**IEnumerable<T> vs ICollection<T> – Simple Differences**

| **Feature** | **IEnumerable<T>** | **ICollection<T>** |
| --- | --- | --- |
| ✅ **Basic Meaning** | Can loop through the items (read-only) | Can loop **and** add/remove items |
| ✅ **Main Use** | Just **reading** the data | **Reading, adding, removing, counting** items |
| ✅ **Has Count Property?** | ❌ No | ✅ Yes (Count property) |
| ✅ **Can Add/Remove Items?** | ❌ No | ✅ Yes (Add(), Remove() methods) |
| ✅ **Inherited From** | IEnumerable | IEnumerable<T> |
| ✅ **Common Examples** | LINQ results, List<T>.Where(...) | List<T>, HashSet<T> |

List<string> names = new List<string> { "John", "Apsara", "Mary" };

IEnumerable<string> enumNames = names; // Only for reading

ICollection<string> collNames = names; // Can read, add, remove

 With enumNames, you can only use foreach to read names.

 With collNames, you can also do:

collNames.Add("Sam");

collNames.Remove("John");

int count = collNames.Count;

Here both enumNames and collNames are pointing to the **same list** (names).  
But the difference is **how much access** they are allowed to have based on the **interface type**.

You have a smartphone (List).  
If you only use it **as a phone (IEnumerable)** → you can only **call** (read).  
If you use it **as a smartphone (ICollection)** → you can **call, install apps, delete things** (read, add, remove).