**Patient Management System**

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

**Objective**

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

**Requirements**

**1. Data Structure**

* Each patient record should include:
  + **ID**: A unique integer identifier.
  + **Name**: The patient's full name (string).
  + **Age**: The patient's age (integer).
  + **Diagnosis**: A brief description of the patient's medical condition (string).
* Use a List<Patient> collection to store patient records in memory.

**2. File Handling**

* Store patient records in a text file (e.g., patients.txt) in CSV format (e.g., ID,Name,Age,Diagnosis).
* Implement functionality to:
  + **Read** patient 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 Patient**: Allow users to input a new patient’s details (ID, Name, Age, Diagnosis) and add them to the collection and file. Prevent duplicate patient IDs.
* **View All Patients**: Display all patient records stored in the collection.
* **Update Patient**: Allow users to update the Name, Age, and Diagnosis of an existing patient by providing their ID.
* **Delete Patient**: Allow users to delete a patient record by providing their 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 Age 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 Age).
* Provide meaningful error messages to guide the user.

**6. Constraints**

* The system should prevent duplicate patient IDs.
* Input validation should ensure that:
  + ID and Age are valid integers.
  + Name and Diagnosis 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 patients by ID).
* Implement robust exception handling for file operations and user inputs.
* Create a modular design with separate classes for patient data (Patient) and management logic (PatientManager).

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

1,John Doe,30,Flu

2,Jane Smith,25,Allergy

3,Bob Wilson,45,Hypertension

**Expected Output**

Patient Management System

1. Add Patient

2. View All Patients

3. Update Patient

4. Delete Patient

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.