Object-Oriented Design Practice

Goal:
Work in a group to plan out a small software system.
You'll pick one of the six domains in step 1 below and practice designing it using the ideas we've covered:
- Encapsulation
- Inheritance (is-a relationships)
- Composition (has-a relationships)
- Abstraction & Interfaces

You're not coding this project — you're planning it.
Think like a software architect!

Step 1 — Choose Your Domain

1. Zoo Management System – Track animals, enclosures, feeding schedules, and zookeepers.
2. Library Checkout System – Manage books, patrons, loans, and due dates.
3. Arcade Game Center – Handle players, arcade machines, prizes, and tokens.
4. Car Rental Service – Manage customers, vehicles, and rental agreements.
5. Fantasy RPG Game – Players, monsters, weapons, and quests.
6. Hospital Management System – Patients, doctors, treatments, and appointments.

Domain: _____

Step 2 — Brainstorm the Big Ideas

What does your program do? Who are the main users? What kinds of tasks should it handle?

_____

_____

_____

Step 3 — Identify Your Classes

#	Class Name	What does it represent?

Step 4 — "Is-A" Relationships (Inheritance)

Which classes share common traits and could inherit from a parent class? Example: A Lion is-a Animal.

Parent Class	Subclass(es)	What's shared between them?

Step 5 — "Has-A" Relationships (Composition)

Which classes contain or own others? Example: A Zoo has-a list of Enclosures.

Class	Has-A Relationship (contains)	Description

Step 6 — Abstract Classes & Interfaces

Where could you use an abstract class or interface to represent shared behavior?

Abstract Class / Interface	Who uses or implements it?	What common behavior does it define?

Step 7 — Key Methods & Behaviors

Pick 2–3 of your most important classes and describe what they do. Use short pseudocode or method names.

Class	Example Method	What It Does

Step 8 — Rules of the World (Business Logic)

List 3–5 rules your program should follow.
Examples:
- A car can't be rented if it's already booked.
- A patient can't have overlapping appointments.
- A player can't equip two weapons at once.

_____

_____

Step 9 — Ask the AI About Design Patterns

You don’t need to know design patterns in depth yet! Instead, ask an AI tool like ChatGPT: “Given our system design, what design patterns might fit and why?”
Then, see if you can understand the explanation and find a small example that makes sense for your domain.

Reflection

- How did your team decide what should be abstract or concrete?
- What was the hardest part about finding 'is-a' vs. 'has-a' relationships?
- How did planning make the code easier to imagine?