Lecture - 2

Gemini Code Assist Course

→ Versions of Gemini Code Assist
→ IDE Assistant
→ CLI Assistant
→ Example walkthrough
→ Installing Gemini Code Assist CLI

Two main versions of Gemini Code Assist

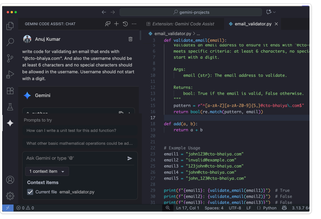
- IDE Assistant → Works inside your editor for in-file tasks

- CLI Assistant → Runs in the terminal for automations and project-wide jobs {add auth with jwt}

- Choosing the right one depends on whether you need real-time help or delegated execution

IDE Assistant (In-editor partner) -

- Lives directly inside your IDE (VS Code, etc)
- Helps without breaking focus (suggestions, fixes and code appear right where you are working)
- feels like a pair programmer sitting next to you
  - * Writing or refactoring code inside a single file
  - * Generating tests/docs for a specific function
  - * fixing syntax issues or small bugs flagged by the IDE



CLI Assistant (Automation Agent) -

- for trigger, automated, system-level tasks
- Works like a delegate - You give the goal, it figures out the steps

> Agent - A Smart intern that never sleeps
> < observes, thinks, and acts in software environments to accomplish tasks on your behalf >

Gemini CLI
- Autonomous
- System-aware (Can run shell commands, read-write files, use external tools)
- Goal-Oriented

- * automating scaffolding or codebase-wide refactors
- * Commands
- * Summarizing or analyzing file directories

Example (IDE + CLI) -

- A new feature module (CLI) → scaffold multiple files (Controllers, services and routes) in one command
- Business logic (IDE) → Switch to IDE for writing function details, refine the code, add inline tests
- Documentation & Configs → CLI to automatically generate API Docs, README, environment configs

→ Installing Gemini Code Assist CLI

- node version (v20+)
- npm install -g @google/gemini-cli

https://github.com/google-gemini/gemini-cli

- gemini <in terminal>