Here's a **self-study roadmap** you can follow to tackle **Clean Code**, **SOLID principles**, and **Design Patterns** systematically. Each day focuses on incremental learning to prevent overload. Adjust it as needed to fit your pace.

Week 1: Introduction to Clean Code

Day 1:

- Read about the importance of clean code.
- Watch a short video or read an article summarizing *"Clean Code"* by Robert C. Martin.

Day 2:

- Study naming conventions for variables, functions, and classes.
- Practice refactoring a piece of messy code.

Day 3:

- Learn about function composition: size, single responsibility, and readability.
- Refactor a function in one of your projects to make it cleaner.

Day 4:

- Explore the concept of comments: when to use and when to avoid them.
- Identify unnecessary comments in a past project and remove them.

Day 5:

- Learn about code smells and their solutions.
- Identify code smells in an old project or open-source code.

Day 6-7:

- Write a small project or program focusing on the principles learned so far.
- Review your code and refactor it for clarity.

Week 2: SOLID Principles

Day 1:

- Overview of SOLID principles: what they are and why they matter.
- Focus on the **Single Responsibility Principle (SRP)**.
- Refactor a class to align with SRP.

Day 2:

- Study the **Open-Closed Principle (OCP)**.
- Understand how to extend functionality without modifying existing code.
- Refactor or write an example implementing OCP.

Day 3:

- Learn the **Liskov Substitution Principle (LSP)**.
- Explore how to ensure derived classes work seamlessly with their base classes.
- Write an example.

Day 4:

- Study the **Interface Segregation Principle (ISP)**.
- Practice creating focused interfaces.

Day 5:

- Understand the **Dependency Inversion Principle (DIP)**.
- Refactor or write code implementing DIP.

Day 6-7:

- Review all SOLID principles with practical examples.
- Write or refactor a small project applying all principles.

Week 3-4: Design Patterns

Day 1-2:

- Learn about **Creational Patterns**: Singleton, Factory, and Builder.
- Implement examples for each.

Day 3-4:

- Study **Structural Patterns**: Adapter, Decorator, and Composite.
- Write examples demonstrating their usage.

Day 5-6:

- Explore **Behavioral Patterns**: Strategy, Observer, and Command.
- Implement these patterns in a mini-project.

Day 7:

- Review all patterns studied and summarize their use cases.

Continuous Practice

- Each week, dedicate **one day to review** and refactor your projects.
- Share your code on platforms like GitHub or Reddit for feedback.
- As you master these concepts, start integrating them into larger projects.