Appointment Management Console Application Design

Objective:
Design a simple console-based application to manage appointments for a cardiologist. This application will be used internally by the clinic staff to efficiently add and search appointments.

Functional Requirements

1. Add Appointment

- **Input:**
    - Patient Name (string)
    - Patient Age (int)
    - Appointment Date and Time (DateTime)
    - Reason for Visit (string)
- **Output:**
    - Return an appointment ID if the appointment is successfully added.

2. Search Appointments

- **Input:**
  A search model with optional filters:
    - Patient Name (string, optional, contains search)
    - Appointment Date (DateTime, optional, exact match)
    - Patient Age Range (Range<int>, optional)
- **Output:**
    - Return a list of appointments matching the filters.

Data Model

- **Appointment Class**

```
public class Appointment
{
   public int Id { get; set; }
   public string PatientName { get; set; } = string.Empty;
   public int PatientAge { get; set; }
   public DateTime AppointmentDate { get; set; }
   public string Reason { get; set; } = string.Empty;
}
```

- **AppointmentSearchModel Class**

```
public class AppointmentSearchModel
{
   public string? PatientName { get; set; }
   public DateTime? AppointmentDate { get; set; }
   public Range<int>? AgeRange { get; set; }
}
```

Implementation Requirements

- Implement an AppointmentService class that provides:
    - A method to **add** new appointments.
    - A method to **search** appointments using **filter chaining** by:
        - Patient Name (SearchByName)
        - Appointment Date (SearchByDate)
        - Patient Age Range (SearchByAge)
- Each search filter method should:
    - Handle null or empty filters gracefully (i.e., skip the filter if not provided).
- Implement **exception handling** to catch and handle errors during add or search operations without crashing the app.

Additional Notes

- Use an interface-based repository pattern to abstract data storage.
- The console app should be user-friendly and robust for clinic staff usage.
- Focus on clean code practices and modularity.