# Problem Set:

How much time do you spend every day on social media sites like Instagram, Vine, or Facebook? Probably more than you think! One of the requirements of this course is for you to spend a little of your social media time enhancing your career by keeping up on new technologies and trends.

Once each month (a total of six times across the semester), you must identify and share an article on something relevant to the course; e.g., Agile Methods, Scrum, Extreme Programming, Continuous Integration, or DevOps. You may choose any relevant article from any relevant site, including:

- http://dzone.com (Links to an external site.) (Links to an external site.)
- https://www.oreilly.com/topics/software-engineering (Links to an external site.) (Links to an external site.)
- http://www.martinfowler.com/ (Links to an external site.) (Links to an external site.)
- http://scrum.org (Links to an external site.) (Links to an external site.)
- https://devops.com/ (Links to an external site.) (Links to an external site.)
- https://www.scrum.org/resources/blog/ (Links to an external site.) (Links to an external site.)

Include the following in the document of your submission:

1. URL: The URL of the article
2. Description: A one-paragraph synopsis of the article, including conclusions. You must describe the article in your own words. Don't cut and paste from the article.
3. Recommendation: Do you recommend this article? Why?

# Solution:

URL - https://dzone.com/articles/the-stigma-around-refactoring

Description – David Bernstein is the author of this piece. It is mostly concerned with the refactoring process. The author talked about why refactoring is so crucial in software development and how to make it more effective and less time-consuming. Refactoring is a genetic phrase, and everybody working in the software industry should be aware of its importance. Refactoring involves more than just rewriting the code; it should function in the same manner it did before, but with less complexity. In terms of performance, the restructuring should produce predictable results. Refactoring in small pieces, according to the author, is the key to achieving it securely. Instead of considering everything at once, take small steps towards each foul odor it produces. Considering everything at once would detract from the refactoring's core goal. Furthermore, testing after each development guarantees that the result remains consistent. It helps us keep on track by motivating us. When an author's code fails to compile, he feels unsafe, according to the author. That is why we must always check minor parts to determine if they are functioning properly. He also stated that new technological tools machines would aid us in working in this manner. Many codes are written incorrectly. We can refactor them in most cases. However, there is a stigma in the software business about legacy code, which is code that was written many years ago (and it is still working). However, the way it was written makes it difficult or impossible to add a new feature. Dealing with this type of code is difficult because the original creator is sometimes unhelpful or unable to provide solutions. Instead of fixing bugs, refactoring legacy code usually introduces new ones. According to the author, we need to redesign the entire system to solve this problem. We need to make room for a new function to be added to the system. Refactoring the present system is one way to accomplish this. The unit tests are one thing that legacy code lacks; nonetheless, automated refactoring can be used to safely refactor legacy code. Refactoring is an extremely effective method for transforming an existing system into a new one. It functions as a link between two different design codes. It improves the code's effectiveness, readability, and comprehension. By learning a simple technique for reworking legacy code, the author hopes to dispel the stigma associated with it. He believes it will alleviate our apprehension about refactoring legacy code.

Recommendation – The importance of refactoring in the software development process is demonstrated in the preceding article. It also concentrated on legacy code, which is a difficult refactoring application. As a result, I strongly suggest reading this article.