**Report on the Use of Design Concepts and Principles to Evaluate Design and Provide Solutions**

The software is designed according to the MVC model, which helps reduce coupling and increase cohesion among the components of the application.

1. **Single Responsibility Principle**

Most classes adhere to the Single Responsibility Principle.

Some objects that exist only once in a session of the software (Cart, AIMSDB, LOGGER) are designed according to the Singleton design pattern.

DTO classes (OrderDTO) facilitate interaction with the database, clearly separating database logic from the Order object logic.

But there are still some errors that do not comply with this principle:

|  |  |  |
| --- | --- | --- |
| **RELATED MODULE** | **DESCRIPTION** | **IMPROVEMENT** |
| PlaceOrderController.java | The Controller performs multiple tasks simultaneously. | Divide into many smaller controllers such as: validateController, typeCheckController, … |

1. **Open/Closed Principle**

|  |  |  |
| --- | --- | --- |
| **RELATED MODULE** | **DESCRIPTION** | **IMPROVEMENT** |
| getAllMedia(), getMediaById(), updateMediaFieldById() | When SQLite changes the way data is organized, or when retrieving a different entity such as a book or CD, or when modifying or adding new get, update, delete functions, the code in the Media class needs to be changed. | • Solution 1: Define an interface with get, update, delete functions to represent the data retrieval process from the database without needing to know the details of its implementation. Then implement it in the subclasses.  • Solution 2: Turn Media into an abstract class and make the functions interacting with the database abstract functions for subclasses to override. |

1. **Liskov substitution principle**

|  |  |  |
| --- | --- | --- |
| **RELATED MODULE** | **DESCRIPTION** | **IMPROVEMENT** |
| Media.java | The subclasses DVD, CD, and Book inherit from the Media class and override the getAllMedia() method, but they do not return the correct objects. | These methods need to be revised or removed in the subclasses. |

1. **Interface segregation principle**

The ISP principle helps you design flexible and extensible software systems. By separating small and specific interfaces, you can easily add new payment methods without affecting other parts of the system.

|  |  |
| --- | --- |
| **RELATED MODULE** | **DESCRIPTION** |
| IpaymentSubsystem.java | In the future, if we expand the system to have more payment methods  then it will be easy to implement this method |

1. **Dependency inversion principle**

The Dependency Inversion Principle (DIP) focuses on reducing dependencies between modules in a software system. Book, CD, and DVD classes inherit and depend on the media class, instead of allowing the modules to depend directly on each other.