Vector – The Workhorse List

Principle: Dynamic array stored in contiguous memory (fast iteration/index).
Analogy: Shelf of boxes — grab any instantly.
Use Cases: API data lists, analytics, numeric processing.
When to Use: Whenever order + fast access matter.

Deque – Fast on Both Ends

Principle: O(1) inserts/removes at both front and back.
Analogy: Conveyor belt — can add or remove either side.
Use Cases: Undo/Redo, chat buffers, sliding windows.
When to Use: When both ends are active.

Stack – The Undo Button

Principle: LIFO (last-in, first-out).
Analogy: Stack of plates — last placed, first removed.
Use Cases: Undo, parsing, recursion, backtracking.
When to Use: When you reverse operations or need nested logic.

Queue – The To-Do Line

Principle: FIFO (first-in, first-out).
Analogy: Line of people waiting — fair order.
Use Cases: Job queues, notifications, request pipelines.
When to Use: When fairness or order matters.

PriorityQueue – Urgency Beats Order

Principle: Processes by importance, not arrival time (heap-based).
Analogy: Hospital triage — urgent first.
Use Cases: Schedulers, alerts, AI decision logic.
When to Use: When priority outranks order.

Map – Fast Lookup Dictionary

Principle: Key → value store with O(1) lookup.
Analogy: Contact list — lookup by name, not position.
Use Cases: Session stores, config maps, business rules.
When to Use: When keys define access, not indexes.

Set – Uniqueness Guarantee

Principle: Unique value collection with O(1) membership.
Analogy: Guest list — no duplicate names.
Use Cases: Unique tags, user IDs, visited trackers.

When to Use: When duplicates must be prevented.

Hashable – Custom Identity for Objects

Principle: Define how objects are compared and hashed.
Analogy: Recognize by ID card instead of face.
Use Cases: Object-based caches, de-duplication, repositories.
When to Use: When storing objects in Set or Map.