# Analysis and Justification – Design Choices

## MVC Design Patterns.

The MVC design pattern was applied to the Smart Board application. Separate packages were created for the model, view and controller sections. The model package incorporated all the aspects of the data for the application. It dealt with the rules for modifying the data and adding to the data. The view package had classes that helped to display to the user what was inside the actual model. The controller package and its classes can be described as a mediator that responded to action events and allowed for the interactions of the user from the view to affect and update the model (Pratyaksa 2022).

## SOLID Design Principles

The code adhered to SOLID design principles. One of the ways that this is implemented is by following the Open-Closed principle. This means that a class is open for extension but closed for modification. There are two interfaces in the code, one is Deleteable and the other is Quotable. The quotable interface is used by the user and the deleteable interface is extended and used by the workspace, board and column classes. The Interface Segregation principle is in effect with the User and DefaultUser classes. The default user extends the abstract class user and every method inside the user is used by the default user appropriately. There is no unnecessary coupling between the classes and the subtype is quite suitable here for the actual base type.

## Other Design Patterns

Other design patterns have been used throughout the coding of the Smart Board application. The Singleton pattern, which falls under the umbrella of a creational pattern is in place to ensure there is only one user instance at a time. It also allows for the user to be easily accessible (Pratyaksa 2022). This has been created in the User Holder class under a well-known access point in the application package. Another pattern that has been utilised is the Facade pattern which is a structural pattern. This pattern allows for reduced dependencies between members of the system and classes. It ensures that the application is of enhances portability and creates an additional layer in the system (Pratyaksa 2022). This is in place in the DataBaseMaster class. This class incorporates all the database methods and is the master of all the database controls so that outside controllers and classes do not have direct access to the database and must call the master data base class in its stead. It ensures that the other classes are not overly dependent and instead can communicate through the façade.

## References

Pratyaksa D (2022) ‘Further Programming’ [PowerPoint Slides - Model-View-Controller (MVC)], RMIT University, Melbourne.

Pratyaksa D (2022) ‘Further Programming’ [PowerPoint Slides - OO Design Patterns], RMIT University, Melbourne.