Contents

[1. System Architecture/Design Specification 2](#_Toc153841575)

[1. System Overview: 2](#_Toc153841576)

[2. Key Components: 2](#_Toc153841577)

[3. Features: 2](#_Toc153841578)

[4. User Interaction: 3](#_Toc153841579)

[5. System Flow: 3](#_Toc153841580)

[6. Expansion Possibilities: 3](#_Toc153841581)

[2. Implementation 4](#_Toc153841582)

[Tables and procedures 4](#_Toc153841583)

[Data connection and Server name 4](#_Toc153841584)

[Adding Student 5](#_Toc153841585)

[Adding Books 5](#_Toc153841586)

[3. Test cases to test two methods/functions of the system. 5](#_Toc153841587)

[Adding Book Data 5](#_Toc153841588)

[1. Introduction 5](#_Toc153841589)

[2. Testing Objectives 5](#_Toc153841590)

[3. Test Environment 6](#_Toc153841591)

[4. Test Data 6](#_Toc153841592)

[5. Test Cases 6](#_Toc153841593)

[6. Test Results and Analysis 6](#_Toc153841594)

[Validation Checks 6](#_Toc153841595)

[Successful Adding 7](#_Toc153841596)

[Showing Data 8](#_Toc153841597)

[Adding Student Data 8](#_Toc153841598)

[1. Introduction 8](#_Toc153841599)

[2. Testing Objectives 8](#_Toc153841600)

[3. Test Environment 8](#_Toc153841601)

[4. Test Data 8](#_Toc153841602)

[5. Test Cases 8](#_Toc153841603)

[6. Test Results and Analysis 9](#_Toc153841604)

[Validation Checks 9](#_Toc153841605)

[Correct Data Adding 10](#_Toc153841606)

[Showing Data 11](#_Toc153841607)

[4. SDLC Model 11](#_Toc153841608)

[1. Requirements Analysis: 11](#_Toc153841609)

[2. Design: 11](#_Toc153841610)

[3. Implementation 11](#_Toc153841611)

[4. Testing 12](#_Toc153841612)

[Main Releases 12](#_Toc153841613)

[Release 1.0 - Initial System Implementation 12](#_Toc153841614)

[Release 2.0 - Enhanced Functionality 12](#_Toc153841615)

[Release 3.0 - System Optimization and Bug Fixes 12](#_Toc153841616)

# System Architecture/Design Specification

## 1. System Overview:

- The system is a Library Management System (LMS) designed to efficiently manage library operations for both administrators and students.

## 2. Key Components:

**Classes:**

* **LMS (Library Management System):**

- Will be an abstract class for common functionalities.

* **Books:**

- Manages book-related operations (add, update, delete, reserve).

* **Library Manager:**

- Handles library management (add, delete, update, reserve books) and admin authentication.

* **Student:**

- Manages student-specific functions (purchase, view reserved books).

## 3. Features:

* **User Roles:**

- Library Administrator and Student.

* **Authentication:**

- Login System for admin and student having username and password

- In case of forgotten password, password can be recovered

* **Book Management:**

- Admin will be able to perform operations like create, delete, update books.

- Students will be able to view and purchase books.

* **Menu Navigation:**

- User-friendly menu for navigation of operations

* **Data Validation:**

- Using exceptions and proper data handling of data like what type for data to be hold.

## 4. User Interaction:

* **Main Menu:**

- Options for Library Administrator and Student roles.

* **Book Actions:**

- Admins can add, update, reserve, and delete books.

- Students can purchase books and view reserved books.

## 5. System Flow:

- User will interact with the system via main menu.

- OOP paradigm provides us facility that classes will handle their specific operations

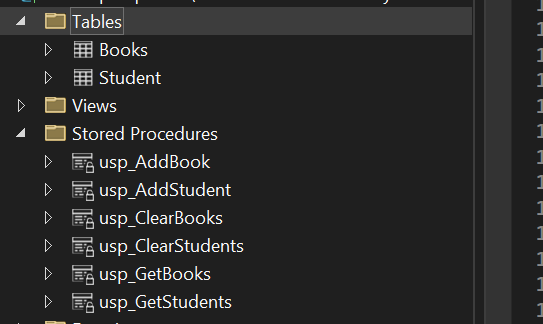
## 6. Expansion Possibilities:

- Designed for simplicity, allowing easy expansion of functionalities in the future.

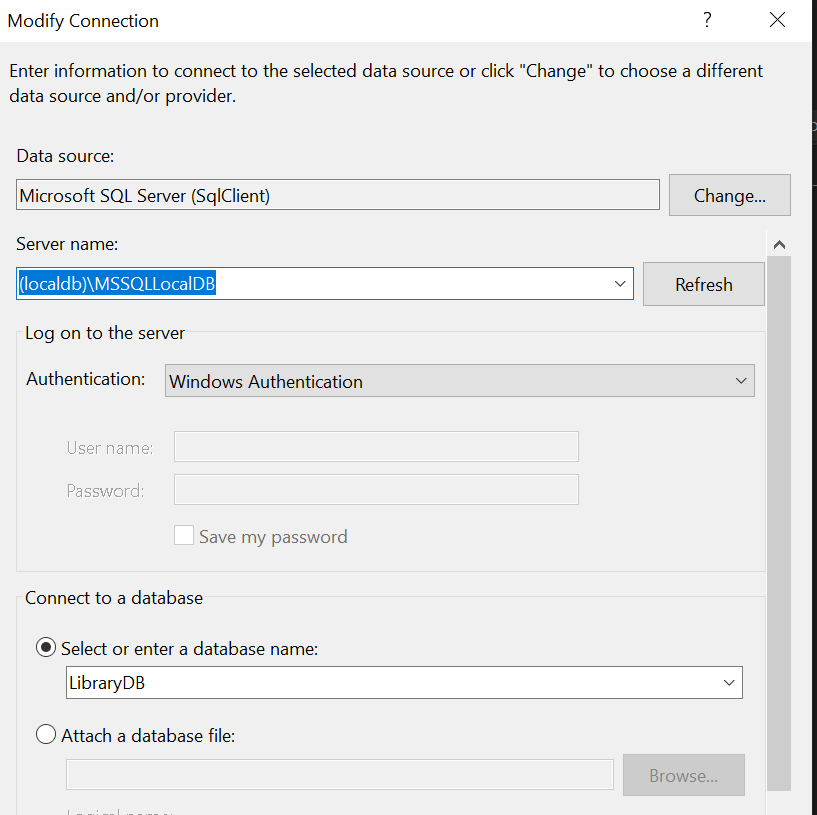
- It can be expanded for additional features like book categorization, user profiles, and more.

# Implementation

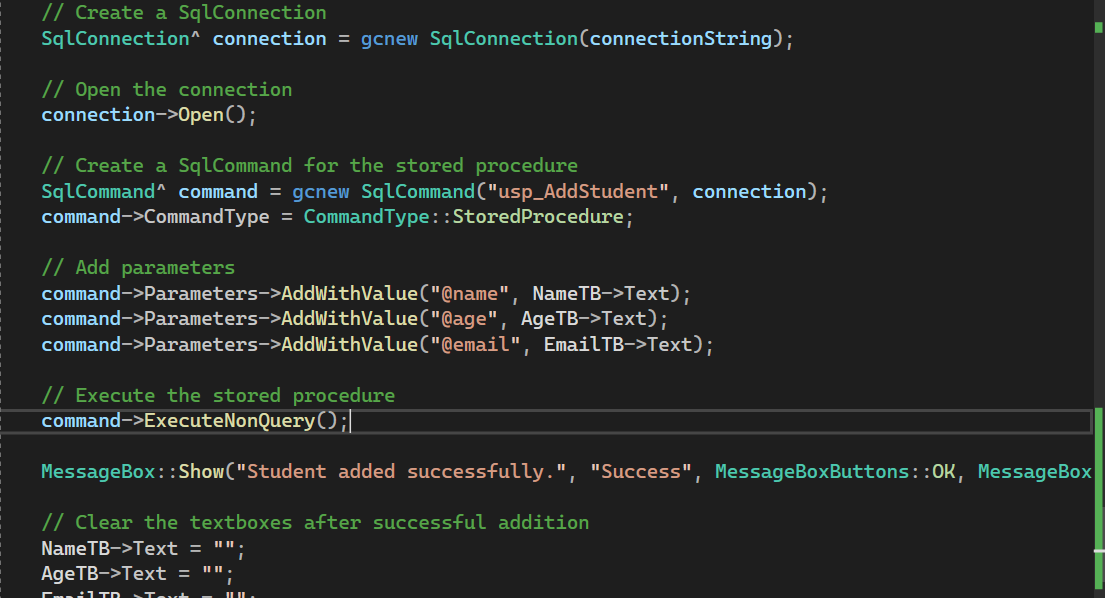
## Tables and procedures



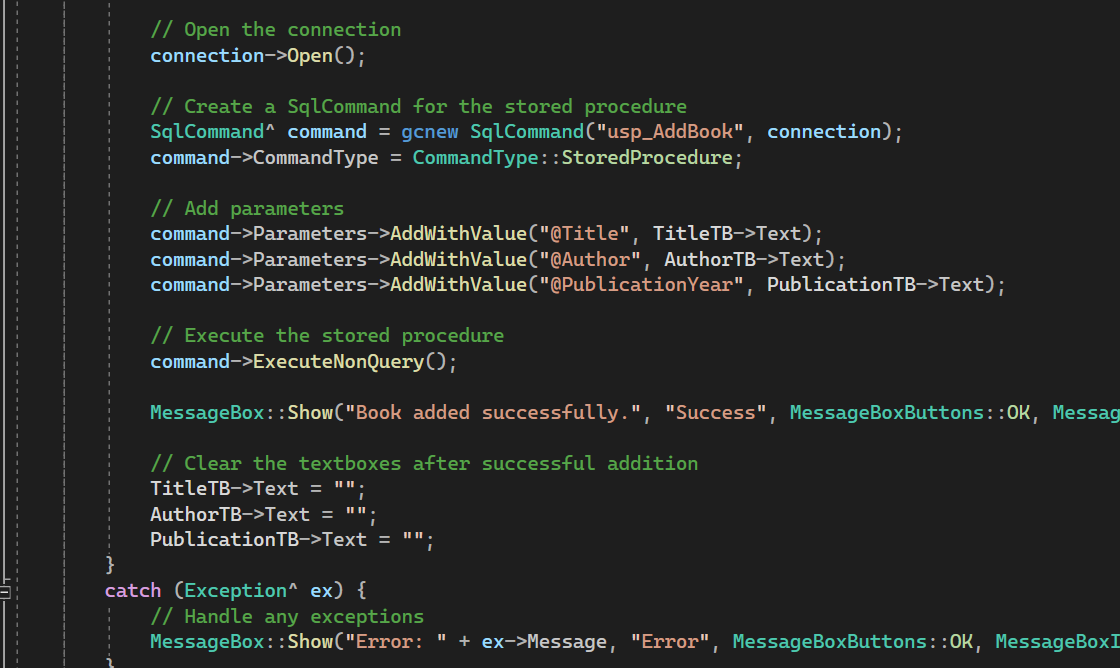
## Data connection and Server name



## Adding Student



## Adding Books



# Test cases to test two methods/functions of the system.

## Adding Book Data

### 1. Introduction

The save\_Click function plays a crucial role in the Library Management System, responsible for adding books. This report outlines test cases designed to assess the robustness and reliability of the save\_Click functionality.

### 2. Testing Objectives

The primary objectives of these test cases are to:

* Verify the correct handling of valid input data.
* Ensure the function gracefully manages invalid input scenarios.
* Validate the behaviour of the function when faced with a non-standard Publication year.
* Confirm the successful addition of a book to the database.

### 3. Test Environment

* Operating System: Windows
* Development Environment: Visual Studio
* Database: SQL Server (localdb)

### 4. Test Data

**Valid Input Data:**

* Title: "Programming Book"
* Author: "PAYEN Iye"
* Publication Year: "2023"

**Invalid Input Data:**

* Title: ""
* Author: "Jane Doe"
* Publication Year: "Invalid Year"

### 5. Test Cases

**Test Case 01: Valid Input - Book Added Successfully**

* Input Data: Valid data from Test Data 1  
  Expected Output: Success message displayed

**Test Case 02: Invalid Input - Empty Title**

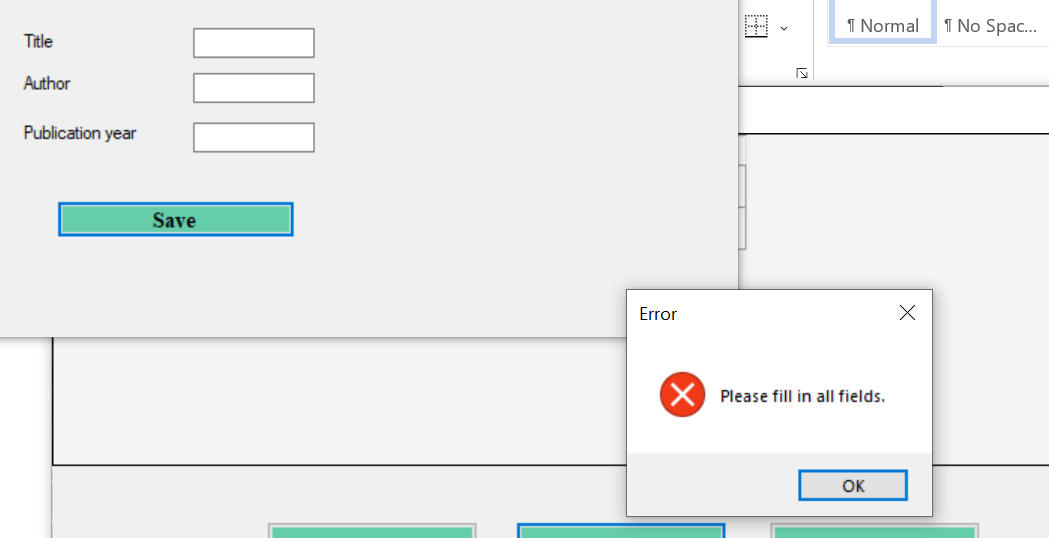
* Input Data: Invalid data from Test Data 2  
  Expected Output: Error message displayed

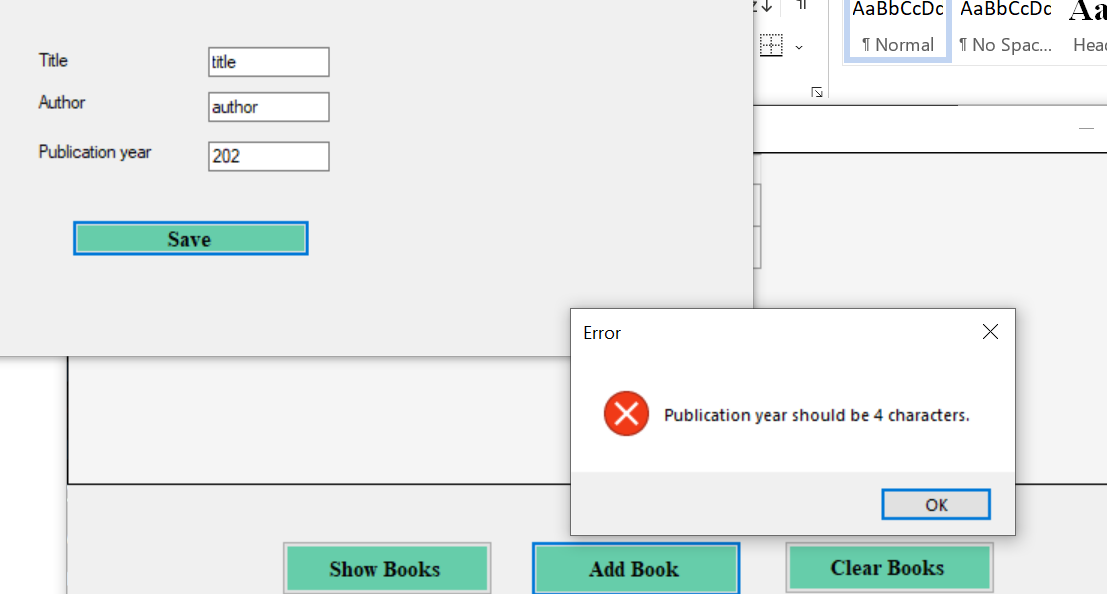
**Test Case 03: Invalid Input - Invalid Publication Year**

* Input Data: Invalid data from Test Data 2  
  Expected Output: Error message displayed

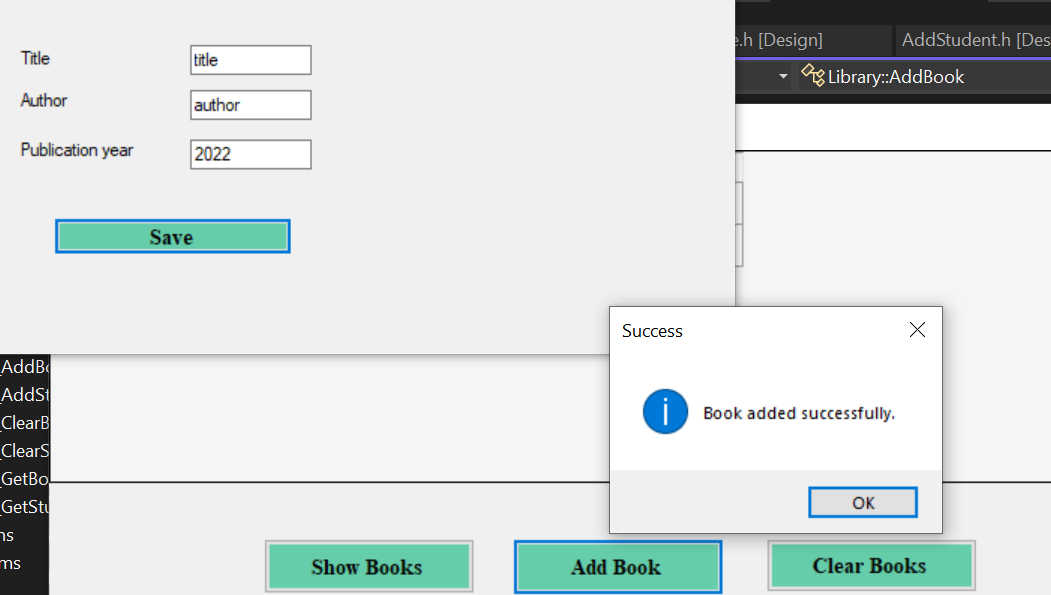
6. Test Results and Analysis

## Validation Checks

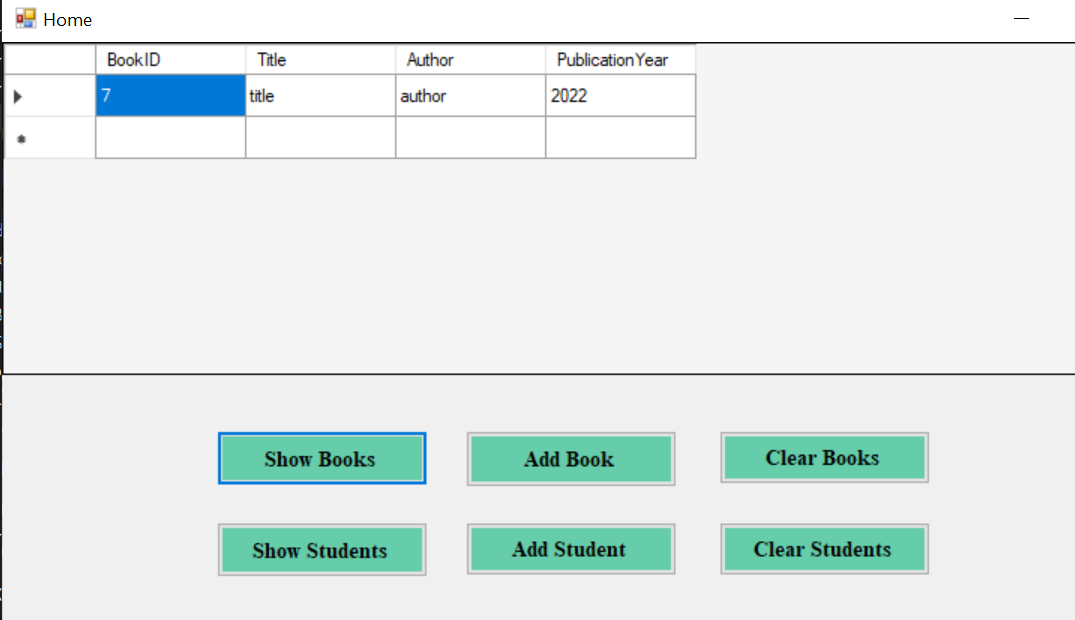




## Successful Adding



## Showing Data



## Adding Student Data

### 1. Introduction

The save\_Click function is a critical component of the Library Management System, dedicated to adding students. This report outlines test cases designed to evaluate the reliability and correctness of the save\_Click functionality.

### 2. Testing Objectives

The primary objectives of these test cases are to:

* Verify the correct handling of valid student information.
* Ensure the function gracefully manages invalid input scenarios, such as empty fields.
* Validate the proper validation of age and email fields.
* Confirm the successful addition of a student to the database.

### 3. Test Environment

* Operating System: Windows
* Development Environment: Visual Studio
* Database: SQL Server (localdb)

### 4. Test Data

**Valid Input Data:**

* Name: "Braham Nike"
* Age: "25"
* Email: "[braham.nikae@gmail.com](mailto:braham.nikae@gmail.com)"

**Invalid Input Data:**

* Empty Fields: One or more fields left empty.
* Invalid Age: "0"
* Invalid Email: "invalid\_email"

### 5. Test Cases

**Test Case 01: Valid Input - Student Added Successfully**

* Input Data: Valid data from Test Data 1  
  Expected Output: Success message displayed

**Test Case 02: Invalid Input - Empty Fields**

* Input Data: Invalid data from Test Data 2  
  Expected Output: Error message displayed for empty fields

**Test Case 03: Invalid Input - Invalid Age**

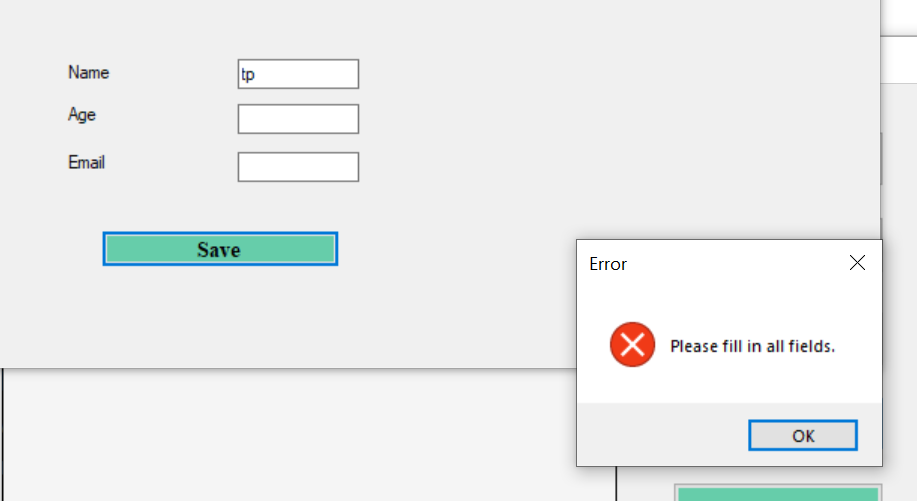
* Input Data: Invalid data from Test Data 2  
  Expected Output: Error message displayed for invalid age

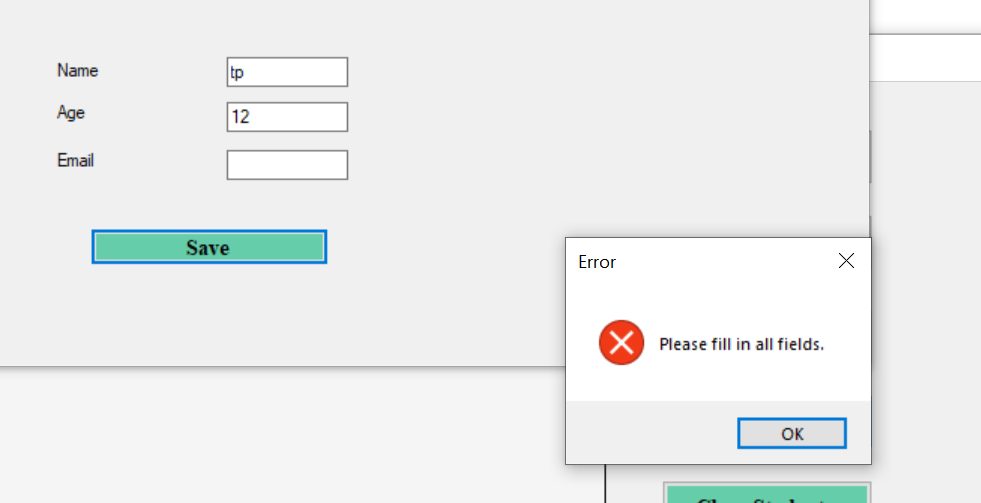
**Test Case 04: Invalid Input - Invalid Email**

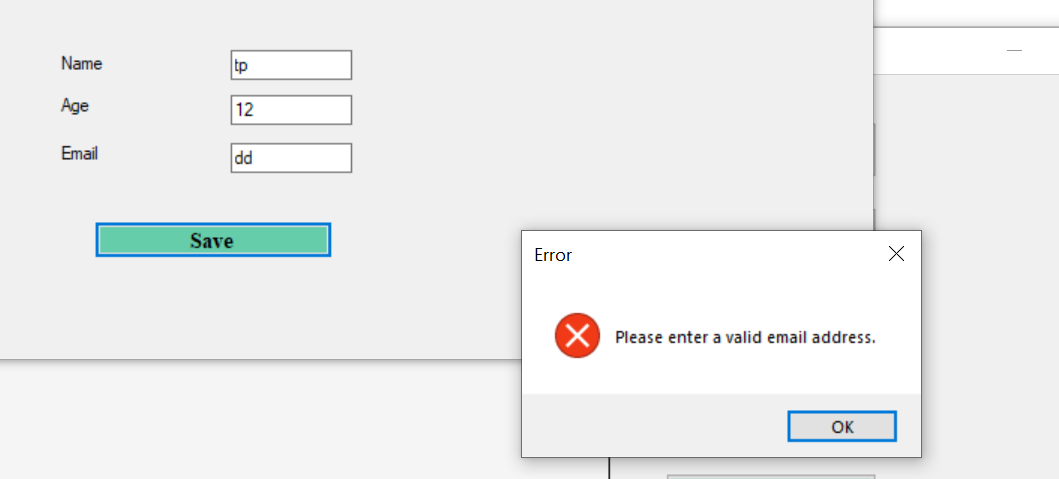
* Input Data: Invalid data from Test Data 2  
  Expected Output: Error message displayed for invalid email

### 6. Test Results and Analysis

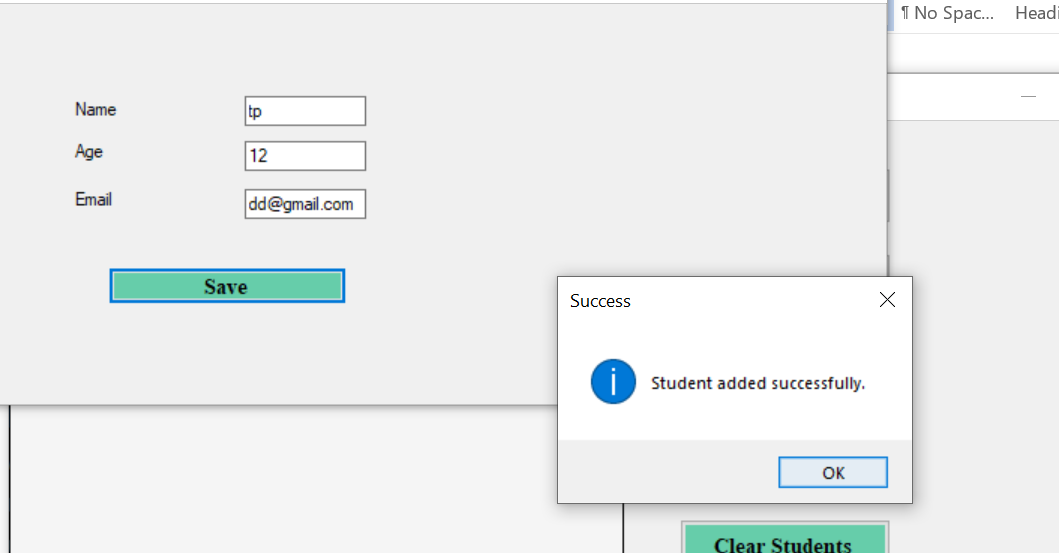
## Validation Checks



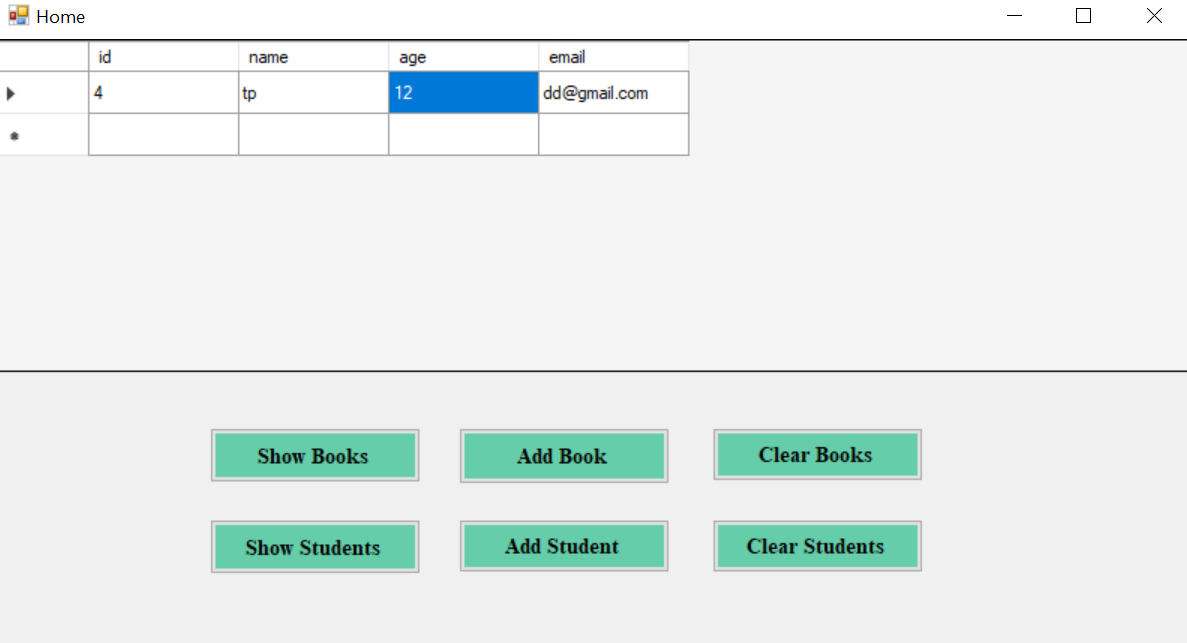




## Correct Data Adding



## Showing Data



# SDLC Model

**Waterfall model** will be suitable for this project as it is flexible and well-defined. Moreover, it is structured and easy to maintain.

## Requirements Analysis:

Identified the need for a comprehensive LMS through discussions with stakeholders that are students, library manager etx

Documented detailed requirements, including user roles, functionalities, and system behaviour.

## Design:

* **System Architecture:**

Defined a modular structure with classes (LMS, Books, Library Manager, Student) for scalability.

* **User Interface Design:**

Created a user-friendly menu for seamless navigation.

Incorporated basic data validation for a smooth user experience.

## Implementation

**Coding:**

Implemented classes and functionalities in C++ based on the design specifications.

Adhered to Object-Oriented Programming principles.

## Testing

* **Unit Testing:**

Checked individual components (classes, methods) for correctness.

* **Integration Testing:**

Ensured proper collaboration between classes and smooth system flow.

## Main Releases

### Release 1.0 - Initial ****System**** Implementation

**Features:**

* Basic book and student management.
* User interface for adding books and students.

### Release 2.0 - Enhanced Functionality

**Features:**

* Advanced search and filtering options.
* Improved user authentication.

### Release 3.0 - System Optimization and Bug Fixes

**Features:**

* Performance optimization.
* Resolution of reported bugs.