Dancing With Zebras Code Review

ENSE 470

Milestone 7

April 10, 2018

ByteThis Software

Taylor Petrychyn, Konstantin Kharitonov, Alwin Baby

1.0 Introduction

The Health and Environment Labs Limited web application is feature rich and works as designed. ByteThis Software was tasked with providing a code review on the codebase. This report will outline our findings including cleanliness, code smells, refactoring suggestions, and other issues we have found.

The application was built with a MySQL database, PHP backend, and Javascript, HTML, and CSS on the frontend. It appears only one external library was used called, “Swift” to manage email notifications.

2.0 Project Structure and Code Style

The project structure appears to be broken down into the main pages/views in the root directory, a “style” folder containing CSS files, a “processing” folder containing the business logic PHP files, and a “javascript” folder containing JS files. The project appears to follow the Model-View-Controller pattern quite well.

The code style is very consistent across the project, following camelCase naming convention in file names, variable names, and CSS classnames. However, the code has inconsistent indentation, this could be an issue from Github, or perhaps multiple developers using tabs vs. spaces. This could easily be fixed in any modern IDE.

3.0 Code Issues

3.1 Repeated Database Connections

One issue found in the code is the repeated opening and closing of connections to the MySQL database. This may have scalability issues when the user base grows as each user session would repeatedly be writing and reading from the database. We recommend using the Singleton Pattern to handle MySQL connection instances. The database connection code is also repeated several times and could be turned into a function, or even a connection class.

3.2 Navigation Bar

The navigation bar code employs the use of the Decorator Pattern to display different links for different user levels. This is a great use of the pattern. However, the navigation bar code is repeated on every page meaning any change to the navigation would require changes to every page. This change preventer could cause issues if the application were expanded. A possible solution would be to use the Composite Pattern and have a reusable Navigation bar component.

3.3 Unclear Parameter Names

When query parameters are used throughout the application, they are usually given short variable names such as, “a’, and “r”. This makes it hard to understand the purpose behind these variables. Specifically, in the submitRequest code, a successful request will return the user to the requestSoftwareTool page, with a parameter, “r=s”. After investigation, we determined this causes an alert banner to display on the page to notify the user of a successful request. We would like to change these to more descriptive names.

3.4 Styling Logic

The following code adds odd and even classnames to table rows so that they may be styled with different background colors:

```
if($isOddRow)
{
        echo '<tr class="oddRow">';
}
else
{
        echo '<tr class="evenRow">';
}
```

Minor improvement could be made by removing this from the logic and instead using the CSS nth-child selector pseudo-class.

3.5 Form Validation

Most of the form validation code is repeated twice, once for individual field onBlur functions, and again when the form is submitted. This is an instance of change preventer and code duplication. This could be refactored so both onBlur and onSubmit events use the same validation functions.

The application forms contain no server-side validation making it easy for a knowledgeable user to bypass the form validation. It also may expose the application to hacking methods such as cross-site scripting and SQL injection.

3.6 Unsafe Font

The use of the Rockwell font causes issues because it is not web-safe. A proposed solution would be to include a font-family to fall back on if the users device does not have the Rockwell font.

4.0 User Experience

4.1 Home Page

With the site on its own, a user is able to login using the credentials that are listed in the ense70.sql file that was included in the milestone 6 folder. Currently, a user is not able to sign up as a new user, so the website uses every approver and analyst as a user. A user is not required to login before accessing the list of software approvers in the system.

4.2 Request Tool Page

The user has access to the request software page and can submit a ticket, also not requiring the user to be logged in, thus not actually needing to be a user in the system previously. This can be very beneficial for analysts as they have a much small list of users with login credentials, minimalize how many users need to be in the system. The downside is that software is much more open, which can result in an overload of users requesting specific software and allows for non legitimate software request to be made. With many requests being made without a venting system, it can lead to some many requests being made that approvers may have a chance to approve a non legitimate software request.

4.3 Approver List Page

The list of approvers does showcase every approver in the system, loading from the ense470.sql file. Unfortunately, the table is not user friendly, as it does not feature a search function. The page also is difficult to distinguish because of the black text portraying on the black background.

4.4 List of Requests Page

This page follows a similar table that the Approver List page, displaying with different values. Each ticket is available to open and it loads a page which will display the option to approve or deny.

5.0 Conclusion

This interpretation of the HELL software is an excellent representation of the functional specifications that was set out in the functional requirements. For the MVP release, each function that was originally said to be implemented in milestone 5.