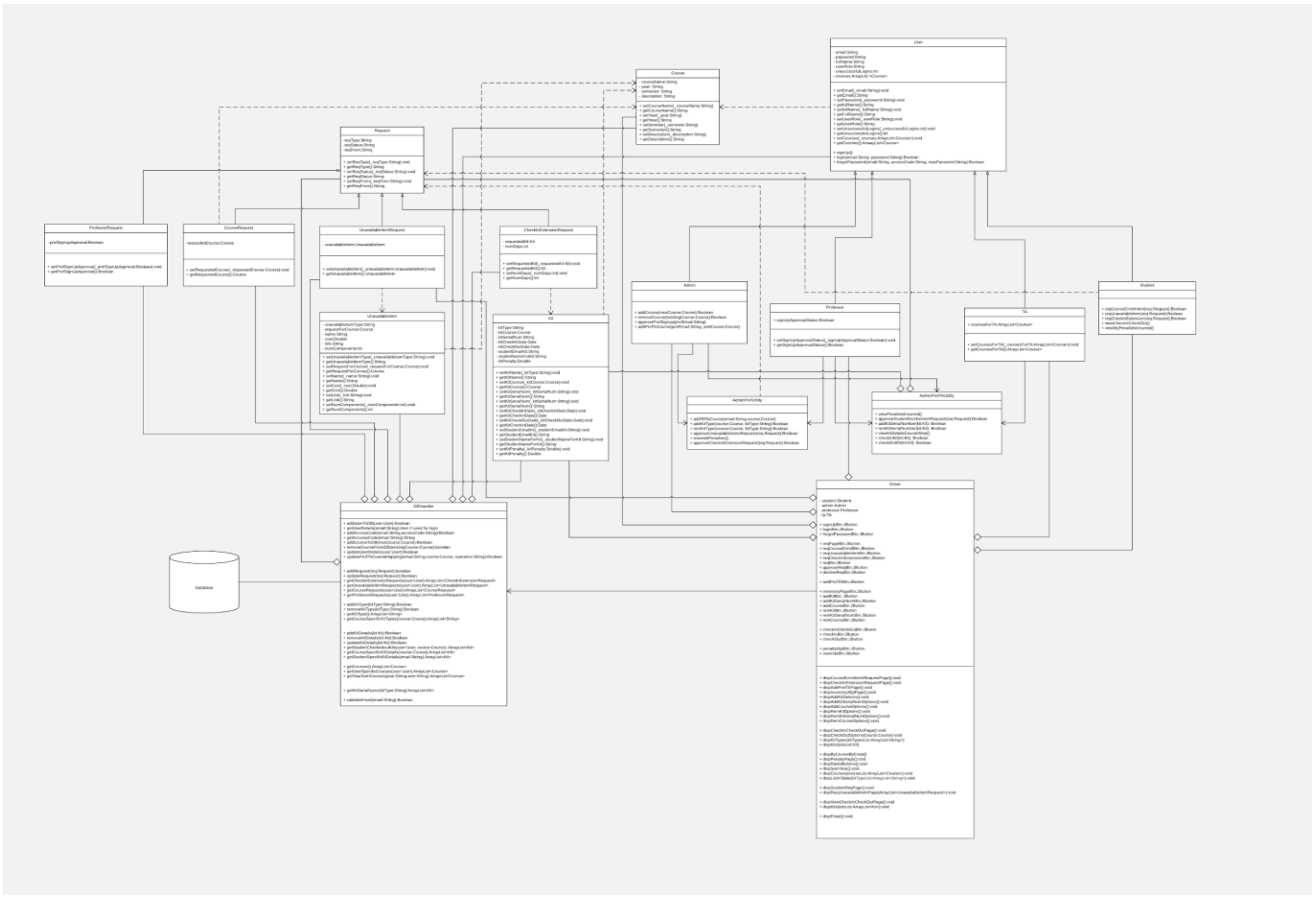
PROJECT PART 3 - REFACTORING

Title: EEZON - An Interactive EE Inventory Management

Team: Sairam Udaya Janardhana Muttavarapu, Chinmay Shah, Rahul Yamasani, Sharath Reddy Vontari

Old Class Diagram:

Good resolution of old class diagram can be found in this link.

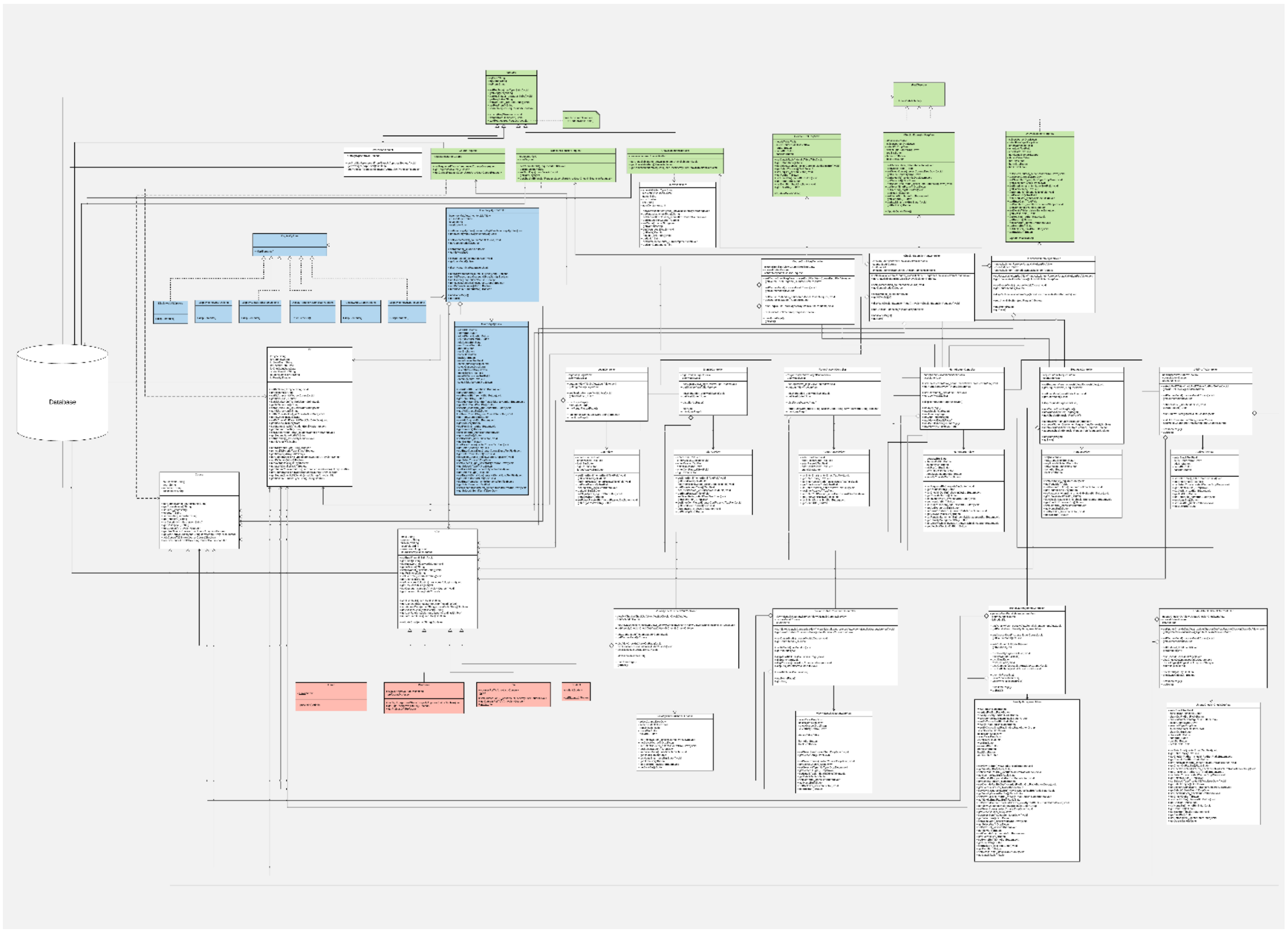


New Class Diagram (with refactoring):
Good resolution of new class diagram can be found in this link.

Three Design Patterns are implemented:

1. Observer Pattern - Green Color
2. Strategy Pattern - Blue Color
3. Singleton Pattern - Red Color

Also MVC Architectural pattern is implemented for all the Views.



Refactoring Changes:

In the initial implementation we had a centralized Driver to control the various actions and a centralized DB handler to access the Database. After discussion with Professor, we have understood that it is a bad architectural practice and such classes are called blob classes. We also removed Utility classes.

Then we went on to make major changes and implemented MVC architectural pattern, because of which we got a more clear class diagram. We implemented the following Design patterns as a part of refactoring:

- **Observer Pattern** (Indicated by Green color in the class Diagram): The ReqObserver class acts as the observer for the Request class and the Request class is the subject. The addRequest() method is implemented by the sub classes of Request class which are CourseRequest, CheckInExtensionRequest, UnavailableItemRequest. When these classes call the addRequest() method, foreach object ‘o’ in the observers o.UpdateDetailsTable() method is called in all the observers. The observer classes implement the UpdateDetailsTable() method according to their need.

- **Strategy Pattern** (Indicated by Blue color in the class Diagram): In general the strategy pattern is implemented between the Controller classes and View classes. Here InventoryMgtController and InventoryMgtView classes are involved in strategy pattern. The DisplayOptions interface is implemented by several other Display classes, which implement the dispElements() method according to their algorithm. For example when addKitBtn:Button is clicked then the dispElements() method inside DisplayAddKitOptions will be executed. Similarly strategy pattern can be implemented to the other controller and view classes in this class diagram.

- **Singleton Pattern** (Indicated by Red color in the class Diagram): This pattern doesn’t allow the creation of more than one instance of a class. So we are using this pattern for the Admin, Professor, TA and Student classes because when a user logs in, only one instance of that user is created.