lieu of this.

## Engineering Culture

In my personal opinion, there is somewhat of a weak engineering culture at CIBC, or at least within my team. Specifically, the mindset. CIBC employs modern industry practices with Agile software development. Some examples include: code is pushed out in sprints monthly, recurring standup meetings to assist in unblocking issues and communicating progress, pods are structured to tackle technical development with an area of focus on a subproduct of the entire platform / product, code reviews, multiple test environments, etc. Don’t get me wrong, there are numerous talented individuals working on my team that deserve the utmost respect due to their contributions, but I’ve found that the primary focus when it comes to software development is to pump out work and push it to production; as long as it works. I’m not a fan of this approach because if things are done poorly initially, there may be bugs and issues that come back to bite because the quality control / check was not good, which will add onto technical debt and stagnate the progress of new features due to issues in production. Do not get me wrong, bugs are bound to happen, they are inevitable. But quality correlates with the amount of bugs that will come back. Personally, I love seeing well-written code and elegant solutions. However, I found that the codebase was pretty ugly. Code quality affects the codebase’s future and some of the code being pushed into the codebase sometimes was not consistent in their styling (indentation, spacing, variable naming, casing). Working in a clean codebase is more ideal, especially since many hands will eventually make their rounds to handle the code. I found on numerous occasions that tech-leads would just approve code with bad styling and would not comment on it. I understand that there may be a lot of things on their plate that rigorous reviews can’t be done, but I believe that things should at least be pointed out, at least to maintain the integrity of the codebase.

I also found that some developers don’t care about small things that may affect the amount of technical debt that their code was causing. The developer team during our weekly meeting stated that there was a new focus on reducing technical debt. I had a conversation with a developer on our Exception Logging Framework team regarding his code where I told him that we should not be referencing literal strings but rather creating a variable and then referencing that variable instead, as literal strings create overhead on the heap. This would optimise his code and reduce tech debt in the future, even if by a small amount. There are millions of exceptions that occur yearly, so it was shocking to me when the developer was not receptive to my opinions. There is something called the string pool in Java, where all literal strings are stored. I assume Apex has something similar, but essentially, the JVM checks for the string in the pool and if the string literal is already present, it just returns that reference to the pooled instance; otherwise, if it is a net new value, a string is created into the pool. My thinking was that if the new framework were to be utilised in all of the distributed systems and code, since he was referencing literal strings in the processing of exceptions, that would mean that every exception that uses that method would create n\*overhead (where the overhead is the processing in which we check n literal strings in the string pool to verify that there isn’t a new one) on the heap. In the future, this would have an effect on the heap if it started to get large / full. A full heap may cause garbage collection (a process to free up memory not in use) to run more frequently. Additionally, memory allocation. When you allocate memory (for example, when you create a new object), the runtime needs to find a block of free memory on the heap large enough to hold the object. If the heap is nearly full, it may take longer to find a suitable block of memory. If the application uses up all of the available memory, the operating system may swap memory to the disk, which is much slower than RAM. There would also be additional processing done to check if the string exists in the string pool. These are all things that will cause an application’s ability to be performant.

## Mentorship, Overall Pace

Upon my first day, I was greeted by two new people, Deep and Shawn who I would be working closely with for the next 8 months. They were in charge of mentoring the co-ops, and they did a very good job in my opinion. I came in with close to little experience working with a large codebase and pushing all my code to the main branch to writing cleaner code, thinking more critically with problem solving, using Git (developer tool) more soundly and in general improving my communication, whether its explaining my code changes, being professional in written communication, or relaying the issues that I was blocked on. One key thing I picked up was asking good questions. My process would be work on task > come across blocking issue > try to search for a solution online, look at supporting documentation > ask mentors if no solution found and explain steps I’ve already tried and the scope of the issue > waiting game for an answer. I needed to minimize the amount of questions asked as they had tasks of their own to do, so I used this sort of workflow to ask more critical questions. They were extremely patient and helped me understand code, problems when I probed them with questions. They were also very easy to work with, and it was comfortable that there was never an atmosphere where I felt pressured. However, at times I did find that working at a bank was a little slow. Sometimes I would have some downtime, either because I finished my tasks relatively quickly, or if I was waiting on a meeting to relay some information and or I was waiting on someone else. I would spend this downtime learning and research more about topics related to the work I was assigned so that I could not only do tasks properly but do them according to industry standard. I would have liked a faster paced environment to really push me to get out of my comfort zone and learn a lot more things. It would have incentivized me to learn outside of work.

## My Future Career Ambitions

As I stated earlier, I’d like to be able to be in a position where I could work out of one of the US tech hubs. To maximize my chances of landing such an opportunity, I plan on working on my personal projects that include a personal website, as well as a side project that assists small business owners manage their EMT transfers. I hope that some recruiters will take note of both my skillset and passion and offer me the opportunity to interview with the hiring teams. Additionally, I am currently looking for internship opportunities in the Summer, Fall, Winter, and next Summer. Although I am doing full-time school come Fall, I anticipate that my course load will be easier and want to manage a job at the same time so that I could gain experience before applying to new-grad full-time jobs. I also plan on applying to full-time jobs as I study. However, I would be grateful to land any type of opportunity, as long as the job position revolved around being a Software Engineer or Software Developer. Ultimately, my dream is to work as a Software Engineer, perhaps in the field of Web Development, Back-end, or Machine Learning. Hopefully, I’ll be able to land a gig after graduation.

## Personal Recommendations and Final Remarks

If I had to give my honest opinion about whether or not CIBC was a good place to work as a Co-op / Intern, I would say that yes, it was a great place to work. Mostly due to work-life balance, and opportunity to work for a Big Five bank in Canada. I thought that CIBC did a fine job in terms of hosting events for their student employees. There were a wide range of fun events available to the Co-ops, from resume building workshops, to connecting with VPs in senior management, as well as learning workshops. I had the opportunity to also connect with other co-ops during these events, as they were structured so that there was enough time for connecting at the end. I met a lot of friendly faces and people and built a small network of people at CIBC. In terms of events with the ECRM team, there would be socials every now and then with complimentary food, and drinks. That was a fun way to unwind with the team. On some Thursdays, I would join them at Scotland Yard Pub, about a five minute walk from 81 Bay St, where there was a Happy Hour for half off poutines. The events really exemplified a sense of community within CIBC.

In terms of the work environment, the office building was brand new and super convenient to travel to as there were GO station terminals connected to the actual office building. If you travelled by TTC, the Union Station was only a 3-minute walk away, and you could reach the office via the Path. I also liked how my manager and co-workers worked. There wasn’t a constant micro-managing, but instead, their instruction were more of a guide, and the way you navigated tasks and duties was up to your free will. However, this is team-dependent, so my experience may not reflect other business departments / units in the bank. There was a cafeteria in every subsystem (every 3 floors is a subsystem / working space dedicated to specific teams), and there were relatively cheap food options in the fridge that you could purchase. Additionally, there were high-quality coffee machines and complimentary tea. Within the office working space, there were meeting rooms where you could congregate with your co-workers, focus rooms for you to pump out some works or have meetings on Teams, and dedicated modern work areas. There were also meditation rooms, lactation rooms for nursing mothers. I like how CIBC puts their employees first. I also liked how there were whiteboards so I could track my process of thought when working on tasks, and each work area was equipped with two 24’ monitors / ultra-wide monitor, a standing desk, an ergonomic chair, and laptop docks / charging stations and outlets. It allowed me to feel that I could really make use of the space to accomplish my work whenever I was in office. Also, although there was a dress code, the team was relatively open-ended in their clothing choice. As long as it wasn’t loud and was more smart-casual attire, then it would be fine. However, the majority of the managers did dress a little more professional. All in all, I enjoyed my time at CIBC, but I just wish I had the opportunity to do more. It’s a fine place to be, regardless if you are a co-op or full-time employee. If work-life balance is a big thing for you, then CIBC is a strong recommend.

## Gallery

|  |
| --- |
|  |
| Candid photo taken at the office by another Co-op. |

|  |
| --- |
|  |
| Candid photo taken at the office by another Co-op. |

|  |
| --- |
|  |
| Making new friends with Adil, a Co-op on the Operations team! |

|  |
| --- |
|  |
| With my fellow Software Developer Co-op Mohit! |

|  |
| --- |
|  |
| With ECRM Director Ioana! 🙂 |

|  |
| --- |
|  |
| With my manager, Sumeet! 🙂 |

|  |
| --- |
|  |
| With one of ECRM Sr. Manager, Marinela 🙂 |

|  |
| --- |
|  |
| The cafeteria on the 21st floor. |

|  |
| --- |
|  |
| Just one of many meeting rooms. |

|  |
| --- |
|  |
| My WFH Setup. |

|  |
| --- |
|  |
| A gorgeous view of the sun setting over the Downtown Toronto skyline. |

|  |
| --- |
|  |
| The CIBC Square Lobby from the 1st Floor |