CMU MSE Application Essays

Part One: A Challenge on a Software Development Team

Practice Area: Quality Management

Spark Change LLC was seven people when I joined. Within my first month, I was sent on-site to our only profitable client — a health system running 12 hospitals, $7.3B in annual revenue — to modify claim generation logic. The work involved a custom "roll-up" script that bundled line items before submission to insurance. Get it wrong, even slightly, and claim generation halted across every hospital. Worse, a field with bad data wouldn't fail loudly; insurance would just start denying claims quietly, sometimes for days, before anyone noticed the cash gap.

There were no automated tests. We manually tested in a cert environment, walked through user validation, and shipped it anyway. That process failed more than once — outages that were embarrassing for the team and genuinely damaging to the client. When the COO pulled me aside and said "figure it out, that's what we pay you for," I didn't have a safe way to do that. My boss was stretched thin running the company. I was on my own.

The real issue wasn't my skill level or even the complexity of the billing logic. It was that we had no reliable way to know whether a change was correct before it hit production. Manual testing can't cover years of accumulated edge cases in claim data. We were flying blind every time.

So I built a test framework. It pulled edge cases from real claims, stored them alongside their expected outputs, and for any new change it ran the logic against that set and diffed the results. I also ran it across the last 14 days of production claims before every deployment to catch anything lurking in the data I hadn't thought to test for explicitly. It wasn't elegant — but it worked. Development cycles dropped from hours to under 30 minutes, and we stopped breaking things.

What I'd change is *when* we did this. Building quality infrastructure after you've already burned the client twice isn't a strategy, it's damage control. That framework should have existed before the first production deployment. The lesson I took forward: automated verification isn't optional in systems where bad output has a direct financial consequence. It should have been the starting point, not the response to failure.

Part Two: Long-Term Professional Goals

I want to build and lead the infrastructure that lets large engineering organizations ship software continuously without breaking things. Right now I run the build and release team at Cboe Global Markets — four people supporting thousands of daily commits, nightly deployments to thousands of servers, and exchanges that process close to $100 billion in trades every day. The goal I'm working toward is getting that cadence from nightly to hourly while keeping stability. That's partly a technical problem, but it's also an organizational one. It means convincing stakeholders, coordinating across teams, and making the case in terms that business leadership responds to. I'm better at the first part than the second.

That's specifically why CMU's MSE program appeals to me. The Engineering Leadership specialization sits at the intersection I'm trying to grow into — technically rigorous but also focused on the kind of communication and cross-functional leadership that I haven't had much formal exposure to. My background is mechanical engineering. I've filled the gaps through seven years of hands-on work, but there are real holes in my fundamentals. A colleague recently traced a network outage in minutes by reading routing tables and syn-ack logs. I was still running iperf3 in a loop. I got there eventually, but that gap — between finding an answer and finding it efficiently — is exactly what I'm trying to close.

Two specific areas I'm hoping to dig into: cybersecurity and data visualization. The systems I manage push code to production exchange infrastructure, which makes them an obvious target. I want to apply what I learn to tighten our permissioning models. On the visualization side, our deployment dashboards are functional but not great at surfacing real-time status clearly. That's a fixable problem.

The MSE program gives me a path to do the work I'm already doing at a bigger scale and with less stumbling. That's what I'm after.