**When to Stop Coding**

Chris Shaw, Sean Tronsen, Dago Aguilera

Department of Computing Sciences, Coastal Carolina University

CSCI 490: Software Engineering II

Dr. Ross Foultz

October 3, 2021

**When to Stop Coding**

Creating a music queue, like any application intended for public use, can have a huge ethical impact on our society. This impact depends heavily on the ethical decisions that software engineers make when designing their software. It is a common misconception that developers are only responsible for is generating technical solutions for a given project. Being said, an overlooked power of software engineers is the ability to make a direct impact on the world, whether it be positive or negative (Alcanja, 2019). This is determined by how a software team chooses to address the various dilemmas that surround a project. In this writing, we will address three different dilemmas that we face in designing our music queue application. The first dilemma we will address is how “bullet-proof” our code should be. We must also take into consideration the measures that we our willing to go to protect the users of our application. Finally, we discuss the notion of prioritizing bugs found in our program based on how it impacts the user. While there are many other dilemmas that occur when designing software, this only scratches the surface of what developers face that exists beyond simply writing code.

The first dilemma to be discussed in this writing is how bullet proof should code be (Wayer, 2014). As mentioned in Chris’ VoiceThread discussion, all developers should strive to design the best solution possible for their software (C. Shaw, personal communication, September, 2021). In order to achieve this, it requires that everyone put great effort into testing their portion of the application. In a real-life setting, the amount of dedication to testing is usually predetermined by the budget and time a potential client is willing to allow for a complete project. In this case our team will act as the client has proposed a reasonable budget and timeline for the best possible technical solution to address their problem. By testing and attempting to break our code we will save ourselves time and energy, thus creating a positive ethical impact on our users.

A dilemma mentioned by fellow classmate, Timothy Kelly, that is worth addressing is “How much Protection is enough,” (Wayer, 2014). In his initial post, Timothy placed an important emphasis on the vulnerability of customers when providing personal data such as credit card information and social security number, especially on an E-commerce application (T. Kelly, personal communication, September 2021). Though our team’s application does not intend to require such information, Timothy’s discussion inspired me to reflect heavily on the measures my team should take to protect the users of our music queue application. Our proposed solution for addressing this dilemma is to focus our application strictly on the statistics that our website provides for artists. Although our application may require a user login as well as personal information such as name, location and musical interests, we plan to only use this information to store likes and dislikes for that user. When it comes to an artist profile, we must consider a different approach. Their information, such as their name (or band) and location will be on display for the public to see for other users to vote on the performance. One proposal to be discussed within our team is to delete artist information from our database withing a certain time period of performing.

The last dilemma that we will expand on in this paper is whether certain bugs should be fixed. Classmate, Brian Taylor expanded on how it is nearly impossible to fix everything wrong with an application. Instead, it is suggested that we prioritize bugs based on how it affects the user. Like Brian’s approach, our plan to address this dilemma is to immediately fix any obvious bugs that we find in our respective portions of the application. It should be noted that applications typically are deployed with unknown bugs as well. Although this is undesirable, there is no malicious intent when this is done, and it remains a simple fact for any application that evolves over time. Our application will be no different and in the perpetual cycle of maintenance/updates, old bugs will continually be removed just as new ones will be inserted into the code. In not enforcing absolute perfection and allowing some bugs to slip through the cracks, our goal is to eliminate any system critical bugs prior to deployments. Our team will be following the practice of “Testing in Production”, which has become a common approach in the present day for handling this dilemma. To provide an explanation, an application will be initially developed, then tested to an acceptable level of functionality. From there, it is deployed to the end users who also function as the testing team for the application (Majors, 2019). These users will go on to report bugs over time that the development team will handle as a part of its maintenance workflow. In other words, the practice involves using them as a tool to locate bugs in deployed applications, thus pulling them further into the software development lifecycle (Majors, 2019). When this practice is followed, the reported items usually consist of minor interface-related bugs that can easily be patched out by the development team. Following this practice will allow us, to focus on future updates and features that makes our application unique rather than being bogged down by the concept of perfection.

In creating any public facing application involving user data, there is a certain level of responsibility that must be recognized by the developers and software engineers involved. The moment an application is deployed, a series of ethical dilemmas arise, making them an important component to planning during development (Alcanja, 2019). Issues can arise related to the standard of perfection the codebase adhered to, the policies used to protect information, and even the number of imperfections a system is allowed to possess. Planning for these beforehand directly plays into the ways that systems are designed, hence the reason that software engineers and developers have a noticeable influence on the world. In creating our Music Queue application, we the development team, aspire to influence the world for the better with the functionality we will go on to create and provide.

References

Alcanja, D. (2019, October 9). *Why we need to address ethical issues in software engineering*. Simple Programmer. Retrieved October 4, 2021, from https://simpleprogrammer.com/ethical-issues-software-engineering/.

Kelly, T (2021, September). Personal communication. Retrieved October 3, 2021 from

https://coastal.voicethread.com/lti-student/1008237/gallery/18480052/32264225

Majors, C. (2019, June 5). *Test in production?* . TechBeacon. Retrieved October 4, 2021, from https://techbeacon.com/app-dev-testing/test-production-yes-you-can-you-should.

Shaw, C. (2021, September). Personal communication. Retrieved October 3, 2021 from

https://coastal.voicethread.com/lti-student/1008237/?tok=449413601615a0d5604d4f7.11531739

Taylor, B (2021, September). Personal communication. Retrieved October 3, 2021 from

https://coastal.voicethread.com/lti-student/1008237/gallery/18487235/32409019

Wayer, P. (2014, April 21). *12 ethical dilemmas gnawing at developers today.* Retreived October 3, 2012, from https://www.infoworld.com/article/2607452/12-ethical-dilemmas-gnawing-at-developers-today.html