**Library Management System**

**Problem Statement:** Library Management System Using Files and Collections

**Objective**

Develop a console-based **Library Management System** in C# that utilizes file handling and collections to manage book records efficiently. The system should enable users to perform CRUD (Create, Read, Update, Delete) operations on book data, store the data persistently in a text file, and handle potential errors gracefully. This application is designed for beginner to intermediate learners to demonstrate practical applications of file handling, collections, and exception handling in C#.

**Requirements**

**1. Data Structure**

* Each book record should include:
  + **ID**: A unique integer identifier.
  + **Title**: The book’s title (string).
  + **Author**: The author’s name (string).
  + **Publication Year**: The year the book was published (integer).
* Use a List<Book> collection to store book records in memory.

**2. File Handling**

* Store book records in a text file (e.g., books.txt) in CSV format (e.g., ID,Title,Author,PublicationYear).
* Implement functionality to:
  + **Read** book data from the file into a collection when the program starts.
  + **Write** the collection back to the file after any modification (add, update, delete).
* Ensure the file is created if it does not exist and handle file access permissions.

**3. Functionality**

* **Add Book**: Allow users to input a new book’s details (ID, Title, Author, Publication Year) and add them to the collection and file. Prevent duplicate book IDs.
* **View All Books**: Display all book records stored in the collection.
* **Update Book**: Allow users to update the Title, Author, and Publication Year of an existing book by providing its ID.
* **Delete Book**: Allow users to delete a book record by providing its ID.
* **Exit**: Terminate the program gracefully.

**4. User Interface**

* Provide a simple console-based menu to select operations (e.g., 1 for Add, 2 for View, etc.).
* Prompt users for input and validate it (e.g., ensure ID and Publication Year are numeric).

**5. Exception Handling**

* Handle file-related errors (e.g., FileNotFoundException, IOException, UnauthorizedAccessException) when reading or writing to the file.
* Handle input errors (e.g., FormatException) for invalid user inputs (non-numeric ID or Publication Year).
* Provide meaningful error messages to guide the user.

**6. Constraints**

* The system should prevent duplicate book IDs.
* Input validation should ensure that:
  + ID and Publication Year are valid integers.
  + Title and Author are non-empty strings.
* The system should be robust against file access issues (e.g., locked files or lack of permissions).
* The program should maintain data consistency between the in-memory collection and the file.

**7. Learning Objectives**

* Demonstrate file handling operations (reading, writing, and appending) using the System.IO namespace.
* Utilize collections (List<T>) for in-memory data management and LINQ for querying (e.g., finding books by ID).
* Implement robust exception handling for file operations and user inputs.
* Create a modular design with separate classes for book data (Book) and management logic (BookManager).

**Sample File Format (books.txt)**

1,The Great Gatsby,F. Scott Fitzgerald,1925

2,1984,George Orwell,1949

3,To Kill a Mockingbird,Harper Lee,1960

**Expected Output**

Library Management System

1. Add Book

2. View All Books

3. Update Book

4. Delete Book

5. Exit

Select an option (1-5):

**Constraints for Implementation**

* Use only standard .NET libraries (no external dependencies).
* Ensure the code is beginner-friendly with clear comments and modular structure.
* Handle edge cases, such as an empty file, invalid input, or missing file permissions.