MD5 Guys: Employee Management Website

“Security above all else”

Website by: Kyle Cooper, Johnathan Lamberson, Mike Bercume

CSCI 370 Final Project

Professor Jung

Table of Contents

[Overview of Project 3](#_Toc449989102)

[*The Database*: 3](#_Toc449989103)

[*On Going Maintenance*: 4](#_Toc449989104)

[Design Concepts and Implementation 4](#_Toc449989105)

[*Design* 4](#_Toc449989106)

[*How to Login* 5](#_Toc449989107)

[*Implementation* 5](#_Toc449989108)

[Appendix and Code 10](#_Toc449989109)

[*Citations* 10](#_Toc449989110)

[*Code* 10](#_Toc449989111)

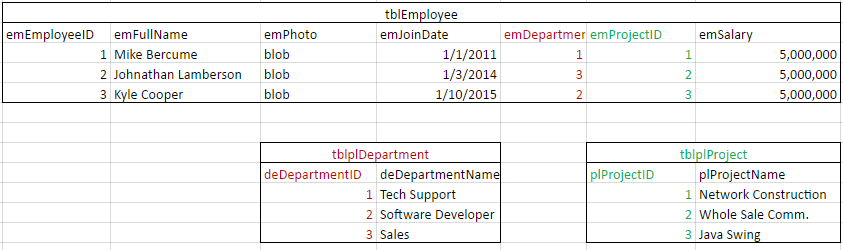
# **Overview of Project**

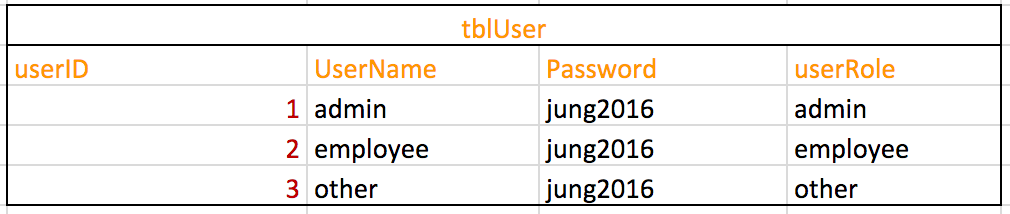
MD5 guys is a research security firm that specializes in encryption and hash algorithms; specifically, MD5. We are a small to medium research firm employing many different people many different areas.

However, we had one problem, we have no internal database to search for our employees and look at what departments and projects they are working on. We decided to start to build a database that would allow us to search through these employees. This project is designed to be an internal database for the company. Not all employees at MD5 guys can have access to this site. Access is granted through a login system. This includes an administrator, to be able to create new accounts. The administration role can make new accounts for users. Anyone who is not an administrator cannot make accounts and cannot go to the admin page to create new accounts. This all resides in the admin tool on the website, located on admin.php. There is also an “employee” role and an “other” role. The employee role can update the records on the site. For example, if a project was finished, the project can be removed from the database and the Employees can then be reassigned to a different project. The other role is read only. A user with a read only role will not be able to delete, add, or update users.

Another important factor was to be able to search through the database in all three tables. For the project table, it will only return current projects in the database. For the department table it will only return department names in the database. For the employee table we wanted to display the name of the employee, there salary, department, current project, employee ID and display a photo for that employee. The search needed to be able to search by project, department and full name. The search can use multiple keywords.

The Database:





We needed the database to be able to hold the Employees ID, Full Name, Photo, Join Date, Department, Project and Salary. There are three tables, Employee, Project, and Department. Employee holds the data for each employee with a primary key of emEmployeeID. The employee table also has two foreign keys, emDepartment and emProjectId. These two columns will hold foreign keys (primary keys of their respective table). This makes it easier to update/delete/add and manage the database. This key is a reference to an entry in the either the project or department table. However, we here at MD5 guys only like our employees to concentrate on one project at a time, therefore they cannot participate in multiple projects. The same is also true for an employee’s Departments.

The table for the users is called tblUser and holds a userID, UserName, Password and userRole. Each row in this table can have access to the database. The userID is a value that is auto incremented. The username is created in the admin.php page by admins only. The password field is an MD5 hash of the user’s password, for security purposes. The last field is userRole which hold the role the user plays on the database.

On Going Maintenance:

Maintenance of the website will be brought by the continuing work of the developers to bring bug fixes, designing and implementing new features, upgrading the website functionality, and design. Another important part of maintenance is ensuring that the site is secure. When new features and more secure ways of doing a certain functionality became available or known we will need to implement them accordingly.

Backups of the site and database will also need to be made so that we can always revert back to an earlier version if needed. This will also ensure that if the website and its data were ever lost, we could bring the site back online within a reasonable time

# **Design Concepts and Implementation**

## *Design*

MD5 guys is a security company, we do our best at creating algorithms. This means that we can design something to work well, but it may not be pretty. In our website design came second to functionality (ensuring our PHP functions worked properly). Our first design challenge was picking a coloring scheme, and since our name is MD5 Guys and we love Five Guys the restaurant, we decided to go with a red theme. Next we needed a logo, for that we also turned to Five Guys. Our image at the top of the page came from, BostonEater.com (see appendix for citation) and is the Five Guys Logo, just changed a little to fit our needs.

We wanted our website to be easy to understand and navigate when our employees are using it. This would help reduce the time spent on employee training and increase simplicity This is why we went with a more basic design. We even have a home page that explains all the functionality of the website. All tables on the website are designed so they are large for users to see and easy on the eyes.

All of our design uses CSS and resides within *project.css*. This makes the website more modern than tradition in-line html styles. It also makes creating different designs a lot easier.

How to Login

In order to view any of the site the user must first login. To do this the user needs to go to the index.php page and then enter their credentials.

Accounts to login with:

|  |  |  |
| --- | --- | --- |
| **Username** | **Password** | **Role** |
| admin | jung2016 | Admin Account |
| employee | jung2016 | Employee (regular account) |
| other | jung2016 | Other (Read Only access) |

Implementation

As a group we decided what functionality we needed. We broke up all the functionality into parts and then everyone was assigned what to work on. However, we first needed to agree on general variable names, functionality and a few other things. For example, the session state variables error and message would be used to display messages on various pages throughout the website. We each worked on our own parts and then brought them together to make them work as one. For the most part this worked well, with almost no conflict.

Once we got the basic functionality working (delete, search, add, and update) we then worked on fixing bugs and small improvements. Then after this we implemented the design using CSS. After this we worked on validation, and other small things to make the site more functional for the user.

To create the website, it was hosted on a Raspberry Pi webserver. Everyone in the group had access so they could SFTP files to the Raspberry Pi and even use MySQL. This allowed everyone to make changes to the project with the most up to date code.

Below is a list of all the pages that are included with the project and a brief description of how they work.

**Project.css**

This page holds all of the CSS for the website. Changes in this file will be reflected on how the site looks.

**index.php**

The main page that welcomes the visitor with a login page. A visitor will be redirected to this page if they try and visit another page when they are not logged in. Once the user has logged in successfully on index.php they will be redirected to the home.php page.

**home.php**

This page gives the user an introduction to the website. It tells them about each functionality of the website. For example, how to use the search function, update function and even the admin tool.

**search.php**

Search.php is the main page where the users search takes place. The search can use multiple keywords separated by a comma and a space. For example, keyword1, keyword2, keyword3. Once the user has entered the term(s) they wish to search they then select what table to search. There is a dropdown box that allows the user to select Employees, project, or department. A search in the Employee table will return rows from that table. The project and department table will return rows from the project or departments tables.

The search all function returns all the rows in whatever table the user has selected from the dropdown box. For example, if a user selected projects from the dropdown menu, and then clicks search all, all of the project rows will be returned.

Once the query is displayed the user can update or delete the entry. The delete button deletes that specific entry and sends the user to delete.php, where it deletes the entry and returns the user back to search.php, unless there is an error. The update checkboxes allow the user to select multiple entries they want to update; they are then sent to update.php.

The print function allows the user to print out the entire html function using window.print(); from JavaScript. Once in this menu they can either print it to a printer or save it to their computer as a PDF.

A session state is kept for each search. Whenever the users load this page their previous search will show up. This makes it easier to keep looking at their search results, instead of having re-search their keywords.

**insert.php**

Insert.php relies on add.php to work. Insert.php askes the user how many entries they want to add into the database and what table they would like to add rows into. For example, if the user wants to add 12 rows into the employee table; insert.php will generate a table displaying 12 rows. Each row will have a place to input an employee’s name, join date, photo, current department, current project, and salary. The current department and current project options are dropdown lists containing up to date lists of projects and departments. The user inputs their data, and then clicks add. There data is validated and then pasted to add.php

**add.php**

add.php adds all the user data from insert.php. Two variables are past to it using $\_POST: parameter and values. The parameter specifies what table the data needs to be entered into. Values holds all the user data from inset.php. The data is then sanitized and validated before putting it into the database. Salary is stripped of commas, and images are converted into blobs. Assuming the query is successful, then it uses a header(); function to return the user back to insert.php, with a message telling them the query either failed or was successful.

**delete.php**

delete.php does not display anything, unless a query failed. It gets its values from search.php. If a user wants to delete an entry, they must query the database on search.php and then click the respective delete button. Then the employeeID of that row is sent to delete.php and removed from the database.

However, since there are dependencies in the database, MySQL will not allow the deletion of rows that are being used as foreign keys in other tables. If the user tries to delete from the projects table or department table, delete.php will display an error and display a table showing them who is still attached to that project or department.

**update.php**

Update.php relies on search.php and updatedb.php in order to function properly. Update.php takes a post value from search.php and creates a table with a form so that the user can update the selected entries. This site generates a row for each item that the user wants to update. It queries the database to get the information for each row and then populates the textboxes within the row. The user then modifies what they need to, and updates the website, sending the data to updateDB.php.

Update.php will display rows with appropriate values for updating whatever table the user wants to update.

**updateDB.php**

UpdateDB.php relies on update.php to get its data. This page receives the data from update.php and inserts the data into the database. It cleans the input from the user before going into the database. For the employee table, updated.php checks whether an image has been uploaded or not. If there is not image to be updated, it leaves the current one in the database, otherwise it will update the image. Once the appropriate table is updated, assuming it did not fail a header will redirect the user back to search.php.

**session.php**

session.php holds the session state for the user account. This page prevents a person who is not logged in from viewing any page. For example, if a user tries to go to search.php but they are not logged in, they will be redirected to index.php, which is the log in page. This is done by using require “./session.php” at the top of every page. It checks the session state to see if the user has logged in, if not they are then redirected.

**login.php**

This page does the actual login validation. It looks in the database for a matching user and password. Since all passwords in the database are inserted into the user table with an md5 hash, this page looks for the same md5 password in the database.

**logout.php**

This page lets the user log out. It checks the session variable to see if a user is logged in if they are then the session variables are destroyed and then they are redirected to the index.php where the logout message is displayed. This effectively logs the user out. The session is ended once the logout message is displayed.

**Conclusion**

In conclusion the search functionality for MD5’s website is good for a small to medium database. However, if the company where to grow significantly past our current number of employees (about 20), then MD5 it might be best for MD5 to consider a new algorithm for search. Search.php is currently using LIKE statements to search for keywords. This can be a little sluggish when dealing with many entries in the database. The database design may also change if the company wants to implement more into the database. However, this does not prevent the search function from performing all of the core requirements that the company needs.

# **Appendix and Code**

## *Citations*

Five Guys image from: Boston Eater (ht<tp://boston.eater.com/2016/3/4/11161268/five-guys-allston-op>ening)

Code: please see zip file