Computer Science & Engineering Department
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
April, 2024

CRICKET TEAM MANAGEMENT SYSTEM

Work Report

ANAND PRAKASH 2212041
PULKIT 2212042

OOPS Project

CONTENTS

1. Problem Statement

The problem this project addresses is the need for an efficient library management system. Traditional paper-based systems are prone to errors and lack scalability. The project aims to develop a library management system using C++ and Object-Oriented Programming principles. It involves creating classes for library items such as books and students, implementing functionalities for adding new items, displaying item details, and managing student records. The system aims to provide efficient organization and tracking of library resources.

1.1 Inefficiency of Traditional Systems:
Traditional paper-based library management systems are inefficient due to manual processes involved in recording and tracking library resources.
Maintaining paper records is time-consuming, prone to errors, and lacks scalability as the library grows.

1.2 Need for Efficiency and Accuracy:
There is a pressing need for a more efficient and accurate system that can handle the complexities of modern libraries.
An automated system can streamline processes, reduce errors, and improve the overall management of library resources.

1.3 Utilization of Technology:
Leveraging technology, specifically C++ and Object-Oriented Programming principles, provides an opportunity to develop a robust library management system.
C++ offers the advantages of performance, portability, and flexibility, making it a suitable choice for implementing such a system.

2. Introduction

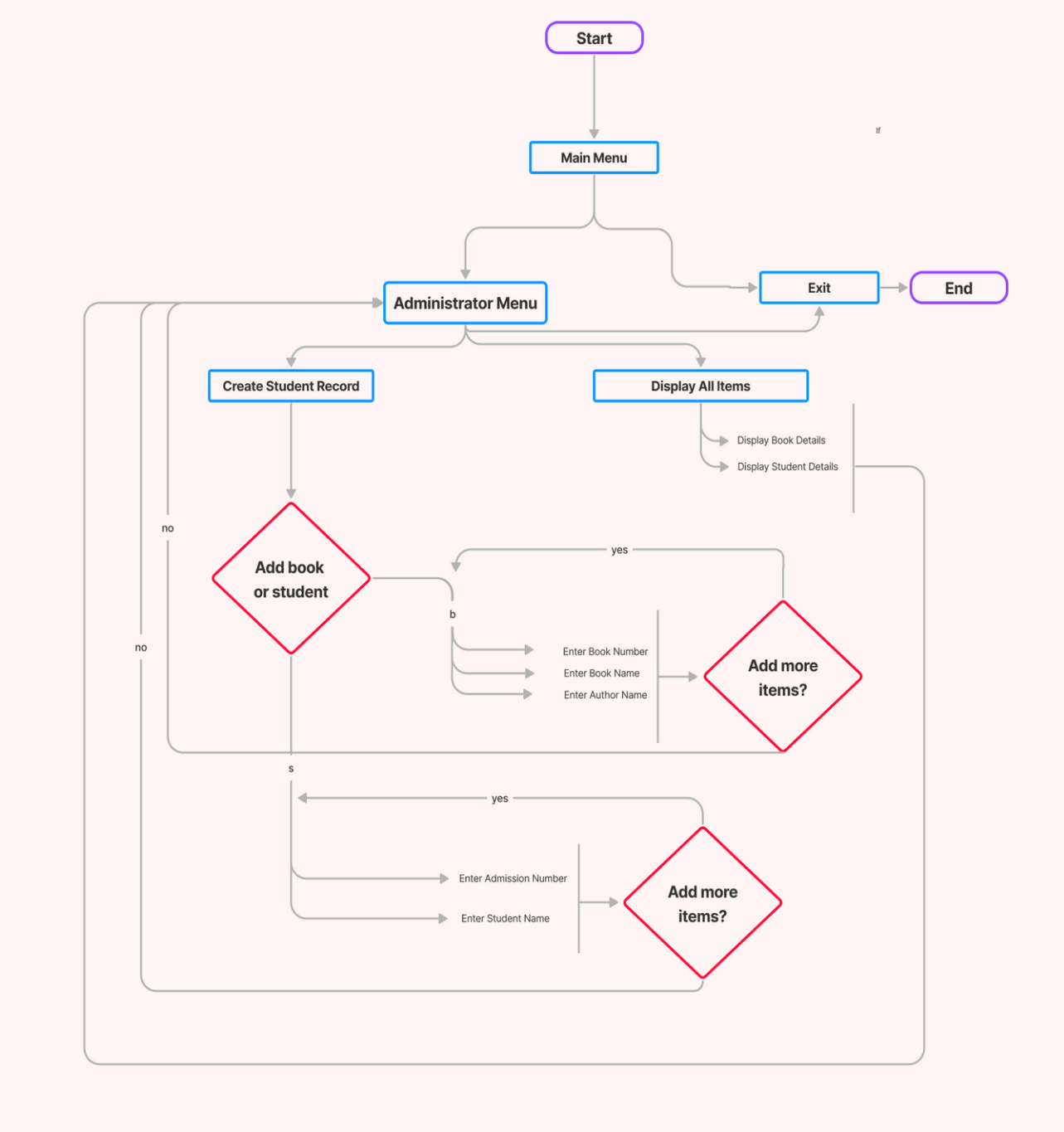
This Object-Oriented Programming (OOP) project is a simple library management system implemented in C++. It leverages key OOP principles such as inheritance, polymorphism and encapsulation. The project consists of classes representing library items, including books and students, each with their own attributes and behaviours. It demonstrates how OOP enables modular, organized, and extensible code by encapsulating data and functionality within classes and leveraging inheritance to promote code reuse. Through polymorphism, it achieves flexibility by allowing different types of library items to be treated uniformly, facilitating easy expansion and maintenance of the system.

In addition to the core OOP principles of inheritance, polymorphism, and encapsulation, this library management system project also emphasizes the importance of abstraction and modularity in software design. By abstracting away implementation details and focusing on defining clear interfaces between classes, the project ensures that each component can be developed, tested, and maintained independently. This modular approach not only facilitates code organization but also enables seamless integration of new features and enhancements without impacting existing functionality. Furthermore, the project adheres to the SOLID principles of object-oriented design, promoting single responsibility, open-closed, Liskov substitution, interface segregation, and dependency inversion. By adhering to these principles, the project aims to achieve a high level of cohesion, flexibility, and maintainability, laying the foundation for a robust and scalable library management system.

The OOPS topics used in the project are:

1. Classes and Objects
2. Inheritance
3. Polymorphism
4. Encapsulation
5. Dynamic Memory Allocation
6. Abstraction

3. Flowchart


Start
Main Menu
Exit
End
Administrator Menu
Create Student Record
Display All Items
Display Book Details
Display Student Details
no
Add book
or student
yes
b
Enter Book Number
Enter Book Name
Enter Author Name
Add more
items?
no
s
yes
Enter Admission Number
Enter Student Name
Add more
items?

4. Algorithms

4.1 Polymorphism:

Polymorphism is utilized through the use of virtual functions in the LibraryItem base class. The displayDetails() function is declared as a pure virtual function in the LibraryItem class, making it mandatory for derived classes (Book and Student) to provide their own implementations. This allows different types of library items to be treated uniformly, as demonstrated in the displayAllItems() function where the displayDetails() function is called on each item without needing to know its specific type.

```
Class LibraryItem {
    virtual void displayDetails() = 0;
}

Class Book : public LibraryItem {
    void displayDetails() {
        // Implementation for displaying book details
    }
}

Class Student : public LibraryItem {
    void displayDetails() {
        // Implementation for displaying student details
    }
}
```

4.2 Dynamic Memory Management:

Dynamic memory allocation is used to create objects of Book and Student classes using the new keyword. The dynamically allocated objects are stored in a vector of pointers to LibraryItem.

```
// Declaration of vector to store pointers to LibraryItem objects
vector
```

4.3 Dynamic Casting:

Dynamic casting (dynamic_cast) is used to check the actual type of each item in the vector during searching operations (checkBookExists and checkStudentExists). This allows the program to differentiate between Book and Student objects stored in the vector of LibraryItem pointers.

```
// Function to check if a book exists in the vector
bool checkBookExists(const vector```
```

4.4 User Input Handling:

The program interacts with the user through the console. User input is collected using cin and validated to perform appropriate actions such as adding a new book or student record, displaying all items, or exiting the program.

```
// Prompt user for input
cout << "Enter choice: ";
// Read user input
cin >> choice;
// Handle user choice using a switch statement
switch (choice) {
    case 1:
        // Perform action for choice 1
        break;
    case 2:
        // Perform action for choice 2
        break;
    // Add more cases for additional choices
    default:
        // Handle invalid input
        break;
}
```

4.5 Menu-Driven Program:

The program utilizes a menu-driven approach to provide different functionalities to the user.
A switch statement is used to handle user choices in both the main() and adminMenu() functions.

```
// Main program loop
do {
    // Display menu options
    cout << "1. Option 1" << endl;
    cout << "2. Option 2" << endl;
    // Prompt user for choice
    cout << "Enter choice: ";
    // Read user choice
    cin >> choice;
    // Perform actions based on user choice
    switch (choice) {
        case 1:
            // Perform action for option 1
            break;
        case 2:
            // Perform action for option 2
            break;
        // Add more cases for additional options
        default:
            // Handle invalid input
            break;
    }
} while (choice != exitChoice);
```

5. Conclusion

In addition to its technical achievements, this project also underscores the broader impact of Object-Oriented Programming (OOP) principles on software development practices. By promoting a structured and systematic approach to problem-solving, OOP encourages developers to think in terms of objects, relationships, and behaviours, fostering a deeper understanding of complex systems. Furthermore, the project exemplifies the collaborative nature of software development, as it involves the collective effort of programmers, designers, and stakeholders to conceptualize, design, and implement the library management system.

Moreover, the adoption of OOP principles in this project reflects a commitment to continuous improvement and innovation in the field of software engineering. By embracing best practices and industry standards, developers can ensure the reliability, scalability, and security of software systems, thereby meeting the evolving needs and expectations of users. Furthermore, the project's emphasis on extensibility and customization underscores the importance of adaptability in an ever-changing technological landscape, where new requirements and challenges emerge regularly.

In conclusion, this project not only demonstrates the tangible benefits of Object-Oriented Programming in developing practical solutions but also highlights its broader implications for software engineering. By embracing OOP principles, developers can create robust, adaptable, and user-friendly systems that empower individuals and organizations to thrive in the digital age. As such, this project serves as a testament to the enduring relevance and efficacy of OOP in addressing real-world challenges and driving innovation in software development.