Team 2 Global Ethical Activity

In this paper we will be discussing three different ethical dilemmas that affect software developers as individuals and companies as a whole and how it relates to, and effects the software we are developing today. Our software is an all in one collection management solution targeted at collectable trading card enthusiast that want a better way to sort and organize their trading card collections. The three we will be discussing are how much to give back to open source, how free does content really want to be, and To bug-fix or not to bug-fix. We feel each of these dilemmas has a potential to be impactful in our decision making during our development and deployment of this product.

From author Peter Wayner’s article he describes 12 ethical issues related to software development. One of the issues we will address in his article is his *dilemma No.9:* *How much to give back to open source.* Our application, that stores information about collectable trading cards, will use a variety of open-source programs and potentially code to help aid our project. Some of these open-source programs include but are not limited to: PHPMyAdmin to create and run our database, as well as XAMPP in order to launch access to the database. According to Wayner’s article “Everyone knows that open source is free… but not everyone contemplates the ethical issues that come with using that free code.” In short, we all can use the open-source programs to our own good, but not many of us think about giving back to the people who provided us their code they work vigilantly on for us. Mark Radcliffe in “*Top 10 FOSS legal developments in 2019* “ describes 10 issues with various companies and people in using open-source programs. One such issue describes how a key Linux kernel developer, Christoph Hellwig, sued VMware in the district court of Hamburg, Germany on the grounds that “VMware had violated the terms of the GPL2 by combining VMware’s proprietary code, called “vmkernel” with Linux that created a derivative work, and [they] did not [provide] the complete corresponding source code of vmkernel under GPLv2.” This issues Radcliffe describes shows us just how serious not abiding by open-source guidelines can really get, especially with big companies such as VMware. For our own project we want to make sure that all 3rd party software and code is fully credited and that is acknowledged following that company’s discretion. To do this, our website will have a page dedicated to following the licenses of the 3rd party software and giving credit to all those software we decide to use.

Another dilemma discussed that could have an impact our software that we would like to discuss is number 3: “How free does content really want to be?“ We believe this could have a lasting impact on the long term sustainability of the software, due to the upkeep and development cost associated with it. The original idea for this project came from our own needs to organize a collection of trading cards. The intention for the final result of our product would be for it to be openly available to anyone that would like to use it free of charge. But when allocating the time and resources to develop this software, we wanted to consider the potential marketability of it as well. And if this was in a theoretical business environment, we want to be able to maintain our product and pay our developers.

In an article titled “Nothing good is free: How Linux and open source companies make money” They discuss how large companies that produce free and open sourced software manage to not go under, while still providing their services. One of the points made is that when your software is competing on a small market with a more specialized needs, such with trading card inventory management, your potential profit and market opportunity goes down. However, this could also potentially mean the opposite, when you consider that the amount of competing software’s might be few and far in between. Allowing you to capitalize on what market share is currently available. This is a delicate balance and can be risky when considering your potential return on investment. Then there is still the subject of how to begin accumulating income to begin with. The article lightly discusses the community editions economic model for a software where you have free editions of your software that is available to everyone, but there are paid alternatives that offer additional features and support. Our team could even take this to another level by offering product services to customers that want to customize the software to better meet their needs. Despite the risks, there is still plenty of potential to maintain a profitable organization through distributing free software.

The fifth ethical dilemma “To bug-fix or not to bug-fix?” could affect our program in several ways. First of all, let’s say our program is due soon but we have still a few bugs. If we cannot fix all of them in time, we would have a dilemma over which ones will affect us the most and try to fix them. This could also be something like small that we do not think we will need to fix and might go under the radar because of that. If this project ended being released to the public at some point, then this dilemma would be a much bigger issue for us. Because the users of our program would likely be varied a good amount since there are many different kinds of cards and there might be both paid users and free users. For example, there could be an issue where we might have to prioritize between fixing various bugs some of which affected paid users and some which affected free users. If we focus on fixing the bugs that are affecting free users first our paid users might get upset and we might lose subscribers. But if we fix the bugs affecting the paid users first, we could lose free users that might have eventually paid for our product.

There are several ways we could attempt to address this issue when creating our program. First of all, whenever we first find an issue, we should immediately write it down and try to fix it, if we cannot do it, we should ask the others in the group to see if they can help us. This is to make sure we do not end up getting to a point where we either do not have enough time to fully fix everything or let any slip through the cracks. Another way we could try to minimize the number of bugs would be to heavily test our code before each “release”. Even though the processes described before would likely help limit the number of bugs that appear in our program, there is still a chance that we might still have to pick between two bugs that are affecting different groups. I think the best way to deal with a problem like this is to take into account how many people it affects, but more importantly how big of an issue it is. If there is a small graphical issue that’s affecting a lot of people, it should be prioritized less than if there is an issue where a smaller group of people are not able to access the application at all.

We hope the information provided here gives you a general idea for the considerations we are taking during our software development process. It is hard to account for all potential issues and decisions that will encounter as we create and deploy our software, but with the research we have done here, we should be able to have a slightly better idea for how to tackle problems as they arise.

Work Cited

Radcliffe, M. (2020, February 13). *Top 10 open source legal issues and developments in 2019: Synopsys*. Software Integrity Blog. Retrieved October 3, 2021, from https://www.synopsys.com/blogs/software-security/top-10-open-source-legal-issues-2019/.

Wayner, P. (2014, April 21). *12 ethical dilemmas gnawing at developers Today*. InfoWorld. Retrieved October 3, 2021, from https://www.infoworld.com/article/2607452/12-ethical-dilemmas-gnawing-at-developers-today.html.

Gewirtz, D. (2016, November 18). *Nothing good is free: How linux and open source companies make money*. ZDNet. Retrieved October 1, 2021, from https://www.zdnet.com/article/how-do-linux-and-open-source-companies-make-money-from-free-software/.