Group Members

- Bushra Ansar (AI-003)
- Tamia Naeem (AI-004)
- Hareem Nadeem (AI-025)
- Bushra Atiq (AI-028)

Library Management System Report

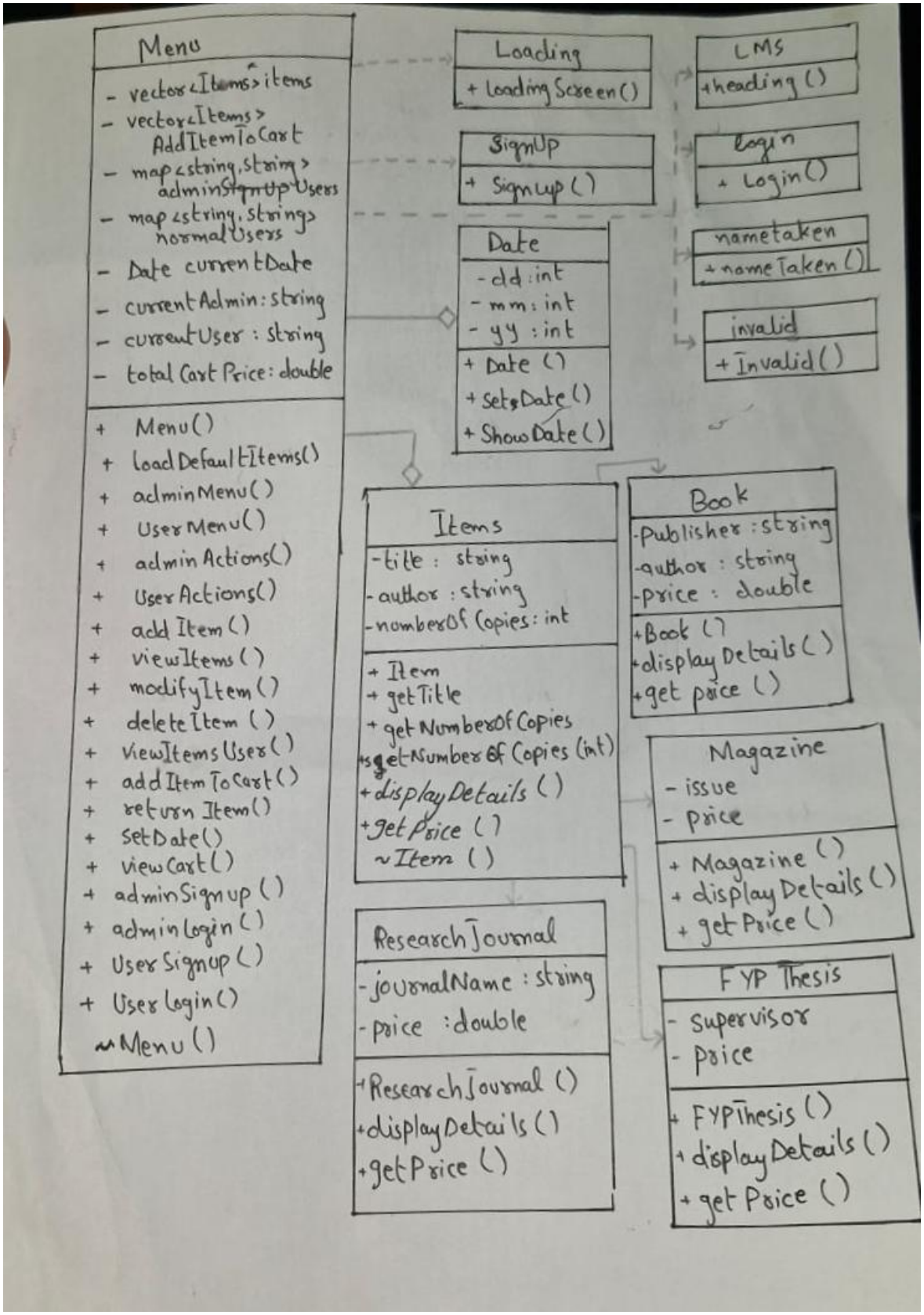
1. Project Introduction

The Library Management System (LMS) is designed to manage library resources efficiently. The system enables both administrators and users to interact with the library. Administrators can add, view, delete, and modify items, while users can browse items, add them to their cart, and return borrowed items.

2. Project Description

The LMS consists of multiple classes representing various components of the system. The main functionalities include user authentication, item management, and cart management. The system supports different types of items such as books, magazines, research journals, and FYP theses.

UML Diagram


Menu
- vector<Items> items
- vector<Items> AddItemToCart
- map<string,string> adminSignUpUsers
- map<string,string> normalUsers
- Date currentDate
- currentAdmin: string
- currentUser : string
- totalCartPrice: double
+ Menu()
+ loadDefaultItems()
+ adminMenu()
+ UserMenu()
+ adminActions()
+ UserActions()
+ addItem()
+ viewItems()
+ modifyItem()
+ deleteItem()
+ viewItemsUser()
+ addItemToCart()
+ returnItem()
+ setDate()
+ viewCart()
+ adminSignup()
+ adminLogin()
+ UserSignup()
+ UserLogin()
~Menu()
Loading
+ LoadingScreen()
SignUp
+ SignUp()
Date
- dd:int
- mm:int
- yy:int
+ Date()
+ setDate()
+ ShowDate()
LMS
+heading()
Login
+ Login()
nametaken
+nameTaken()
invalid
+Invalid()
Items
-title : string
-author : string
-numberOfCopies: int
+ Item
+ getTitle
+ getNumberOfCopies
+setNumberOfCopies(int)
+displayDetails()
+getPrice()
~Item()
Book
-Publisher :string
-author : string
-price : double
+Book()
+displayDetails()
+get price()
Magazine
- issue
- price
+ Magazine()
+ displayDetails()
+ getPrice()
ResearchJournal
-journalName : string
-price :double
+ResearchJournal()
+displayDetails()
+getPrice()
FYP Thesis
- Supervisor
- price
+ FYPThesis()
+ displayDetails()
+ getPrice()

3. Explanation of Classes

1. loading

This class provides a static function to display a loading screen with a delay.

- **Function:**
  - static void loadingScreen(): Displays a loading message with a delay to simulate loading time.

```
class loading {
public:
    static void loadingScreen() {
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\tLOADING";
        for (int i = 0; i < 3; ++i) {
            cout << ".";
            cout.flush();
            this_thread::sleep_for(chrono::seconds(1));
        }
        cout << "\n";
    }
};
```

2. LMS

This class provides a static function to display the heading of the Library Management System.

- **Function:**
  - static void heading(): Displays the heading of the Library Management System.

```
class LMS {
public:
    static void heading() {
        cout << "\t\t\t\t\t\t\t==========================================\n";
        cout << "\t\t\t\t\t\t\t        LIBRARY MANAGEMENT SYSTEM         \n";
        cout << "\t\t\t\t\t\t\t==========================================\n";
    }
};
```

3. signUp

This class provides a static function to simulate a signup process.

- **Function:**

  - static void signup(): Displays a signup successful message with a delay.

```
class signUp {
public:
    static void signup() {
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\tSIGNUP SUCCESSFUL";
        for (int i = 0; i < 3; ++i) {
            cout << "!";
            cout.flush(); // Ensure the ! is printed immediately
            this_thread::sleep_for(chrono::seconds(1));
        }
        cout << "\n";
    }
};
```

4. login

This class provides a static function to simulate a login process.

- **Function:**

  - static void Login(): Displays a login successful message with a delay.

```
class login {
public:
    static void Login() {
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\tLOGIN SUCCESSFUL";
        for (int i = 0; i < 3; ++i) {
            cout << "!";
            cout.flush(); // Ensure the ! is printed immediately
            this_thread::sleep_for(chrono::seconds(1));
        }
        cout << "\n";
    }
};
```

5. nametaken

This class provides a static function to display a message indicating a username is already taken.

- **Function:**

  - static void nameTaken(): Displays a message indicating the username is already taken with a delay.

```
class nametaken {
public:
    static void nameTaken() {
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\tUSER NAME IS ALREADY TAKEN";
        for (int i = 0; i < 3; ++i) {
            cout << "!";
            cout.flush(); // Ensure the ! is printed immediately
            this_thread::sleep_for(chrono::seconds(1));
        }
        cout << "\n";
    }
};
```

6. invalid

This class provides a static function to display an invalid credentials message.

- **Function:**
    - static void Invalid(): Displays an invalid credentials message with a delay.

```
class invalid {
public:
    static void Invalid() {
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\tINVALID CREDENTIALS";
        for (int i = 0; i < 3; ++i) {
            cout << "!";
            cout.flush();
            this_thread::sleep_for(chrono::seconds(1));
        }
        cout << "\n";
    }
};
```

7. Item

An abstract base class representing a general item in the library.

- **Members:**
    - string title, string author, int numberOfCopies: Attributes common to all items.
- **Functions:**
    - Item(string t, string a, int n): Constructor to initialize common attributes.
    - virtual void displayDetails() const = 0: Pure virtual function to display item details.
    - virtual double getPrice() const = 0: Pure virtual function to get the price of the item.
    - virtual ~Item() = default: Virtual destructor.

- string getTitle() const, int getNumberOfCopies() const, void setNumberOfCopies(int n): Accessor and mutator functions.

```
class Item {
protected:
    string title;
    string author;
    int numberOfCopies;

public:
    Item(string t, string a, int n) : title(t), author(a), numberOfCopies(n) {}
    virtual void displayDetails() const = 0;
    virtual double getPrice() const = 0;
    virtual ~Item() = default;

    string getTitle() const { return title; }
    int getNumberOfCopies() const { return numberOfCopies; }
    void setNumberOfCopies(int n) { numberOfCopies = n; }
};
```

8. Book

A class derived from Item representing a book.

- **Members:**
  - string publisher, string edition, double price: Additional attributes specific to books.
- **Functions:**
  - Book(string t, string a, int n, double p, string pub, string ed): Constructor to initialize book attributes.
  - void displayDetails() const override: Displays book details.
  - double getPrice() const override: Returns the price of the book.

```cpp
class Book : public Item {
    string publisher;
    string edition;
    double price;

public:
    Book(string t, string a, int n, double p, string pub, string ed)
        : Item(t, a, n), price(p), publisher(pub), edition(ed) {}


    void displayDetails() const override {
        this_thread::sleep_for(seconds(1));
        cout << "\t\t   __________\n";
        cout << "\t\t  |   BOOKS   | \n";
        cout << "\t\t  ''''''''''";
        this_thread::sleep_for(seconds(2));
        cout << "\n-------------------------------------------------\n";
        cout << "\nBook: " << title << "\nAuthor: " << author << "\nPublisher: " << publisher <<
            "\nEdition: " << edition << "\nCopies: " << numberOfCopies << "\nPrice: $" << fixed
            << setprecision(2) << price << endl;
        cout << "-------------------------------------------------\n";
    }
    double getPrice() const override {
        return price;
    }
};
```

9. Magazine

A class derived from Item representing a magazine.

- **Members:**
  - string issue, double price: Additional attributes specific to magazines.
- **Functions:**
  - Magazine(string t, string a, int n, double p, string i): Constructor to initialize magazine attributes.
  - void displayDetails() const override: Displays magazine details.
  - double getPrice() const override: Returns the price of the magazine.

```cpp
class Magazine : public Item {
    string issue;
    double price;

public:
    Magazine(string t, string a, int n, double p, string i)
        : Item(t, a, n), price(p), issue(i) {}

    void displayDetails() const override {
        this_thread::sleep_for(seconds(1));
        cout << "\t\t  _______________\n";
        cout << "\t\t |   MAGAZINE    | \n";
        cout << "\t\t '''''''''''''''''";
        this_thread::sleep_for(seconds(2));
        cout << "\n-----------------------------------------------------\n";
        cout << "\nTitle: " << title << "\nAuthor: " << author << "\nIssue: " << issue <<
        "\nCopies: " << numberOfCopies << "\nPrice: $" << fixed << setprecision(2) << price << endl;
        cout << "-----------------------------------------------------\n";
    }
    double getPrice() const override {
        return price;
    }
};
```

10. ResearchJournal

A class derived from Item representing a research journal.

- **Members:**
  - string journalName, double price: Additional attributes specific to research journals.
- **Functions:**
  - ResearchJournal(string t, string a, int n, double p, string jn): Constructor to initialize research journal attributes.
  - void displayDetails() const override: Displays research journal details.
  - double getPrice() const override: Returns the price of the research journal.

```cpp
class ResearchJournal : public Item {
    string journalName;
    double price;

public:
    ResearchJournal(string t, string a, int n, double p, string jn)
        : Item(t, a, n), price(p), journalName(jn) {}

    void displayDetails() const override {
        this_thread::sleep_for(seconds(1));
        cout << "\t\t _______________________\n";
        cout << "\t\t|   RESEARCH JOURNAL    | \n";
        cout << "\t\t`````````````````````````";
        this_thread::sleep_for(seconds(2));
        cout << "\n-------------------------------------------------\n";
        cout << "\nTitle: " << title << "\nResearcher: " << author << "\nJournal: " << journalName
             << "\nCopies: " << numberOfCopies << "\nPrice: $" << fixed << setprecision(2) << price << endl;
        cout << "-------------------------------------------------\n";
    }
    double getPrice() const override {
        return price;
    }
};
```

11. FYPThesis

A class derived from Item representing a final year project thesis.

- **Members:**
  - string supervisor, double price: Additional attributes specific to FYP theses.
- **Functions:**
  - FYPThesis(string t, string a, int n, double p, string s): Constructor to initialize FYP thesis attributes.
  - void displayDetails() const override: Displays FYP thesis details.
  - double getPrice() const override: Returns the price of the FYP thesis.

```cpp
class FYPThesis : public Item {
    string supervisor;
    double price;

public:
    FYPThesis(string t, string a, int n, double p, string s)
        : Item(t, a, n), price(p), supervisor(s) {}

    void displayDetails() const override {
        this_thread::sleep_for(seconds(1));
        cout << "\t\t _____________________\n";
        cout << "\t\t|      FYP THESIS      |\n";
        cout << "\t\t`````````````````````";
        this_thread::sleep_for(seconds(2));
        cout << "\n-------------------------------------------------\n";
        cout << "Type: FYP Thesis\n";
        cout << "Title: " << title << "\nAuthor: " << author << "\nSupervisor: " << supervisor
             << "\nCopies: " << numberOfCopies << "\nPrice: $" << fixed << setprecision(2) << price << endl;
        cout << "-------------------------------------------------\n";
    }
    double getPrice() const override {
        return price;
    }
};
```

12. Date

This class represents a date.

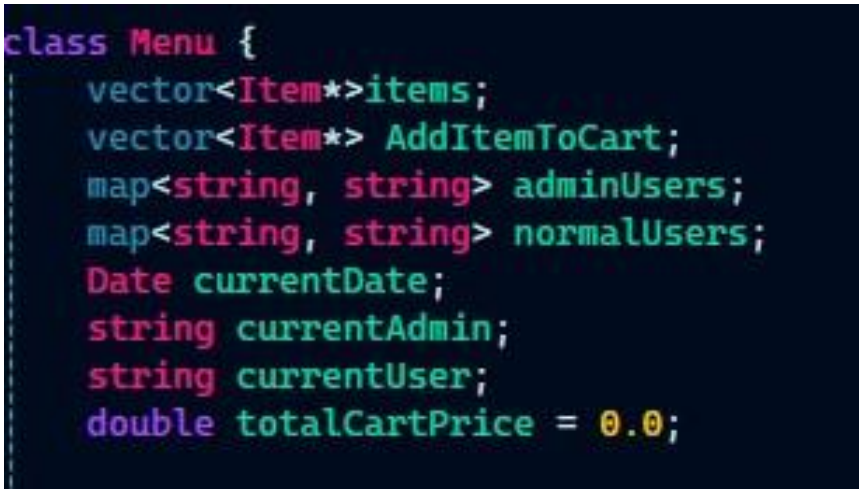
- **Members:**
  - int dd, mm, yy: Attributes representing day, month, and year.
- **Functions:**
  - Date(): Constructor to initialize date to default values.
  - void setDate(): Sets the date based on user input.
  - void showDate() const: Displays the current date.

```
class Date {
private:
    int dd, mm, yy;

public:
    Date() : dd(1), mm(1), yy(2000) {}

    void setDate() {
        cout << "Enter date (dd mm yyyy): ";
        cin >> dd >> mm >> yy;
    }

    void showDate() const {
        cout << "Current Date: " << dd << "/" << mm << "/" << yy << endl;
    }
};
```

13. Menu

This class represents the main menu and handles user interactions.

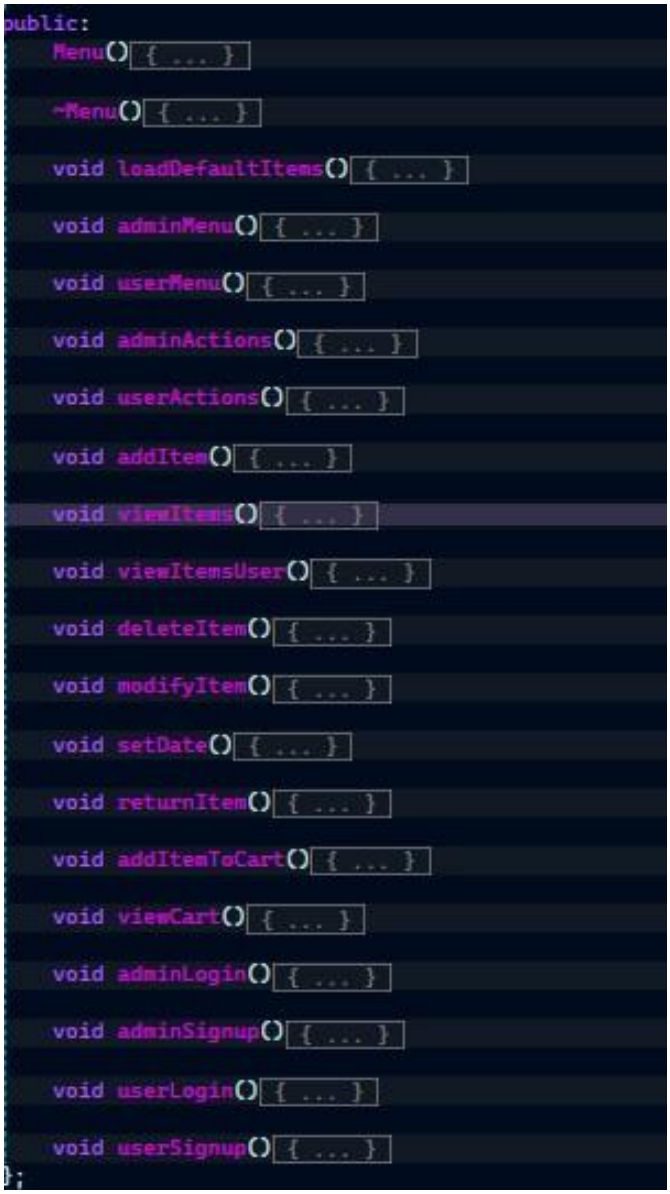
- **Members:**
    - vector<Item*> items: A collection of all items in the library.
    - vector<Item*> AddItemToCart: A collection of items added to the cart.
    - map<string, string> adminUsers, map<string, string> normalUsers: Maps to store admin and user credentials.
    - Date currentDate: Represents the current date.
    - string currentAdmin, string currentUser: Stores the currently logged-in admin and user.
        - double totalCartPrice: Tracks the total price of items in the cart.



```
class Menu {
    vector<Item*>items;
    vector<Item*> AddItemToCart;
    map<string, string> adminUsers;
    map<string, string> normalUsers;
    Date currentDate;
    string currentAdmin;
    string currentUser;
    double totalCartPrice = 0.0;
```

- **Functions:**

  - Menu(): Constructor to initialize the menu and load default items.
  - ~Menu(): Destructor to clean up dynamically allocated items.
  - void loadDefaultItems(): Loads default items into the library.
  - void adminMenu(): Displays the admin menu and handles admin actions.
  - void userMenu(): Displays the user menu and handles user actions.
  - void adminActions(), void userActions(): Handles admin and user login/signup actions.
  - void addItem(): Adds a new item to the library.
  - void viewItems(), void viewItemsUser(): Displays all items or available items to the user.
  - void deleteItem(): Deletes an item from the library.
  - void modifyItem(): Modifies an existing item.
  - void setDate(): Sets the current date.
  - void returnItem(): Returns an item to the library.
  - void addItemToCart(): Adds an item to the user's cart. ○ void viewCart(): Displays items in the user's cart.
  - void adminLogin(), void adminSignup(), void userLogin(), void userSignup(): Handles admin and user login/signup processes.



Explanation of Functions

Each class and function have been explained above in the context of the classes they belong to. The primary functions handle the display of information, user input, and managing items in the library.

Explanation of Libraries

1. <iostream>

- Used for input and output operations.

2. <vector>

- Provides the vector container for storing a collection of items.

3. <string>

- Used for handling string operations.

4. <map>

- Provides the map container for storing key-value pairs, used for storing user credentials.

5. <algorithm>

- Provides functions like find_if for searching through collections.

6. <cstdlib>

- Provides functions like system("cls") to clear the console screen.

7. <thread>

- Used for creating threads and adding delays.

8. <chrono>

- Provides time-related functions for handling delays.

9. <iomanip>

- Used for setting the precision of floating-point numbers for display.

4. Relationships Between Classes

4.1 Inheritance

- The Book, Magazine, ResearchJournal, and FYPThesis classes inherit from the Item class, allowing them to share common attributes and methods.

4.2 Polymorphism

- The Item class defines virtual functions displayDetails() and getPrice(), which are overridden by derived classes to provide specific implementations.

4.3 Abstraction

- The Item class is an abstract class, providing a base interface for different item types without exposing implementation details.

4.4 Encapsulation

- Each class encapsulates its data and provides public methods to interact with it, ensuring data integrity and security.

4.5 Aggregation

- The Menu class aggregates multiple Item objects, managing their lifecycles and interactions.

4.6 Composition

- The Menu class includes a Date object to manage date-related operations, demonstrating a strong relationship where Menu owns Date.

5. Limitations

- The current system does not support persistent storage; all data is lost when the program terminates.
- User authentication is basic, with no encryption or security measures for stored passwords.
- The interface is text-based and lacks a graphical user interface (GUI).

6. Future Enhancements

- Implement persistent storage using databases.
- Enhance security for user authentication and data storage.
- Develop a GUI for better user experience.
- Add more item types and functionalities.

7. Conclusion

The Library Management System is a comprehensive project that demonstrates the use of various object-oriented programming concepts such as inheritance, polymorphism, and encapsulation. It provides a clear structure for managing different types of library items, user interactions, and administrative tasks. The project utilizes standard C++ libraries for handling input/output operations, collections, and time management, making it a robust and interactive system.

8. Group Members' Contributions

Tamia Naeem: Tamia is responsible for designing and developing the core classes that form the backbone of the Library Management System. She created the Item class, which serves as a base class for all types of items in the library. She then derived specific classes such as Book, Magazine, ResearchJournal, and FYPThesis from the Item class, ensuring that each type of item has its own unique attributes and methods. Tamia's work involved implementing inheritance and polymorphism, allowing for a more flexible and scalable system. Her contributions were crucial in establishing a structured and organized framework for managing different library items.

Bushra Atiq: Bushra focused on developing the user interface and several utility classes. She created classes like nametaken, LMS, loading, and invalid to handle various user interactions and feedback. These classes are responsible for displaying messages to the user, showing loading screens, and indicating errors such as invalid login attempts or duplicate usernames. Additionally, Bushra implemented the logic for user authentication, which includes the signup and login classes. These classes manage the processes of registering new users and authenticating existing users, ensuring secure access to the system. Her work was instrumental in creating an intuitive and user-friendly interface.

Bushra Ansar: Bushra was responsible for developing the Date class, which is used to handle date-related operations within the system. This class allows for setting and displaying dates, which is essential for tracking due dates for borrowed items and other time-sensitive activities. In addition to the Date class, Bushra implemented various admin actions, enabling administrators to manage library items, modify existing entries, and perform other administrative tasks. Her contributions ensured that the system has robust date handling and comprehensive administrative functionality.

Hareem: Hareem's contributions involved conducting research and implementing user actions. Her research helped inform the design and functionality of the system, ensuring that it meets the needs of its users. Hareem implemented actions that users can perform within the system,

such as viewing available items, adding items to their cart, and managing their borrowed items. Her work focused on enhancing the user experience by providing clear and effective ways for users to interact with the library system.