Problem Statement: Car Auction Management System

You are tasked with implementing a simple Car Auction Management System. The system should handle different types of vehicles: Sedans, SUVs, Hatchbacks and Trucks.
Each of these types has different attributes:

- Hatchback: Number of doors, manufacturer, model, year, and starting bid.
- Sudan: Number of doors, manufacturer, model, year, and starting bid.
- SUV: Number of seats, manufacturer, model, year, and starting bid.
- Truck: Load capacity, manufacturer, model, year, and starting bid.

The system should allow users to:

1. Add vehicles to the auction inventory. Each vehicle has a type (Sedan, SUV, or Truck), a unique identifier, and respective attributes based on its type.
2. Search for vehicles by type, manufacturer, model, or year. The search should return all available vehicles that match the search criteria.
3. Start and close auctions for vehicles. Only one auction can be active for a vehicle at a time. Users should be able to place bids on the vehicle during an active auction.
4. Implement error handling for the following scenarios:
    a) When adding a vehicle, ensure that the unique identifier is not already in use by another vehicle in the inventory. Raise an appropriate error or exception if there's a duplicate identifier.
    b) When starting an auction, verify that the vehicle exists in the inventory and is not already in an active auction. Raise an error if the vehicle does not exist or if it's already in an auction.
    c) When placing a bid, validate that the auction for the given vehicle is active and that the bid amount is greater than the current highest bid. Raise an error if the auction is not active or if the bid amount is invalid.
    d) Handle any other potential error scenarios and edge cases that you identify during your implementation. Consider cases like invalid inputs, out-of-range values, or unexpected behaviour.

Your task is to design a C# solution that uses object-oriented design principles to model this system with the appropriate tests. There is no requirement for any UI or database, with the focus being on the structure of the code and the quality of the tests.

Your solution should include:

- Definition of the classes and their properties and methods.
- Implement the auction management operations (add vehicles, search vehicles, start an auction, place a bid, and close the auction).

Pay special attention to edge cases, such as attempting to bid on a vehicle that doesn't have an active auction.
Please ensure that your code is clean and efficient. You should aim for a solution that is easy to understand and modify.

Deliverables

- The source code for the car auction management system includes all necessary classes, interfaces, etc.
- Unit tests for the auction management operations.
- A brief writeup explaining your design decisions and any assumptions you made.