I. Research Background

The rapid growth of backend technology has been inseparable from various challenges in the last ten years [1]. The increasing demand and complexity of features in backend applications cause development to cut back quality, especially maintainability aspects during the development process [2]. It is indicated by a huge number of code duplications in the codebase [3]. This issue is because developers must read the entire code and must consider the changes that do not impact the existing flow which will result in prolonging the feature development process.

The previous problem can be solved by applying a concept called clean architecture [4] that helps increase the maintainability of the application because a system can have lower complexity compared to a system that does not adopt this architecture. In addition, it can also reduce the responsibilities of each function, decrease maintenance costs, and minimize the possibility of errors in the system. The problem of low maintainability and huge numbers of code duplication can be controlled by adopting the principle of clean architecture proposed by Uncle Bob because each layer is not responsible for the layer outside it [5]. The code duplication issue occurs often, and it leads to expensive costs for the maintainability of the codebase if the workflow needs to be modified [6].

This research proposes procedures on how to implement clean architecture principle on the backend application to improve the maintainability of the application, especially when additional features are added. The scope of this research is to apply clean architecture to improve any inefficient code writing such as code duplication or unnecessary code problems, by following our proposed systematic refactoring methods [7]. To validate our methods, we use several codes metrics related to maintainability, such as cyclomatic complexity, weighted methods score, Halstead's score, Kan's defect, and maintainability index. We measured these metric results from the codebase that has been refactored using our proposed refactoring methods that follow clean architecture principle and from the original codebase without refactoring.