SMART Design Explanation and Justification

Throughout the development of this project we have encountered various design-level issues which required the usage of various design patterns to solve. One such example is the issue of needing many-to-many relationships in our database, and thus using associative tables to achieve this functionality in a proven and widely recognized manner. In our database design, every time we encounter a many-to-many relationship, we have an associative table to handle that type of entity relationship. Also in our database design, we have made good use of lookup tables in situations where we need a specific list of accepted values, such as options available in dropdown menus. Additionally, all of our application's interactions with the database are designed to be performed via stored procedures, which have multiple advantages over direct queries.

With our database structure there are many places in which we need many-to-many relationships, such as students being in many classes and classes having many students. Many-to-many relationships do not technically exist within SQL relational database management systems, such as MySQL which we are using for this project. The recognized solution to this many-to-many relationship problem is to include an entity, an "associative table", "between" these entities with the many-to-many relationship. This table then stores all of the many-to-many associations between the two other entities, and thus allows us to effectively create many-to-many relationships within our database.

Often within our application we need to reference a set list of values, such as the public school levels of our applicants, the types of accounts the application uses, the subject and subject level of our courses. In these cases, we designed the database of our application to use lookup tables to simply store these values to be referenced in our other database entities. There are various ways to achieve the same functionality, such as hard-coding all the values in our Javascript, but we determined that lookup tables are the most appropriate design solution for our application. Lookup tables give us one location to find all the values we need when using our dropdowns and maintain consistency within our application, rather than, for example, trying to manually match up all of our values between every page and script within the application.

Our application needs to interact with the database often, whether retrieving, storing, or updating data therein. Because of this, we have designed the application to only interact with the database via stored procedures. This design allows us to essentially re-use the same queries from multiple different places within the application and have the same functionality. In that way it makes it easier to implement certain functionalities when we have the fundamental stored procedures to build off of. Stored procedures are also more secure than direct queries from the code in protecting against SQL injection attacks. Because stored procedures are already built in the database with specific parameters they expect, they are less susceptible to SQL injection, although input validation and sanitization is still important. For these reasons, we decided to design our application to make use of stored procedures when interacting with the database.