Team Eh's analysis of Hell's IT Tech Support

Nickolas Schmidt, Quinn Bast, Abdulaziz Alghamdi

Code Cleanliness

Hell's IT Tech support has fairly clean code with only a few exceptions. It was extremely nice to see the use of Object Oriented Programming in PHP through the use of classes in PHP. This makes the implementation of Design Patterns easier to recognize within this team's code. The team also has good use of enumeration types for the 'status' field, as seen in the image below.

```
class eRequestStatus
{
        const APPROVED = "APPROVED";
        const AWAITING_APPROVAL = "AWAITING_APPROVAL";
        const REQUIRE_EDIT = "REQUIRE_EDIT";
        const DENIED = "DENIED";
        const REMOVED = "REMOVED";
}
```

The team's design patterns were Singleton, Composite, and Model. These can be verified through the screenshots below. The singleton was used on the database connection as only one database connection will ever be needed as shown below.

```
final class DB
{
    public $conn = null;

    public static function Instance()
    {
        static $inst = null;
        if ($inst === null)
            $inst = new DB();
        return $inst;
    }
```

Figure 1 Singleton

The Composite design pattern was used in order to display success and failure Modals on the page when specific actions were completed. This can be seen through the javascript below.

```
function getUrlParams( prop ) {
    var params = {};
    var search = decodeURIComponent( window.location.href.slice( window.location.href.indexOf( '?' ) + 1 ) );
    var definitions = search.split( '&' );

    definitions.forEach( function( val, key ) {
        var parts = val.split( '=', 2 );
        params[ parts[ 0 ] ] = parts[ 1 ];
    } );

    return ( prop && prop in params ) ? params[ prop ] : params;
}

$(document).ready(function(){

    if(getUrlParams('s') === 'formfail')
        $('#formFail').modal('show');


});
```

Figure 2 Composite

Finally, Model was used through the use of MySQL in the backend to store the data.

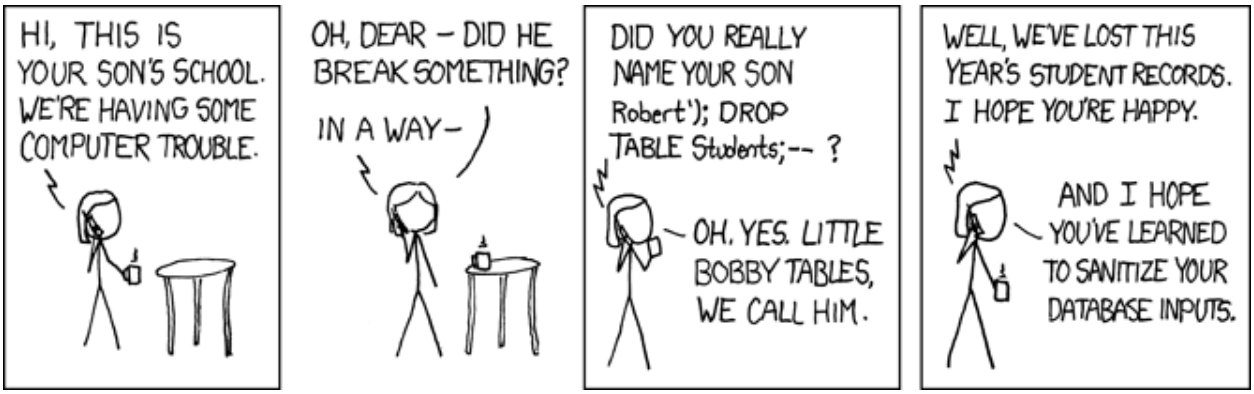
Overall the code for the project was fairly well created.

Code Smells

Although the code was well done, there are some areas where the code has 'smells', majority of which are Dispensibles. The first code smell that was identified was the lack of comments in the code. Because of the lack of comments, some code is harder to follow as it requires travelling to the class functions in order to understand exactly what is happening behind the scenes. This could be done better by having better function names or comments that describe what is happening along the way.

The second code smell that was found is a large amount of copy-pasta or duplicate code. This can be found on lines 254-412 of the manage.php file. This code is exactly the same with the exception of one line in each modal, the text that appears in the message box. These Modals could each be their own class and be instantiated through the use of the factory design pattern.

In addition, it has been determined that when inputting any text into the program, the text is not sanitized. This is extremely dangerous, as our team could drop their tables if I wanted to. Queue relevant xkcd.



Although I wasn't able to cause any damage myself, the inputs are definitely not sanitized, as inputting an ' in any input field will not submit any request. Whereas any query without a ' will successfully be input.

Finally, the last problem that was found was that there are multiple files for "approve", "deny", "edit", "implement", "remove" requests. These files call specific class functions, however, this could be simplified into one file which uses a parameter to tell the file which type of request is needed. This would make the code simpler and put similar functions into one spot. This will also reduce code duplication in these files as the code in each of these files is extremely similar and could be combined.

approve_request.php	Added the manage page with approve, deny, edit, and implement for ana...
deny_request.php	Added the manage page with approve, deny, edit, and implement for ana...
edit_request.php	Added the manage page with approve, deny, edit, and implement for ana...
implement_request.php	Added the manage page with approve, deny, edit, and implement for ana...

Refactoring

When considering to refactor this code, there are some techniques that can be used. The best technique come from the Composing Methods, Extract Method. There is quite a bit of code duplication and these functionalities can be extracted and grouped together in their own classes. This can also be done with the "Replace method with method object", which creates a class for a given method. All code smells that were determined can be refactored using these techniques for refactoring.