**Library Management System – Report**

**1. Introduction**

The **Library Management System (LMS)** is a simple C++ application designed to manage books in a library. It allows users to perform various operations such as adding books, displaying the list of books, searching for books, issuing books, and returning them. This system is implemented using **file handling** to persist book data between sessions, making it simple yet efficient for small-scale library management.

**2. Objectives**

The objective of this project is to develop a library management system with the following goals:

* Provide a **centralized platform** for managing books.
* Enable basic **CRUD operations** (Create, Read, Update, Delete) for books.
* Allow **book issue and return functionality** for library users.
* Use **file handling** to store and retrieve book data.
* Offer an easy-to-use **command-line interface** for interacting with the system.

**3. Features**

The system provides the following features:

* **Add Books**: Users can add books to the library by specifying the book's ID, title, and author.
* **Display Books**: Users can view a list of all books available in the library, along with their current status (Available/Issued).
* **Issue Books**: Users can issue books by providing the book ID, and the system will mark the book as issued.
* **Return Books**: Users can return issued books, and the system will mark the book as available.
* **Search Books**: Users can search for books based on their title or author.
* **Data Persistence**: All books' data is stored in a text file (library\_data.txt), ensuring that the information is retained between program executions.

**4. System Design**

The system is built around a Library class, which is responsible for managing the list of books and handling operations like adding, displaying, issuing, and returning books.

* **Book Structure**: Each book is represented as a Book structure, which contains:
  + id: A unique identifier for each book.
  + title: The title of the book.
  + author: The author of the book.
  + isIssued: A boolean flag indicating whether the book is currently issued.
* **Library Class**: The Library class manages a collection of books in a vector. The class includes methods for loading books from a file, saving books back to the file, adding books, displaying books, issuing books, returning books, and searching books.
* **File Handling**: The Library class uses ifstream to read book data from a file (library\_data.txt) and ofstream to write data back to the file.

**5. Data Flow**

* **Adding Books**: The user is prompted to enter the ID, title, and author of the book. The book is then added to the books vector and saved to the file.
* **Displaying Books**: The system displays a list of all books, showing their ID, title, author, and current status.
* **Issuing Books**: When a book is issued, the isIssued flag is set to true, and the book’s status is updated in the file.
* **Returning Books**: When a book is returned, the isIssued flag is set to false, and the file is updated accordingly.
* **Searching Books**: The system allows the user to search for books by entering keywords that match either the title or the author.

**6. Code Explanation**

The following key components of the code are essential to its functionality:

1. **Book Structure**:
   * This structure defines the attributes of a book and includes the display() method to show the details of the book in a user-friendly format.
2. **Library Class**:
   * The Library class manages a vector of Book objects and provides methods to handle books.
   * The loadBooks() method reads data from the library\_data.txt file and loads the books into the vector.
   * The saveBooks() method writes the updated list of books back to the file.
3. **Main Function**:
   * The main function creates an instance of the Library class and continuously presents the user with a menu to perform various operations.

**7. User Interface**

The system operates through a simple **command-line interface (CLI)**. The user is presented with a menu containing the following options:

1. Add a new book.
2. Display all books in the library.
3. Issue a book.
4. Return a book.
5. Search for books by title or author.
6. Exit the system.

**8. Sample Execution**

📚 Library Management System

1. Add Book

2. Display Books

3. Issue Book

4. Return Book

5. Search Book

6. Exit

Enter your choice: 2

Library Books:

ID Title Author Status

--------------------------------------------------------------

1 The Great Gatsby F. Scott Fitzgerald Available

2 To Kill a Mockingbird Harper Lee Issued

3 1984 George Orwell Available

Enter your choice: 5

Enter title or author to search: George Orwell

Search Results:

ID Title Author Status

--------------------------------------------------------------

3 1984 George Orwell Available

**9. Challenges**

While developing this system, the following challenges were encountered:

* **File Handling**: Ensuring the correct format for file reading and writing.
* **User Input Validation**: Handling invalid input from users, such as incorrect book IDs or missing data.
* **Data Persistence**: Ensuring data integrity when loading and saving book information between program executions.

**10. Future Enhancements**

The system can be enhanced in the following ways:

* **User Authentication**: Implement a login system to differentiate between Admins and Users, with different permissions for managing books.
* **Graphical User Interface (GUI)**: Create a GUI using libraries like **Qt** or **Tkinter** to make the system more user-friendly.
* **Database Integration**: Replace file-based storage with a database (e.g., MySQL) to handle a larger volume of books and users.
* **Book Categories**: Add functionality to categorize books (e.g., fiction, non-fiction, science, etc.) for better organization.

**11. Conclusion**

The Library Management System successfully implements basic functionalities to manage a library's collection of books. Through simple operations like adding, displaying, issuing, returning, and searching for books, this system provides a practical solution for small-scale library management. The code is easy to extend with additional features and improvements, making it a solid foundation for future developments.