Independent study report

* Rohan D. Shah
* Supervised By: Dr. Amanda L. Hughes

Contents

[Problem statement: 4](#_Toc395484373)

[Laying the foundation: 5](#_Toc395484374)

[Taking the first steps: 6](#_Toc395484375)

[Listing down the bugs: 7](#_Toc395484376)

[Solving the bugs: 8](#_Toc395484377)

[Remember me: 8](#_Toc395484378)

[Associating the event-creator with the event: 8](#_Toc395484379)

[1) No user interface for the admin to see the existing users, events and 8](#_Toc395484380)

[Adding users to the events: 8](#_Toc395484381)

[Showing only those users who are not yet added to the event: 9](#_Toc395484382)

[Displaying IDs instead of words: 9](#_Toc395484383)

[Cannot delete category in the archive mode: 9](#_Toc395484384)

[Dialog box in streaming mode not displayed properly: 9](#_Toc395484385)

[Change of event name was not reflected in the category name: 9](#_Toc395484386)

[Role based authorization control: 9](#_Toc395484387)

[RBAC hierarchy 10](#_Toc395484388)

[Legend of the hierarchy nodes : 10](#_Toc395484389)

[Inconsistency in database on deletion: 11](#_Toc395484390)

[Stumbling blocks and challenges faced: 12](#_Toc395484391)

[Lessons learnt: 13](#_Toc395484392)

[Future Works: 14](#_Toc395484393)

[Final thoughts: 15](#_Toc395484394)

# Problem statement:

To improve the existing functionality of the “public information officer (PIO) monitoring application (PMA)” by solving the existing bugs in the application and adding more features to it, so that it could be tested on site with real users and can be deployed in an environment where its ability to connect the PIOs with social networks can be unleashed.

# Laying the foundation:

The application is developed using Yii framework in PHP. Being new to the framework I had to first read about it and learn the framework by going through various tutorials. The framework uses MVC architecture and everything is nicely separated into well defined classes. I learned the framework by going through these tutorials - 1) developing web applications using the Yii PHP framework by Larry Ullman 2) Web application development with Yii and PHP by Jeffery Winesett. After getting enough confidence and knowledge about Yii, I moved further with exploring the application.

# Taking the first steps:

To know about the application and how data was stored and accessed, I read Dr. Amanda L. Hughes’ dissertation. The dissertation was very clear and gave enough information about how the data was organized and stored in the database. I got a clear idea about how the application was working at the back end. The dissertation also listed some of the features to be implemented. After reading about the architecture of the application, I wanted to see how the application actually worked, as a user. So, I ran the application as a normal user to see what was missing. This recce helped me to figure out the bugs in the software application which I needed to solve.

# Listing down the bugs:

After using the application a couple of times, and applying various combinations of inputs to test various features, I finally came up with a list of things that were to be improved and had to be taken care of and they are as follows:

1. “Remember me” feature on the login screen was not working appropriately.
2. The user was not able to delete a category in the archive mode.
3. When deleting a category in the streaming mode, the “yes” and “no” buttons were not visible in the dialog box and the dialog box was in the far left corner of the screen which was not very user friendly.
4. There were some pages where feedback to the user was necessary but missing.
5. When a new category is created in the streaming or archive mode, the user needs to refresh the page to view the new category.
6. The user was not able to add users to an event in the “Event Access” page.
7. In the Event Access page, when one wanted to add a user to the current event, all users were shown instead of just those who are not a part of the current event.
8. There is not much distinction between a normal user and admin and if there is, then on what basis the distinction is made is yet to be found out.
9. Whenever a user created an event, the creator was not associated with event even though he had created it.
10. There was no explicit link to manage users, events and user events.
11. While viewing the user events, user Id and event id was shown which makes no sense to the user reading it. Event name and user name would have been a better combination.
12. Chances of inconsistencies in the database upon deletion of some items. Referential integrity has to be checked.

These are some of the major bugs and things that needed to be improved among many others as they hindered functionality and user experience.

# Solving the bugs:

I tackled bugs one by one and segregated it on the basis of how difficult would it be to solve them on face value. I started with the easiest one and slowly moved on to the more difficult ones as I gained more confidence in Yii.

## Remember me:

I thought this was the simplest and hence I started with this one. Since authentication and cookie management was done by Yii framework I speculated that this feature was probably not working because cookies are not enabled. Hence, I located the file in which cookies were handled by the Yii and enabled and it worked fine later on. Though this work seemed easy but took a lot of time to figure out the problem and finally solve it.

## Associating the event-creator with the event:

One of the many good features of this application is that a user can be associated with the event by the admin or the creator of that event. Now it should be obvious that the creator of the event should be the first person to be associated with the event. But it was not and it was a bug. I dug into the code and started to view the respective model and controller files. Finally I figured out what the problem was and wrote some lines of code which achieved the desired functionality. These lines were written in the actionCreate() of the event and I only had to make a new userEvent model which had the creator as a part of the event.

1. No user interface for the admin to see the existing users, events and users associated with various events:

This was a fairly easy task as I had to make a page consisting of links to users, events and user events. From here the admin could perform CRUD operations on those tables. Using RBAC, this page will be visible only to the admin users. It has some descriptions about the existing links and can be used as a “Master GUI portal” to manage the entire application.

## Adding users to the events:

In the event access page, the creator of the event had the power to add users to his event so that the users can work on it in collaboration. However, there was a bug that the users were not added. After going through various files of models and controllers, and using various debugging technique, I was able to locate the error. Hence I wrote code that now accomplishes what it is suppose to. The problem was that the IDs of the users were not passed to the function that took care of adding the users. I made changes in actionReturnForm() of User controller. I wrote code which found all the users and returned it back to where it was needed. Using the userIds of the returned array of users, new UserEvent can be created which solved this issue.

## Showing only those users who are not yet added to the event:

After accomplishing the above task, I noticed a new bug. The users who were previously added to the event were also shown when the creator wanted to add new users to his event. I wrote some code which showed only those users who were not a part of that particular event. This code is doing nothing but filtering.

## Displaying IDs instead of words:

When the admin wanted to see all the users associated with events, the respective IDs were displayed; this makes no sense to the reader. These IDs are used by the database for referential integrity. So I wanted to change that by displaying actual event names and user names. This task looked easier than it was as it is easy to write a normal SQL query but to accomplish the same using the powers of the framework was difficult. But at the end it was accomplished.

## Cannot delete category in the archive mode:

This was one tricky bug. Though the code in the archive mode to delete a non-root category was same as the one in the streaming mode, yet it was behaving completely different than it. After much inspection I found out what was going wrong. Instead of correcting the code, I decided to change the code altogether without affecting the functionality and it worked. The previous code was written in a combination of jquery and ajax. I simplified it and write in simple javascript and php. The changes were made in /views/site/coding.php file.

## Dialog box in streaming mode not displayed properly:

In the streaming mode the user was able to delete a non-root category but the confirmation box was not clear. It got displayed on the top-left corner of the screen which was not user friendly and the text on the buttons was not visible at all. The buttons were implemented in jquery which I corrected and made them a javascript implementation.

## Change of event name was not reflected in the category name:

The name of the root category comes from the name of the event. But when you change the name of the event, the category root name remains the same. This was the bug and needed to be handled. I wrote a code which now displays the current event’s updated name.

## Role based authorization control:

At the start of this report it was mentioned that the application did not have any mechanism to differentiate between the users, authors and administrators. There was no power distribution among the kind of users. This is where, power of the Yii framework comes into picture.

# RBAC hierarchy

## Legend of the hierarchy nodes :

|  |  |
| --- | --- |
|  | Roles |
|  | Authorized to do these functions |

The RBAC is designed as per the above hierarchy. The hierarchy consists of roles and functions which users can perform. A user can perform all the functions listed as its nodes. A user cannot perform his parents’ functions. Parent inherits all the functions of all his children. To achieve this functionality, three new tables were made in the database named as “authitem”, “authAssignment” and “authItemChild”. In the code, different roles, tasks and operations where defined and connected accordingly. This code can be found in actionSetUp() in the site controller. The website administrator does not need to run this action everytime but only once when the above mentioned three tables are empty and the website is being deployed for the first time.

Note: Whenever new operations, roles, tasks are created in the setup(), it is mandatory to run the setup() action. Also it should be noted that actionSetUp() should be executed only once and not more than that.

AuthAssignment table stores the authorization/roles/functions of each kind of the user. Whenever it has been truncated, the existing users’ roles have to be inserted manually in that table for the authorization to work. Duplicate entries in authItem and authItemChild should be avoided.

To enforce these authorizations, changes were made in the following functions and/or files:

|  |  |
| --- | --- |
| Name of the files | Changes made |
| Protected/config/main.php | Added code at line 39 for RBAC |
| siteController.php | Added actionSetup() code for RBAC |
| userController.php | Edited - actionCreate(), actionUpdate(), actionAdmin(), actionIndex(), accessrules(), filters() |
| eventController.php | Edited - actionCreate(), actionUpdate(), actionAdmin(), actionIndex(), accessrules(), filters() |
| userEventController.php | Edited - actionCreate(), actionUpdate(), actionAdmin(), actionIndex(), accessrules(), |

## Inconsistency in database on deletion:

Since some fields are related to other fields across table via a foreign key, they are related to each other. But in the previous version of the database, whenever a foreign key in table was deleted, it did not result in the deletion of the related record in the other tables. This created inconsistencies and gave runtime errors resulting in bugs. I changed the referential integrity to “on delete cascade on update cascade”. Now the changes are propagated across tables. Somewhere the “on update” constraint is restricted because of the demand of the application functionality.

# Stumbling blocks and challenges faced:

1. Being unaware of the framework and never had worked in a framework based environment, it was difficult to start off with learning it in a short time.
2. I had known the MVC architecture in theory before, but actually learning how data is passed and using various predefined method was one of the many challenges in the project.
3. Though framework frees the developer from coding simple things and writing mundane SQL queries, it is not easy to know how to how and where to use this feature in the beginning.
4. Digging into code and understanding what and how it is accomplishing certain functionality was difficult as the various tasks are done by various methods which are located in different files. To understand who is doing what was a challenge.
5. Debugging a code in server side programming language is a challenge in itself let alone doing it in a MVC architecture where things are located in different files and some things are done by javascript and jquery.
6. There were times when I could not locate a piece of code which was driving a particular functionality but Dr. Hughes came to the rescue, showed faith in me and guided me aptly.
7. Designing and thinking about the schema for RBAC was a challenge and I had to take care of conflicting roles and responsibilities.
8. Learning role based authorization and implementing it in the code. Had to make a lot of changes and had to remember them.

# Lessons learnt:

1. I got to learn about frameworks and how to use them which is very valuable. I got to apply what I learnt in the application which further strengthened my learning and gave me confidence in using it for my future projects.
2. I used to know PHP and MySQL but learning object oriented programming was something I wanted to do since a long time and this project gave me the right opportunity to do the same.
3. I learnt to analyze code and decipher its functionality just by reading it.
4. It gave me an opportunity to improve my debugging skills by employing new techniques to figure out the exact point where there is a problem.
5. It made me think from the users’ perspective and taught me to develop projects in a way that is user friendly and has functions which a user would want.
6. This was the first time I was working on a project which was majorly developed by somebody else, hence getting used to somebody else’s style of writing code was a challenge and I learnt that whenever I am developing code, I should keep in mind that it should be easier for somebody else to read my code.
7. I learned how to be more systematic in software designing and debugging approaches.
8. Solving problems step by step by analyzing the root cause of it is an important skill. This project taught me the same.

# Future Works:

1. The scalability of the application has to be tested as a lot of tweets will be stored and retrieved. There is a need to test the performance of the application in an environment when there is a lot of data in the database.
2. UI can be changed a bit and can be made more intuitive.
3. Whenever a new node under the root category is created in the streaming or archive mode, the user has to refresh the page to see the changes. This was speculated to be a browser related problem but after testing across different browsers, it was found that it was not. The bug can be more fundamental and the point of error has to be determined and solved.
4. The application should undergo rigorous black box testing.
5. RBAC can be made more efficient. I have written some code for the future changes and compatibility.
6. In the event access page, the owner can delete himself from the event. This should be prevented in the future. Owners should not be deleted.

# Final thoughts:

This project was a great learning curve and I did not expect to learn so much. At the end of the day I liked what I was doing. There were some phases of frustration but those entire situations made me more robust as a developer and strengthened my analyzing skills. I regret that I was not able to deploy it in the field where it would have been used by the target audience. This proved to be a great independent study for me.

* Rohan D. Shah

A01943549