MMCM Computer Laboratory Manager

A Hands-on Project in IT102 Object-Oriented Programming

In Partial Fulfillment of:

BS in Computer Science

Term 1, School Year 2024-2025

Date of Submission:

November 6, 2024

Course Name:

IT102 – Object Oriented Programming

Team Members:

Aque, Kit Janbren

Magtibay, Zuriel

Yleaña, John Cyrus

Submitted to:

Clyde Balaman | ccrbalaman@mcm.edu.ph

Table of Contents

1. Introduction

1.1 Needs

This application addresses the school computer laboratory's need for an efficient time recording software. Our application provides a secure and centralized system to track student logins, monitor usage duration, and manage access control, ensuring that each session is accurately recorded. By automating login validation, session tracking, and account management, this solution not only enhances the security of lab resources but also allows administrators to streamline oversight of computer usage.

1.2 Benefits

For **professors**, this application brings improved oversight and ease of access, allowing them to reserve computer lab time, view student usage, and promote responsible digital conduct within the lab. Professors can monitor student engagement in real-time, enabling them to ensure students are focused on academic tasks and to intervene if misuse occurs.

For **students,** the system offers a seamless and fair experience, with clear time-tracking that ensures equitable access to lab computers, particularly during peak usage times. Students can view their own login history, fostering accountability and time management, while benefiting from an efficient login/logout process.

1.3 Solution

The **solution** provided by this application is a comprehensive computer lab management system designed to efficiently monitor and manage student activity within the school computer lab. It addresses key issues by allowing students to securely log in using their student ID and password, tracking their usage time and automatically recording their session duration. Professors and administrators can monitor active sessions from a central dashboard on the server, where they can see which computers are in use and view individual session details, such as login history and usage hours, for each student.

The system also includes features to improve security and accountability, such as account locking after multiple failed login attempts, and administrative controls for unlocking accounts and creating new ones directly in the application. Additionally, it provides a clear log-out mechanism to ensure that student sessions end properly, updating both the client-side and server records.

1.4 Differentiation

This application distinguishes itself by providing the school's first structured system for managing computer lab usage and tracking student activity in real-time. Unlike existing solutions**—which are largely manual or non-existent**—this application automates critical aspects of lab management, such as time-tracking, login authentication, session monitoring, and account security.

2. Objectives

2.1 General Objective

The general objective of this project is to develop a comprehensive computer laboratory management system that streamlines the monitoring, usage, and security of computer resources in a school setting. This system aims to automate time-tracking, user authentication, and session management, providing real-time insights and secure access control to ensure efficient and accountable use of computer lab facilities by students.

2.2 Specific Objectives

1. To implement a secure login and account management system

Develop a login system that allows students to authenticate with unique IDs and passwords, tracks login attempts, and prevents unauthorized access. [Specific objective 2]

2. To create a session tracking system

Design and implement a timer-based session management feature that records each student's usage time. This system should track login and logout times, calculate session durations, and log this data in a centralized database.

3. Streamline User Access and Logout Processes

Design and implement login and logout workflows that are quick and user-friendly, supporting a smooth experience and accurate session tracking for all users accessing the computer lab systems.

3. Object-Oriented Programming Discussion

3.1 Understanding OOP

Object-oriented programming (OOP) is built around four core principles: encapsulation, inheritance, polymorphism, and abstraction. Encapsulation bundles data and methods into a class while restricting access to protect the integrity of the data, promoting modularity. Inheritance allows a subclass to inherit properties and behaviors from a superclass, enabling code reuse and hierarchical relationships. Polymorphism enables objects of different classes to be treated as instances of a common superclass, allowing methods to operate on various types of objects flexibly. Abstraction simplifies interaction with complex systems by hiding implementation details and exposing only essential features, typically through abstract classes or interfaces. Together, these principles foster the development of robust, scalable, and maintainable software systems.3.2 Implementation in the Project

In my Computer Management project, I applied the principles of object-oriented programming (OOP) as follows:

Encapsulation: Each class in the project encapsulates its data and behavior. For instance, the ClientForm1 class manages the client-side login process, containing attributes like StudentID and Password, and methods like ValidateLogin() and LogOut(). By using private fields and public methods, I restrict direct access to sensitive data, ensuring that interactions with the class are controlled and data integrity is maintained.

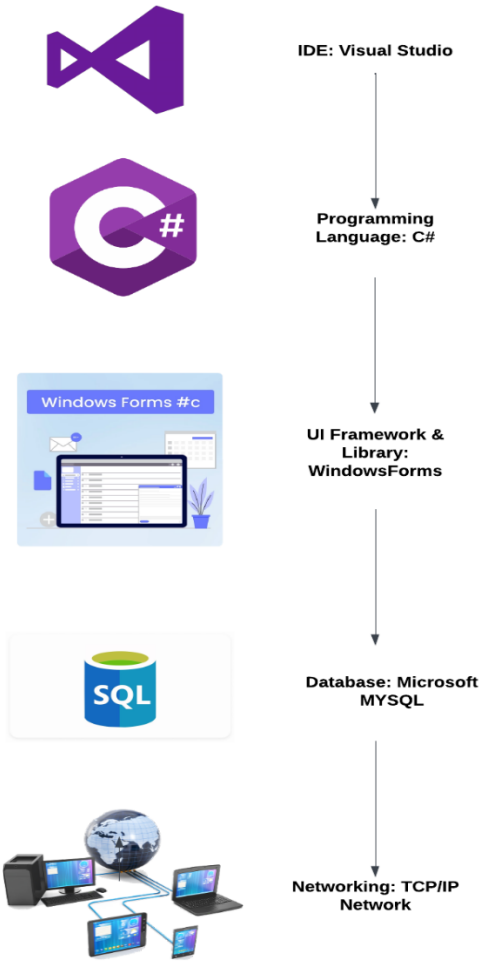
Inheritance: I created a base class called ClientSession that encapsulates common functionality for both ClientSessions1 and ClientSessions2. This class includes shared methods such as StartSession() and EndSession(), which handle session management. The subclasses inherit these methods, allowing me to extend functionality while reusing the existing code. For example, ClientSessions1 and ClientSessions2 each have specific attributes for their respective client PCs while leveraging the shared methods from ClientSession.

Polymorphism: I implemented polymorphism through the use of interfaces and method overriding. For instance, the ISessionManagement interface defines a method UpdateSession(), which is implemented differently in ClientSessions1 and ClientSessions2. This allows the system to treat both types of sessions uniformly while executing their respective logic when the method is called, providing flexibility in handling session updates.

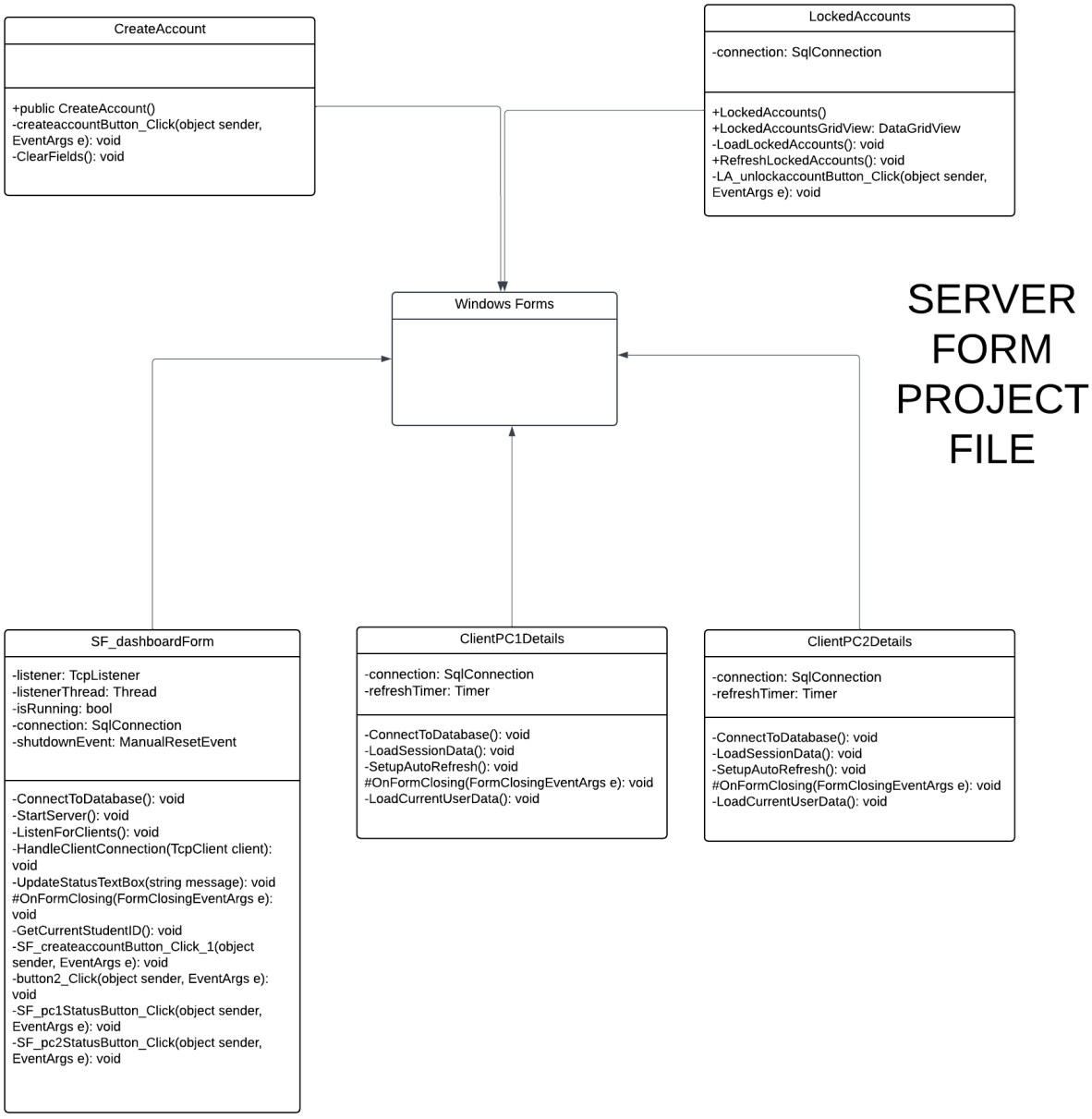
Abstraction: I utilized abstraction in the CreateAccounts class, which exposes a simple interface for account creation while hiding the complexity of database interactions. The CreateAccount() method accepts user input (like name and email) and interacts with the database to store the account details. Users of this class only need to know how to call CreateAccount(), without needing to understand the underlying database logic, thereby simplifying the interaction with the account creation process.

4. Diagrams

4.1 Conceptual Diagram



4.2 Class Diagram



4.3 Entity-Relationship Diagram

[Insert Entity-Relationship Diagram here.]

5. User Manual

5.1 User Interface Components

[Provide screenshots and functional descriptions for each User Interface component.]

5.2 Database Environment

[Insert screenshots of the database environment and describe the tables.]

6. Test Plan

6.1 Test Cycles

[Discuss the progress for each cycle, including objectives, test cases, results, and any issues encountered.]

7. Results and Discussion

[Present the results of your project, including any findings, challenges, and overall performance.]

8. Members Curriculum Vitae

8.1 [Member Name]

[Insert formal recent photo here (white background).]

8.2 Education:

[Details about education]

8.3 Skills:

[Relevant skills]

8.4 Role:

[Contributions list]