What is a Design Smell?

A **design smell** is a symptom in the design or architecture of software that may indicate a **deeper problem**. Unlike code smells (which are low-level), design smells affect the **structure** and **relationships** between classes or components in a system.

They don't necessarily break the software, but they **reduce maintainability, scalability, and clarity**, making the system harder to understand or extend.

In our train tickting system

Design Smell: God Class / God Module

In our project, if we have a single class or component (e.g., TicketSystemManager) that handles:

- Booking tickets
- Canceling trips
- Managing user authentication
- Processing payments
- Updating train schedules
- Sending notifications

Then this class is doing **too much**. That's a **God Class** — it has too many responsibilities.

What is the problem in this?

- **Violates the Single Responsibility Principle**
- **Makes the system harder to maintain or test**
- **Increases the risk of bugs when updating a small part**

How to Avoid It?

Apply **Separation of Concerns**:
Break the God Class into smaller, more focused components. For example:

- AuthenticationService: Handles sign-up, login, logout
- TicketBookingService: Manages booking, cancellation, ticket printing
- PaymentService: Handles all payment-related tasks

- AdminPanelService: Deals with station management, schedule updates
- NotificationService: Sends and stores notifications