Ever since the release of ChatGPT, Large Language Models (LLM’s) have been quite popular in usage due to their numerous helpful capabilities. These capabilities include answering questions with more direct details, ensuring written content is grammatically correct, and generating software code for programming assistance. Because LLM’s have the ability to generate software code, this has struck fear into many software engineers as they may potentially be replaced. Analysis of software unit tests generated by LLM’s is crucial to determine if a LLM’s software unit tests are good enough to replace manually written software unit tests of a software engineer as well as to see what issues and benefits arise if they are to be chosen. Faults in software unit tests can affect a piece of software's quality, reliability, and security.

Given that the output of LLM’s are not 100\% accurate, thorough evaluation of their software unit tests is critical to ensure they are valid, effective, and readable. This in turn will help verify whether software quality, reliability, and security standards remain high if software unit tests written by LLM's are to be used and if software engineers jobs are affected. This paper provides evaluation metrics for software unit tests written by LLM's. I propose an evaluation framework that dictates whether software unit tests written by LLM's are valid, effective, and readable enough to replace unit tests written by software engineers into six metrics: build \& success rate, unit test failure explanation \& evaluation, line coverage, mutation score, readability of code comments, and readability of the code itself. Build \& Success rate measures whether the unit tests are able to be compiled and run successfully without human intervention. Unit test failure explanation \& evaluation measures whether unit test failures are due to the LLM creating a faulty unit test or actually identifying a bug in the input program. Line coverage measures how much of the input program is being executed when unit tests are being executed. Mutation score measures how many faults a unit test can detect if the faults are purposely inserted into the input program. Readability of code comments measures how easy and effective they are for other software engineers to understand the purpose and functionality of the code. Readability of code measures how easily a software engineer can understand and interact with the code through maintenance and modification. The organization of the rest of the paper is as follows: Chapter 2 discusses related work and other research papers used for the motivation of this study. Chapter 3 discusses my proposed evaluation framework to determine if unit tests written by LLM's are valid, effective, and readable. Chapter 4 discusses how the unit tests written by LLM's are evaluated with experiments. Chapter 5 presents the results of my experiments. Chapter 6 discusses and analyzes the results of my experiments. Finally, Chapter 7 provides a conclusion of my study as well as future research directions.

Problem is if we don’t analyze it, could take away jobs plus affect software in terms of quality, reliability, and security.