**Submitted by- Simran meena**

**Que5.**

**OBJECTIVE:** identify the design principle that is being violated in relation to the given scenario

**Background**: A good object oriented design not only meets the specified requirements but also addresses implicit requirements. There are five design principles which address most of the implicit requirements:

Software Design Principles:

1 Abstraction: Focus on solving a problem by considering the relevant details and ignoring the irrelevant

2 Encapsulation: Wrapping the internal details, thereby making these details inaccessible. Encapsulation separates interface and implementation, specifying only the public interface to the clients, hiding the details of implementation. 3 Decomposition and Modularization: Dividing the problem into smaller, independent, interactive subtasks for placing different functionalities in different components

4 Coupling & Cohesion: Coupling is the degree to which modules are dependent on each other. Cohesion is the degree to which a module has a single, well defined task or responsibility. A good design is one with loose coupling and strong cohesion.

5 Sufficiency, Completeness and Primitiveness: Design should ensure the completeness and sufficiency with respect to the given specifications in a very simple way as possible.

Problem Description:

Which of the following design principle(s) have been violated in the following scenarios?

1. Abstraction

2. Decomposition and Modularization

3. Coupling & Cohesion

4. Encapsulation

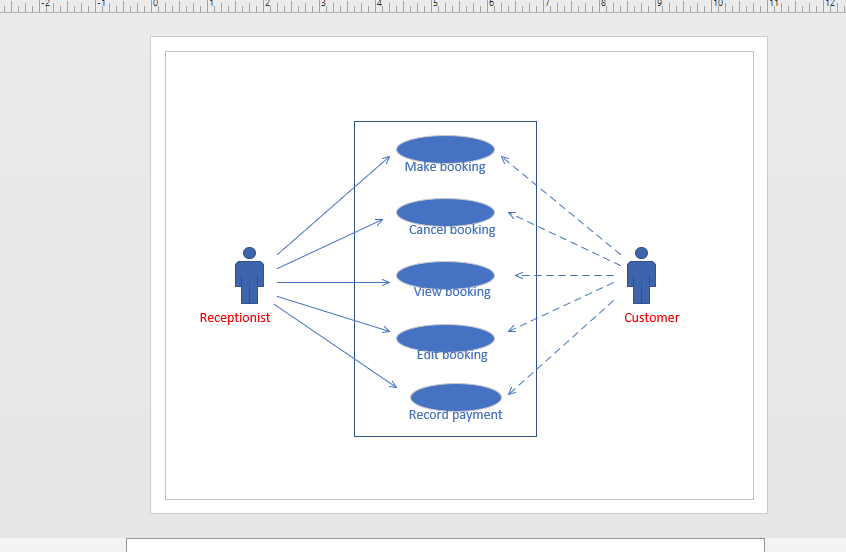
5. Sufficiency, Completeness and Primitiveness

6. All

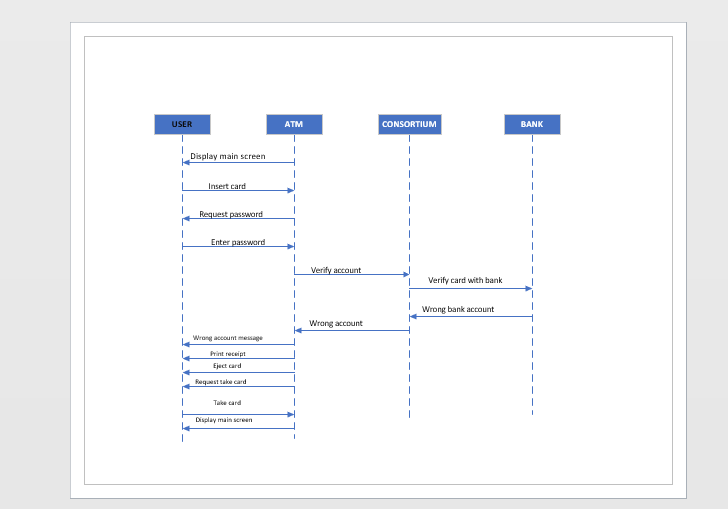
**Summary:** identified the relevance of design principles.

|  |  |  |
| --- | --- | --- |
| **S no.** | **Description** | **Principle being violated** |
| 1 | Important information of a module is directly accessible by other modules. | Encapsulation |
| 2 | Too many global variable in the program after implementing design | Coupling & cohesion, decomposition and modularization |
| 3 | Code breaks in unexpected places | All |
| 4 | Unfulfilled requirements in the code after the design has been implemented | All |
| 5 | Cyclic dependencies among classes | Coupling & cohesion, decomposition and modularization |
| 6 | Huge class doing too many unrelated operations | Abstraction, decomposition and  modularization |
| 7 | Several unrelated functionalities/tasks are carried out by a single module | Decomposition and Modularization, Abstraction |
| 8 | All data of all classes in public | Encapsulation |
| 9 | Design resulting in spaghetti code | Sufficiency, completeness and primitiveness |
| 10 | An algorithm documented as part of design is not understandable by the programmers | Sufficiency, completeness and primitiveness |

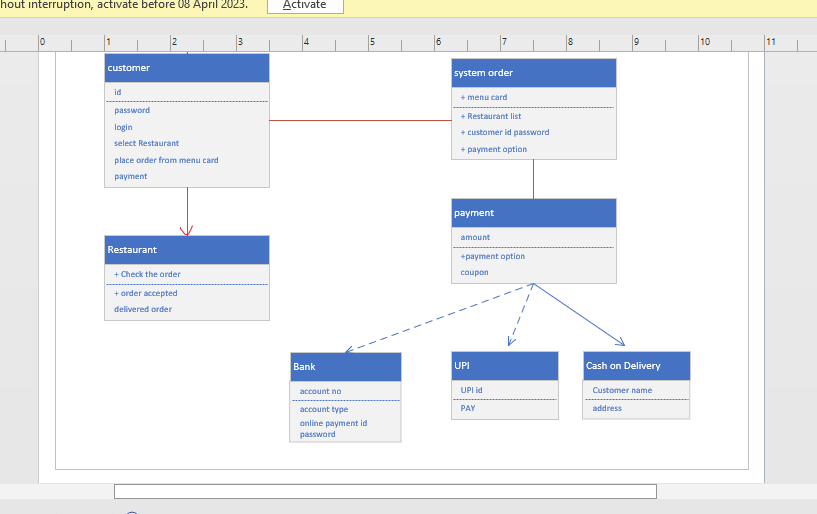
Q1.Implement hotel management system through Use case Diagram.



Q2. Design the Sequence diagram for ATM System taking into consideration different scenarios

. 

Q3. Design the class diagram for online food ordering system.



Q4. Design the activity diagram for student Attendance management system. 