Software Design Patterns

Luke Foley  
T00224345

Table of Contents

[Introduction 3](#_Toc182430253)

[Existing System Overview 3](#_Toc182430254)

[Design and Architectural Pattern Exploration 3](#_Toc182430255)

[Design and Architectural Pattern Implementation 3](#_Toc182430256)

[Testing and Validation 3](#_Toc182430257)

# Introduction

# Existing System Overview

The ToolSYS application is a management system designed to streamline operations for a tool rental business. The application was developed during the second year "Requirements Engineering" and "Software Engineering Project" modules and was built using Visual Studio with Windows Forms for a straightforward, user-friendly solution.

The system is organised into five main sections: Rates, Tools, Customers, Rentals, and Admin.

* Rates Module – This module allows users to define categories of tools and their associated rental rates, ensuring that rental fees are calculated consistently.
* Tools Module – This module manages all aspects of tool management, including adding new tools, updating their details, removing unavailable tools, and viewing the complete inventory.
* Customers Module – This section focuses on customer information, enabling users to add, update, and view customer details.
* Rentals Module – This module facilitates the tool rental process, allowing users to rent tools, return them, and view rental records.
* A screenshot of a computer

  Description automatically generatedAdmin Module – This section provides analytical features, such as annual revenue reporting and tool type usage analysis, offering valuable insights into business performance.

A diagram of tools

Description automatically generated

While the original system fulfilled its primary purpose, it was developed with a focus on functionality rather than architectural principles or scalability. This made it a suitable choice for this assignment, providing opportunities to implement design and architectural patterns

## Main problems

Limited Scalability: Adding new features, such as customer categorization or dynamic discounts, require extensive modifications to the existing codebase. This inflexibility made it difficult to adapt the system to evolving requirements

Violation of SRP: Several classes handled multiple concerns. Forms managed UI interactions, executed business logic, and communicated directly with the database. This mix of responsibilities made the code challenging to debug and reuse.

Tightly Coupled: The program has little separation. The UI, business logic, and database are very closely linked, with business logic in the UI

# Design and Architectural Pattern Exploration

# Design and Architectural Pattern Implementation

# Testing and Validation