SAM PHIPPS
Software Engineer
(320) 428–1559
samuel.v.phipps@gmail.com
https://github.com/Samuelvphipps

In your opinion what is ‘good code’?

Good code has two main properties. First, it almost goes without saying, but the code should ‘work.’ By this I mean it completes its purpose in the larger program. There are infinite ways to do this, but the most important part is the second part of good code: the code should be readable/maintainable.

Whether it is me, as the author, going back through the code, or another developer going through the code, it should be understandable. This sometimes means not writing it in the most ‘efficient’ way using the ‘fanciest’ techniques, but writing it in a concise yet understandable way where the code stands by itself. This is vital to the long-term use of the code as it will need maintenance, likely by another engineer with a different style and set of experiences.

In my career I have always built systems following the KISS method, and code should follow this as well. When I supervised military operations as a platoon leader, I did not focus on more than what was needed to successfully accomplish a mission. While designing a vehicle recovery system process this took the form of whittling down the communications to the basic information needed to make a maintenance decision prior to deploying the vehicles on a mission. We, as a team, developed and replaced what was a formerly long radio call with a set of 6-8 questions that condensed all the needed information into a 30 second conversation.

What do you look for in your ideal team in terms of practices, processes, and behaviors?

A good team is based on trust. Everything else grows from that.

Teams should practice openness. Each person has a completely different set of strengths and weaknesses, whether that is the senior developer, or the new bootcamp graduate. No one should be afraid to share those strengths with the team.

Teams should also continuously reflect on their work. A lot of times this means admitting mistakes to each other with no judgement, because the mistakes are where the improvement and learning happens.

Finally, teams should respect one another no matter who they are or where they come from. This emerges from a team where each person can be their authentic self, no matter what that means, without judgement or shame.

When I was in the Army, I worked on a team where the leaders trusted each person completely. We were tasked with closing a Military base in southern Afghanistan on a two year timeline while partnering with the Afghan National Army. Our trust in each other allowed each of us to bring our expertise to the team no matter what our rank or position. In this case, I was the lowest ranking officer. The team, however, knew that I was the contract expert, from auditing for the U.S. Government, to the day to day needs and planning the with Head Contractor. The team we created allowed me, as the lowest ranking leader in the room, to supervise the planning and communication alongside the civilian department supervisors. The partnerships I formed with the contractors, both at the leadership level and the individual team member level, enabled us to plan a flexible base closure plan. Do to the team's collective work, and the individual expertise we each contributed, we successfully accelerated the base closure plan and closed it over a year early, something contracting command told us was impossible.

What attributes do you think an ideal software developer demonstrates?

I think successful software developers demonstrate a willingness to learn from other people, no matter how experienced they are. Everyone has something to teach, and something to learn, whether they have been developing for 20 years or one.

A software developer should also be patient and have the ability to take a step back. It is easy to get sucked into a problem, and not stop and reflect on the approach you are taking. The ability to step back, rethink your approach, and then start the problem from a new direction allows you to see a new solution instead of fighting the original assumptions. I have begun setting timers to force reflection when I am tackling a particularly tough problem to help remember this fact.

Finally, an ideal developer is someone who can communicate. When working on complex technical problems it can be easy to talk past each other and not truly hear what one another is saying. Communicating is a skill, especially technical communication.

What are your thoughts and experiences with TDD?

I have been using Feature Driven Development throughout my time developing software, from scoping to deployment. My main experience with TDD was this week. I built my first C# .NET server this week while working on a group project with prewritten tests. During this process we learned a valuable lesson: that tests should be run concurrently with the development. We made a mistake writing the server without running any client side tests and instead we simply tested the endpoints in postman. This led to test failures, and a rewrite of ¾ of the endpoints.

This experience showed me the power of TDD. It is a way to ensure that as you build, you build in the correct data and correct data flow. This prevents bugs, streamlines the development process, and provides feedback along with valuable information regarding the issue. I am excited to learn more about testing and writing tests for code as I progress through my career, and the above lesson is a fantastic starting point in this journey.

What do you believe is the difference in behaviors and skills between a Senior Software Engineer and an Intermediate Software Engineer?

Both must know how to program effectively, including how to write working and readable/maintainable code. What sets the two roles apart is the Senior Engineer’s ability to manage the big picture, and coach/mentor the team. Each person may be the most talented individually, but without direction, coaching, and vision, the ability of the team is weakened. This is where a senior developer matters most.

In my time in the military and while teaching construction I have led teams, and we had the most success when each level of leadership mentored and coached, while also providing accountability. For example, I once taught a new crew leader how to intervene in a conflict between members of his crew. We sat down, I walked him through some strategies, then reflected with him after each of his mediation attempts. This is the senior engineer’s job: Manage, coach, lead and represent the team of skilled intermediate and beginning developers to ensure the team is successful.

What are your salary expectations?

Negotiable.