Parallel Programming vs. Concurrency

- **Concurrency**: Handling multiple tasks during the same time period by switching between them, even if they run on a single processor. The focus is on managing many tasks at once.
- **Parallelism**: Executing multiple tasks at the exact same time using multiple cores or processors. The focus is on doing many things simultaneously.
- **Key Difference**: Concurrency is about *dealing with* many things at once, while parallelism is about *doing* many things at once.

Unit Testing and Test-Driven Development (TDD)

- **Unit Testing**: Testing small, individual parts of the code (such as a method or a class) in isolation to verify that they work correctly.
- **TDD (Test-Driven Development)**: A development process where you write the tests first, then write the minimal code to make the tests pass, and finally refactor the code. The cycle is: **Red (failing test) → Green (passing test) → Refactor (improve code)**.
- **Key Difference**: Unit testing focuses on checking the code after it is written, while TDD integrates testing as a design step before and during coding.

Asynchronous Programming with async and await

- **Asynchronous Programming**: Improves efficiency and responsiveness by running long operations (like I/O or network calls) without blocking the rest of the program.
- **async**: A keyword used to declare a method as asynchronous.
- **await**: Used inside an async method to pause execution until a task completes, then continue from the same point without blocking the main thread.
- **Result**: Cleaner and more readable code that behaves like synchronous code, but achieves better performance and user experience.
