🆚 Multitasking vs Multithreading

Both **Multitasking** and **Multithreading** allow multiple operations to run **simultaneously**, but they work differently.

Feature	Multitasking	Multithreading
Definition	Running multiple **processes** at the same time	Running multiple **threads** within a single process
Resource Sharing	Each process has **separate memory** and resources	Threads share the **same memory & resources**
Communication	Slow (Inter-process communication needed)	Fast (Threads share data easily)
Execution Speed	Slower (More overhead)	Faster (Less overhead)
CPU Usage	More CPU-intensive	Less CPU usage
Example	Running **Chrome + Spotify + Notepad** at the same time	A **text editor** running **UI updates + Auto-save + Spell-check** simultaneously
Used in	Operating Systems, Multi-app execution	Parallel computing, Real-time applications

🚀 When to Use What?

Use Multitasking When...	Use Multithreading When...
Running multiple independent apps (Chrome, Spotify, VS Code)	Running **multiple tasks inside the same app** (UI updates, Auto-save, File processing)
Requires separate memory allocation	Needs **fast communication & shared memory**
Example: OS scheduling processes	Example: **Game rendering, Real-time apps**

🆚 Process-Based vs Thread-Based Multitasking

Feature	Process-Based Multitasking	Thread-Based Multitasking
Definition	Multiple programs running separately.	Multiple threads running within a single program.
Memory Usage	High (each process has separate memory).	Low (threads share memory).
Communication	Difficult (Inter-Process Communication needed).	Easy (Threads share data).
Speed	Slower (Heavy context switching).	Faster (Lightweight context switching).
Example	Running Chrome, Spotify, VS Code together.	A single app handling UI updates, file saving, and processing together.