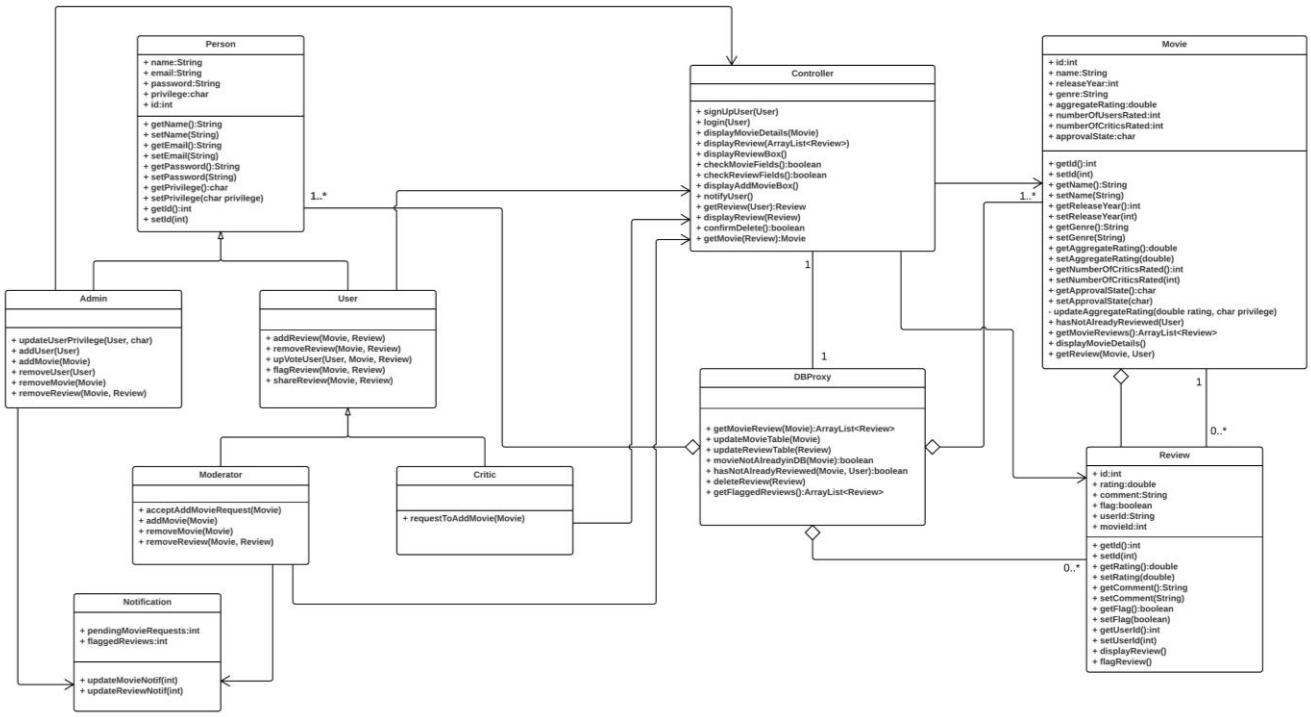
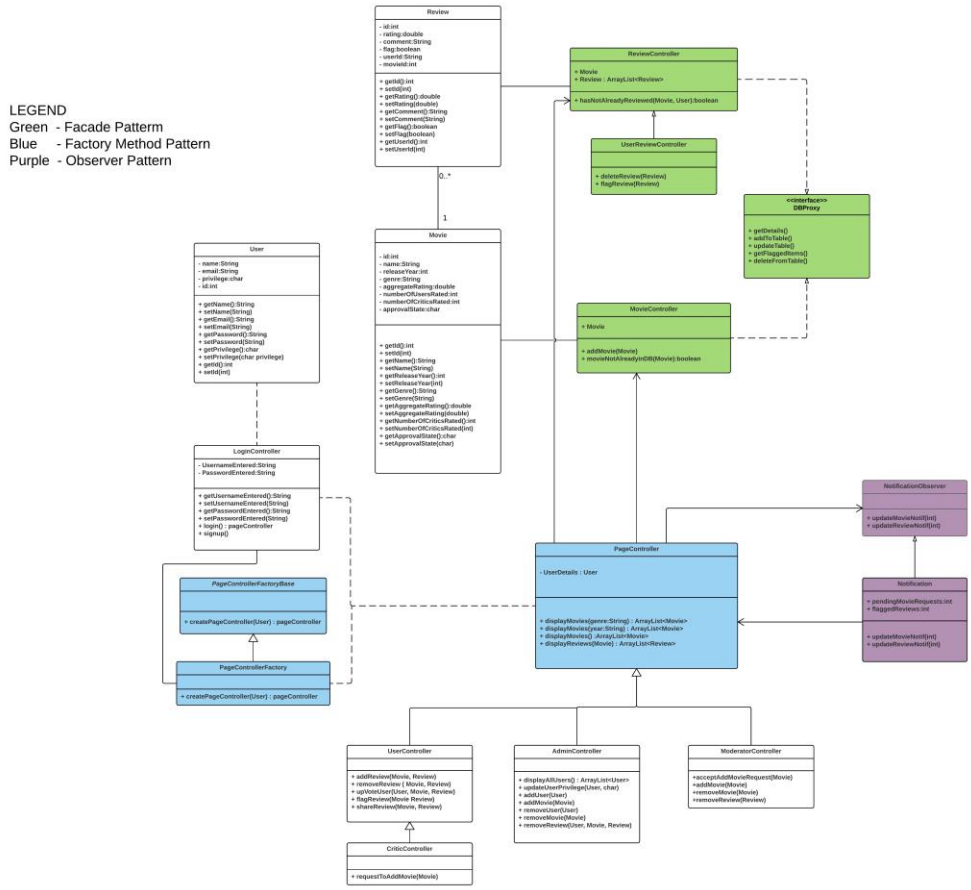
39_Rated-OnlineMovieReviewSystem_ProjectPart3



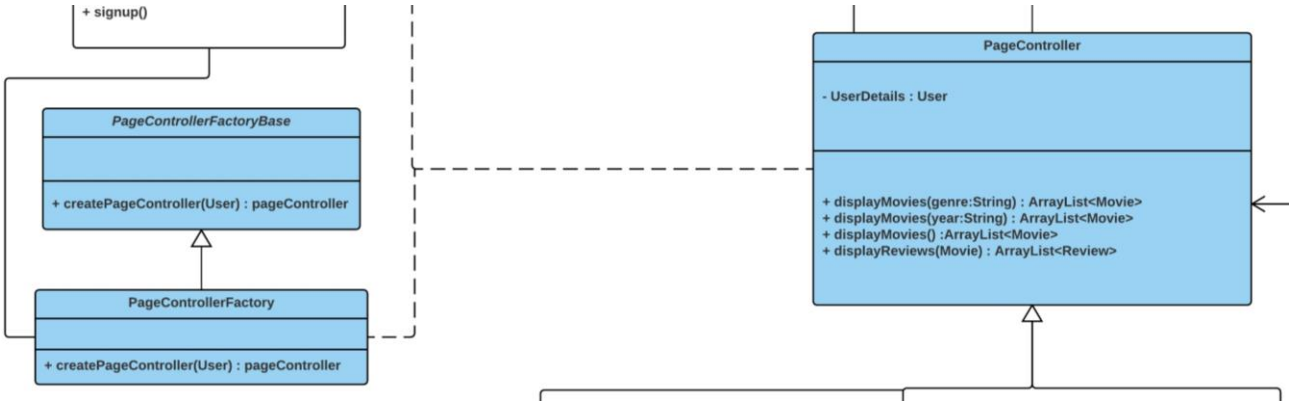
The diagram can be seen here.

The new refactored class diagram is as below.



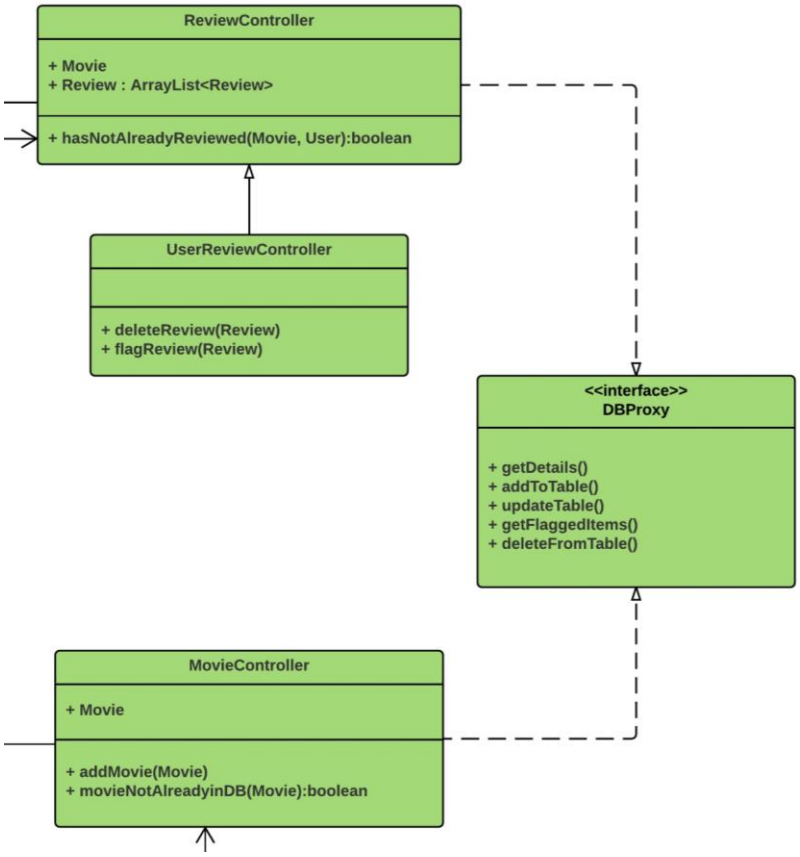
In the first class diagram, we had used just a single controller class that handles all the functionality. This was refactored by delegating the work to different controller class which handle their own responsibility.

A new PageController class was created which would handle the data to be displayed on a page based on the user type. This page controller class along with PageControllerFactoryBase and PageControllerFactory implements the Factory Method design pattern as shown in the figure below. This is because the number of PageController objects is not known before hand.



Since the UserController, AdminController, ModeratorController and CriticController inherit from the PageController, they would be the different products of the PageController.

The DBProxy was changed to an interface through which the user can read data from the database. This implements Façade design pattern as we created a single new interface to access the data than having multiple interfaces. Since Façade design pattern implicitly implements Singleton pattern, the ReviewController and MovieController classes would be singleton classes.



We also modified the notification class to implement the Observer design pattern because whenever there is a notification, all the corresponding classes that are involved must be notified and appropriate changes are to be made. This is shown in the figure below where there is a notification observer which is extended to a notification class which updates the PageController class when a notification occurs.