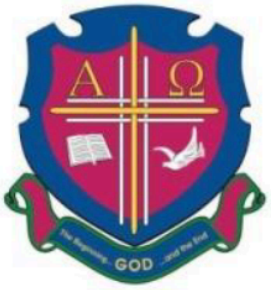

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

FACULTY OF ENGINEERING DESIGN AND TECHNOLOGY

NAME: ISAIAH MUKISA

REG NO: S21B23/007

ACCESS NO: A94160

COURSE: BACHELOR OF SCIENCE IN COMPUTER SCIENCE (BSCS)

COURSE UNIT: SOFTWARE CONSTRUCTION

1. Single Responsibility Principle (SRP) Violation:
- The “Report” class violates SRP as it is responsible for both generating a report and writing specific reports for different roles.

2. Open/Closed Principle (OCP) Violation:
- The “Report” and “BonusCalculator” classes violate OCP because they need modification whenever a new role is added.

3. Liskov Substitution Principle (LSP) Violation:
- The “Manager” and “Developer” classes, both inheriting from “Employee”, may not adhere to LSP as they add specific methods (`calculate_manager_bonus`, `manage_team`, `calculate_developer_bonus`, `code_review`) not present in the base class.

4. Dependency Inversion Principle (DIP) Violation:
- The “Report” and “BonusCalculator” classes directly depend on the “Manager” and “Developer” classes there by violating DIP.

Refactoring Plan:

1. For Single Responsibility Principle Violation:
- Create a separate class for report generation.
- Each role should have its own class responsible for writing specific reports.

2. For Open/Closed Principle Violation:
- Introduce a strategy pattern for report generation and bonus calculation.
- Each role's report and bonus calculation logic should be encapsulated in its own class.

3. Liskov Substitution Principle Violation:
- Ensure that all subclasses adhere to the same interface by introducing abstract methods in the base class.

4.Interface Segregation Principle Violation:
- Break down the “employee” class into smaller interfaces for specific roles.

5. Dependency Inversion Principle Violation:
- Use dependency injection to decouple the “Report” and “BonusCalculator” classes from specific implementations.