Design Pattern – Use Case Mapping

# KitHub – Campus Used Equipment Library

This document describes how software design patterns are applied to key use cases in the KitHub system to improve modularity, flexibility, and maintainability.

## Pattern–Use Case Mapping Matrix

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Design Pattern** | **Justification** |
| Register and Log In | Factory Method | Used to instantiate different user types (Student, Admin) through a unified interface. Improves code extensibility. |
| Browse and Search Equipment | Decorator | Enables dynamic composition of search filters (e.g., category, availability) without modifying base code. Enhances extensibility. |
| Borrow Equipment | Strategy | Supports interchangeable borrowing rules based on user roles and conditions. Improves modularity and flexibility. |
| Admin Manages Inventory | Observer | Notifies dependent modules like student views or notification systems when inventory changes occur. |

## Detailed Pattern Application

### 1. Register and Log In – Factory Method

Factory Method pattern allows creating objects without specifying the exact class of the object. In KitHub, it is used to instantiate either StudentUser or AdminUser depending on login credentials. This promotes cleaner code and supports future user roles.

### 2. Browse and Search Equipment – Decorator

The Decorator Pattern allows flexible composition of filtering behaviors. For example, base search logic can be extended with additional filters (by category, by availability, by location) using decorators, without altering the base class.

### 3. Borrow Equipment – Strategy

The Strategy Pattern supports defining different borrowing policies (e.g., by user type or duration). Each policy is encapsulated in a separate strategy class, allowing the borrowing logic to remain clean and adaptable. The Strategy Pattern supports defining different borrowing policies (e.g., by user type or duration). Each policy is encapsulated in a separate strategy class, allowing the borrowing logic to remain clean and adaptable.

### 4. Admin Manages Inventory – Observer

When an admin modifies the equipment inventory, other modules (such as student dashboards and notification panels) should automatically reflect the changes. The Observer Pattern facilitates this by decoupling the inventory management logic from the UI update mechanisms.

## Task Matrix for This Document

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Responsible Member** |
| Pattern Identification | Matched each use case with a suitable design pattern | Betül Biçer |
| Technical Justification | |  | | --- | |  |  |  | | --- | | Wrote explanations for why each pattern was appropriate | | Mehmet Karatekin |
| Document Formatting | Created matrix table and structured the detailed pattern descriptions | Aylin Barutçu |
| Final Consistency Check | Reviewed for clarity, alignment with SRS/use cases, and consistency of roles | İlbey Efe Taşabatlı |