Pet Adoption Center Manager — Assignment

📋 Objective
Build a **console-based Pet Adoption Center Manager**. This app lets staff at a pet adoption center **manage pets for adoption** by creating, viewing, updating, and deleting pet records. All data must be stored in a **file** (`pets.txt`) so it persists between program runs.

📚 Instructions
* Build a **console application** using your language of choice (**Node.js** or **C#**).
* Use **file I/O** to persist data in a simple text-based format.
* Handle invalid input gracefully — do not crash.
* Data file: **pets.txt**. Start empty if no file exists yet.

Run the app using your usual method:
* For Node.js: `node app.js`
* For C#: compile and run your `.exe`

🖥️ Menu Design & Flow
When the app starts, show a main menu like this:

```
==============================
 Pet Adoption Center Manager
==============================
1. Add new pet
2. View all pets
3. Update pet information
4. Remove pet
5. Exit
==============================
Select an option (1-5):
```

If the user enters an invalid option, display:
```
Invalid option. Please enter a number between 1 and 5.
```
Then redisplay the main menu.

📑 Pet Data Model
Each pet record in your data file should be stored as a single line with fields separated by pipes (|):

```
1|Buddy|Dog|Golden Retriever|3|Available
2|Whiskers|Cat|Persian|2|Available
3|Fluffy|Rabbit|Angora|1|Adopted
4|Rex|Dog|German Shepherd|5|Removed
```

Fields (in order):
* **id** — Unique integer ID (incremental, starting from 1)
* **name** — Pet's name
* **species** — e.g., Dog, Cat, Rabbit
* **breed** — Breed of the pet
* **age** — Age in years
* **status** — Available / Adopted / Removed

📝 Menu Options

1️⃣ Add New Pet
Prompt the user for:
* Pet name
* Species
* Breed
* Age (validate it's a positive number)

Set `status` to `Available` by default.
Assign a unique `id` (find the highest existing ID + 1).
Save to file and display:
```
Pet added successfully with ID: X
```
Then return to main menu.

2️⃣ View All Pets
Load and display all pets in a nice table format:
```
ID  Name      Species  Breed            Age  Status
------------------------------------------------------------
1   Buddy     Dog      Golden Retriever  3    Available
2   Whiskers  Cat      Persian           2    Available
3   Fluffy    Rabbit   Angora            1    Adopted
```

If no pets exist:
```
No pets found.
```

Bonus: Filter to show only "Available" pets in the view.

3️⃣ Update Pet Information
Ask user for Pet ID to update.
If found, prompt for each field and allow editing. (Press Enter to keep old value.)
Save changes to file and display:
```
Pet information updated successfully.
```

If not found:
```
Pet with ID X not found.
```

4️⃣ Remove Pet
Ask user for Pet ID.
If found, set `status` to `Removed` and save. Display:
```
Pet removed from the system.
```

If not found:
```
Pet with ID X not found.
```

5️⃣ Exit
Display:
```
Goodbye!
```
and exit the program.

🧩 Technical Notes

Text File Format Details
* Each pet occupies one line in the text file
* Fields are separated by the pipe character (|)
* No headers in the file - just data rows
* Empty file means no pets exist
* When updating/deleting, read all lines, modify in memory, then rewrite the entire file

Data Validation
* Ensure age is a positive number

* Handle file I/O errors gracefully
* Validate that pet IDs are numeric when searching/updating
* Treat "Removed" pets as no longer available for adoption

Example File Content (pets.txt):

```
1|Buddy|Dog|Golden Retriever|3|Available
2|Whiskers|Cat|Persian|2|Available
3|Fluffy|Rabbit|Angora|1|Adopted
4|Rex|Dog|German Shepherd|5|Removed
```

💡 Additional Features (Optional)

* **Search functionality:** Allow staff to search for pets by name or species
* **Adoption process:** Change pet status from "Available" to "Adopted"
* **Statistics:** Show total count of available, adopted, and removed pets
* **Age validation:** Ensure realistic age ranges for different species