Semester Project

Final Report

By:

*QwickPic*

CSIT 337-01

Internet Computing

Dr. John Jenq

Database name: qwickpic

Team:

Arthur Rozenberg (Team Leader)

Chris Dahdouh

Table of Contents

**I.** [**Contributions**](#_58ztzklsbmwm)………………………………………...………………………………………...[**2**](#_58ztzklsbmwm)

**II.** [**Project Summary**](#_x8c6b8h5jmt9)………………………………………...…………………………………...[**4**](#_x8c6b8h5jmt9)

**III.** [**List of existing accounts’ info**](#_e8xzw9ojy5ao)……………………………………………………………….[**5**](#_e8xzw9ojy5ao)

**IV.** [**Front end structure**](#_n81jjg6v0pt1)………………………………………...………………………………...[**6**](#_n81jjg6v0pt1)

**A.** [File relationships](#_9f6b6if9q76g)…………………………………………………………………………... [7](#_9f6b6if9q76g)

**V.** [**Back End Scripts**](#_tloctpm6lxov)…………………………………………………………………………….[**18**](#_tloctpm6lxov)

**A.** [Dbh](#_974hbuvu6het)………………………………………...……………………………………………... [19](#_974hbuvu6het)

**B.** [Login Header & Login Functions](#_9wu2ueal8t11)………………………………………………………... [20](#_9wu2ueal8t11)

**C.** [Header & Display Functions](#_91eb5tckpf89)……………………………………………………………... [23](#_91eb5tckpf89)

**D.** [Get Functions, Create Functions, Edit Functions](#_51qn05i8by1d)………………………………………... [27](#_51qn05i8by1d)

**E.** [Album Submit & Upload Submit](#_fkbmi31th8mm)……………………………………………………….... [30](#_fkbmi31th8mm)

**F.** [Javascript Functions](#_pmesfz44gn7q)………………………………………………………………………. [31](#_pmesfz44gn7q)

**G.** [Logout](#_fv6ue5i5kvzt)……………………………………………………………………………………. [33](#_fv6ue5i5kvzt)

**VI.** [**Database Layout**](#_kgt80vd56oqt)…………………………………………………………………………....[**34**](#_kgt80vd56oqt)

**A.** [Accounts Table](#_coeldg8xn700)………………………………………………………………………….... [34](#_coeldg8xn700)

**B.** [Album Table](#_s8u914wrwsnp)…………………………………………………………………………….... [36](#_s8u914wrwsnp)

**C.** [Pictures Table](#_i39wqcamp4zg)…………………………………………………………………………….. [37](#_i39wqcamp4zg)

**D.** Admin\_accounts Table……………………………………………………………………38

**VII.** [**How to Upload QwickPic**](#_ln2j4r9rtycu)…………………………………………………………………[**3**](#_ln2j4r9rtycu)**9**

# Contributions

List of files and tasks that each team member worked on

**Arthur:**

* qwickPic\_dbh.php
* qwickPic\_jsFunctions.js
* qwickPic\_logout.php
* qwickPic\_loginFunctions.php
* qwickPic\_displayFunctions.php
* qwickPic\_getFunctions.php
* qwickPic\_createFunctions.php
* qwickPic\_editFunctions.php

**Chris:**

* qwickPic\_home.php
* qwickPic\_header.php
* index.php
* qwickPic\_upload.php
* qwickPic\_createAlbum.php
* qwickPic\_forgotpassword.php
* qwickPic\_loginHeader.php
* qwickPic\_createPassword\_form.php
* qwickPic\_createAdmin.php

**Both:**

* qwickPic.css
* qwickPic\_albumSubmit.php
* qwickPic\_uploadSubmit.php
* qwickPic\_search.php
* qwickPic\_pictures.php
* qwickPic\_profile.php
* Database construction

# Project Summary

The name of our team and website is QwickPic. The goal of our website is for users to be able to create accounts and upload pictures to their accounts quickly. Only the user and admins can see these pictures, so they are private (users cannot see other user’s pictures). The user can also create albums and put pictures into these folders. The user can view these pictures or can search for a particular album. The pictures can also be edited or deleted. The user can also change their password if they wish. If the user forgets their password upon request, they can send an email to themselves and QwickPic will generate a temporary password for them to use. The user then will have to change their password. QuickPic also has administrators that can create other admin accounts. Admins can also search for any user and delete some of their pictures if deemed inappropriate.

# List of existing accounts’ info

The following is a list of all account’s that currently exist in QwickPic

**Regular Users:**

* **Username:** shadowarty **Password:** ilovecats33 \*NOTE: FOR TESTING USE THIS ACCOUNT AS IT ALREADY HAS MANY PICTURES ALREADY EXISTING!
* **Username:** doomsday **Password:** doomsday

**Admin accounts \*NOTE: YOU MUST FIRST LOG INTO ONE OF THESE ACCOUNTS TO CREATE A NEW ADMIN, AS ONLY ADMINS CAN CREATE OTHER ADMINS!\*:**

* **Username:** rozenberg **Password:** rozenberga
* **Username:** dahdouh **Password:** cdahdouh

# Front end structure

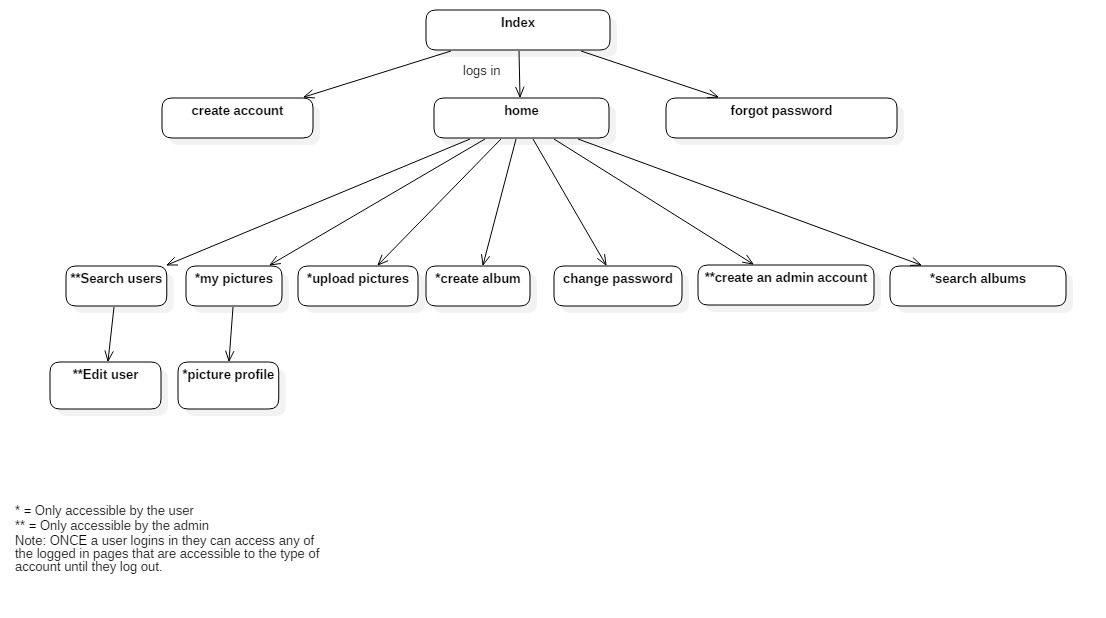


Figure 1: Hierarchical accessibility of pages in QwickPic

## File relationships

The entire qwickPic website uses a single CSS file. The goal of the CSS file is to decorate everything on the site. Whether it’s adding padding to a div or increasing the size of a textfield, anything related to styling happens here.

The first page any user sees when accessing qwickPic is the default index page. The index page is the login page. The user is prompted to enter their username and password. If they do not have an account they can click on the link to make one. Also, if they forgot their password, they can click on the link to access the forgot password page. The index page is composed of the login header page which will be discussed during the back end portion. Essentially the login header page contains a preset which we can use across multiple pages to have a similar look. Ultimately, the main goal of the index page is for the user to log in.



Figure 2: the index page for a user to be able to login to QwickPic

The next page is the createAccount page. The user can access this page by clicking the link on index to create an account. They will be prompted with several text fields, to create a username, password, email, first name and last name. If the username or email already exists they will be prompted to enter re-enter their information until the account is successfully created. Once, the account is created they will be redirected to the index page. Just like the index page, createAccount is composed of the loginHeader page.



Figure 3: The create account page allows the user to sign up for a account, so they can log in

The next page is the forgot password page. This page includes a single text field for the user to enter an email which is associated with their account. If the user enters an email that does not exist with an account they will be prompted with a message to re-enter their email. Once they enter the correct email, they will be sent a temporary password. The goal of this page is to get the email from an account. Just like the index page, createAccount is composed of the loginHeader page.



Figure 4: The forgot password page allows a user to enter their email, and their password is reset and sent to that email.

The next page is the home page. This page can only be accessed once you successfully log in. This is the first page you see when you login. The goal of this page is provide a “hub” of different things you can do ranging from changing your password or logging out, etc. You will see your first and last name at the top, this was created using PHP with Session variables from database which will be discussed later. Depending on the type of account user or admin, they will see a different profile picture. This page is composed of a header page which creates a preset, so that any page containing the header will have the preset( i.e. all pages with header will have the navigation bar).

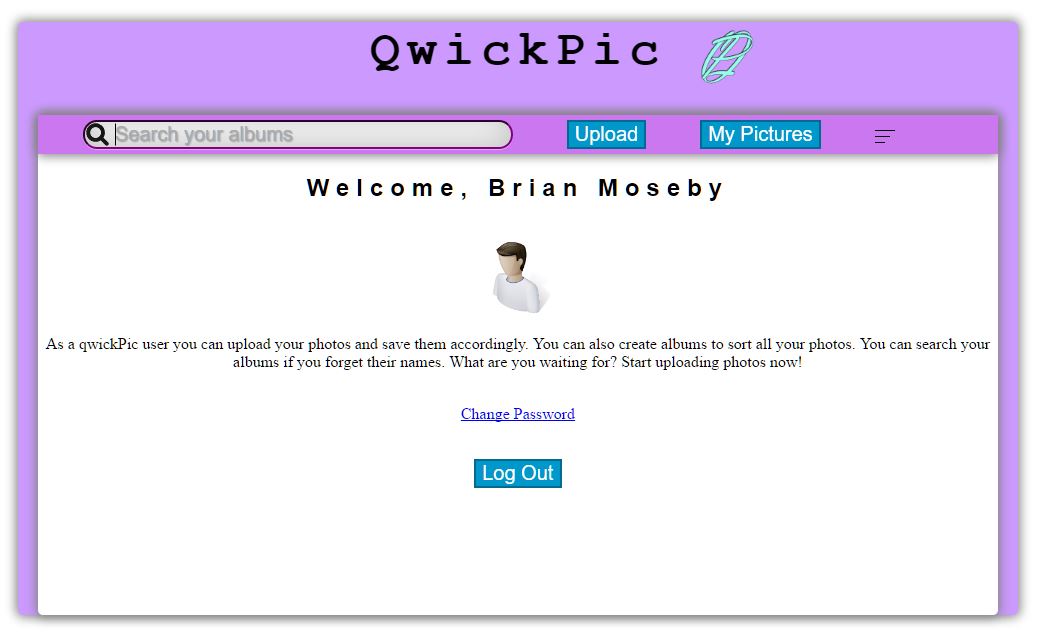


Figure 5: The landing page where you are taken after a successful login

The change password can be accessed from the home page or the side bar. The main goal is to of course, change your password. There are three text fields; the old password of the user, the new password and confirm new password. First we check if the old password is correct of the current user (session variable in php). If it is not correct the user’s new password will not be added. If it is correct then it will check if new password and confirm password are the same, if so the new password for the account is added. Just like homepage, this page includes the header page.

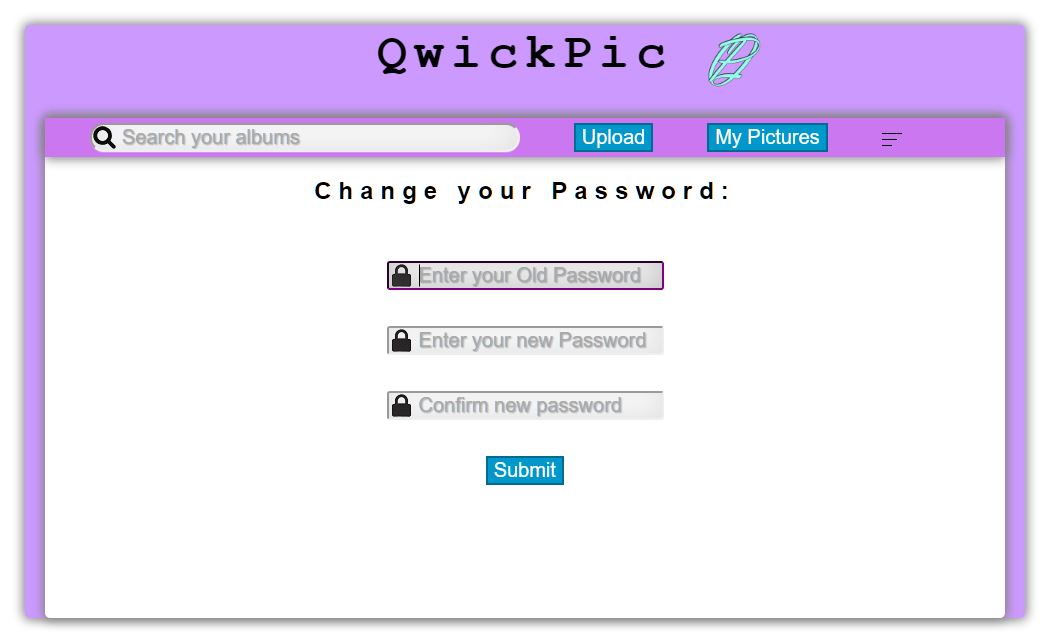


Figure 6: allows the user to change their password if they enter their current password correctly

The create album page allows a user to create a new album to store their pictures into. This page can only be accessed by users from the sidebar. If an admin tries to access this page they will be prompted with an error. Albums consist of an album name and a cover photo. The cover photo is uploaded to the cover photo folder. Cover photos can only be in .jpg, .png, .tiff, and .gif. Also all images must be less than 10MB. If these conditions are violated then an error will occur. Just like homepage, this page includes the header page.

The upload picture page allows a user to upload a picture to the website for viewing. All files will stored in the Images folder. Pictures can only be in .jpg, .png, .tiff, and .gif. Also all images must be less than 10MB. If these conditions are violated then an error will occur. The entire goal of this website is dependent on this page, so it must work! The user also has to pick an album to put the uploaded picture into from a drop down.. If no albums exist user will first be prompted to create an album. This page can be accessed only by users from the navigation bar or side bar. If an admin tries to access this page they will get an error. Just like homepage, this page includes the header page.



Figure 7: allows the user to upload a picture with a unique caption name and an album to put the picture into.

The my pictures page allows the user to view the pictures they uploaded to the site. When this page is first entered the default is that all the uploaded pictures are shown to the user. However, there will be a drop down menu of albums which when clicked on will dynamically only show the pictures in the particular clicked album. This page is only accessible to the user by the navigation bar or the side bar. If an admin tries to access this page they will be prompted with an error. Just like homepage, this page includes the header page.

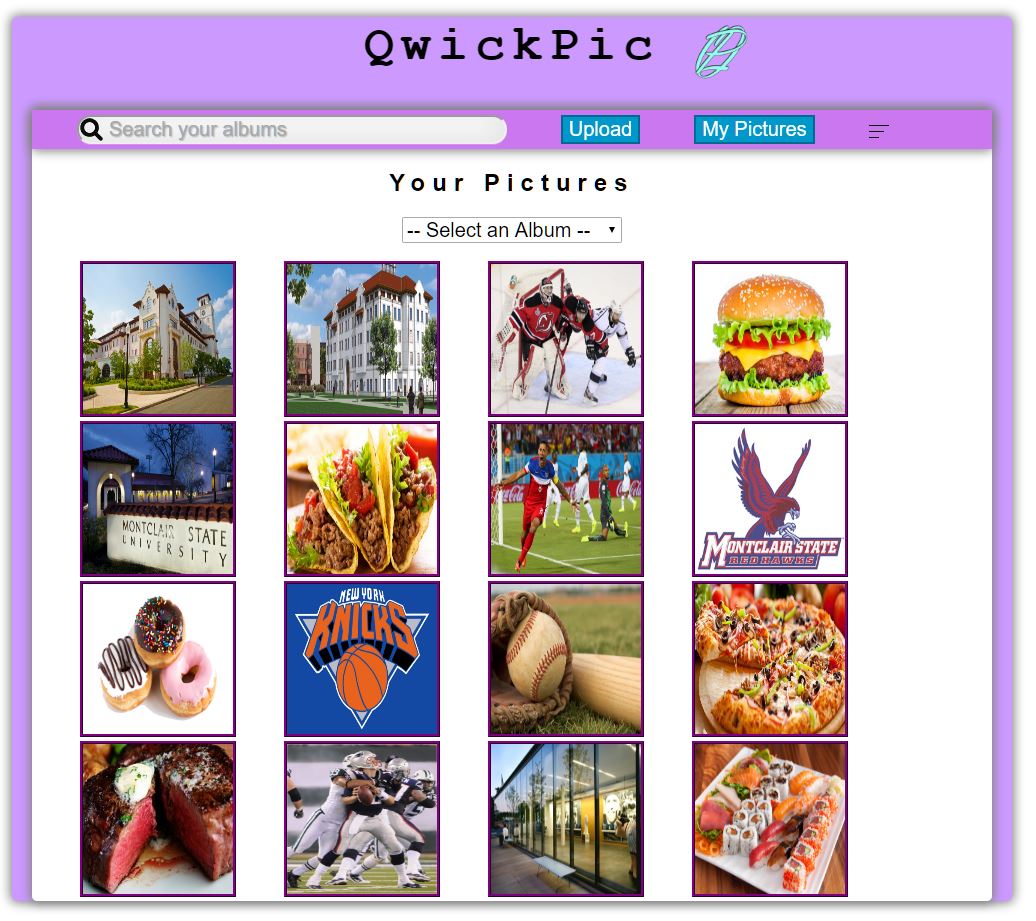


Figure 8: displays all the pictures that the user uploaded.

The picture profile page is accessed when a user clicks on a particular photo on the my photo page. It goes to a profile page for that picture using generated from a php function and the pictures id from the database. This page can only be accessed by users from the my pictures page. Once on the page the user can delete the picture or move it to another album. If an admin tries to access this page they will be prompted with an error. Just like homepage, this page includes the header page.



Figure 9: A profile for each picture that was clicked on, you can edit the picture with the 3 buttons.

The search album is accessed when a user types into the search bar a particular album they want to look at. If the album does not exist, then a message will be displayed. Otherwise the user will be taken to the my pictures page with that album. This page can only be accessed by users since, only users can have albums. If an admin tries to access this page they will get an error. Just like homepage, this page includes the header page.



Figure 10: You can also search for an albums for a specific album

The create admin page is only accessible to admins. The goal of this page is if an admin is requested they will need their own account, so thus other admins have the power to do so. This page can only be accessed by admins on the navigation bar or the side bar. Just like the regular create account page there will be the same text fields. If the username exists in either the users or admins tables then the account will not be created. If a user tries to access this page they will get an error. Just like homepage, this page includes the header page.



Figure 11: admins can create more admin accounts for new admins if requested to do so.

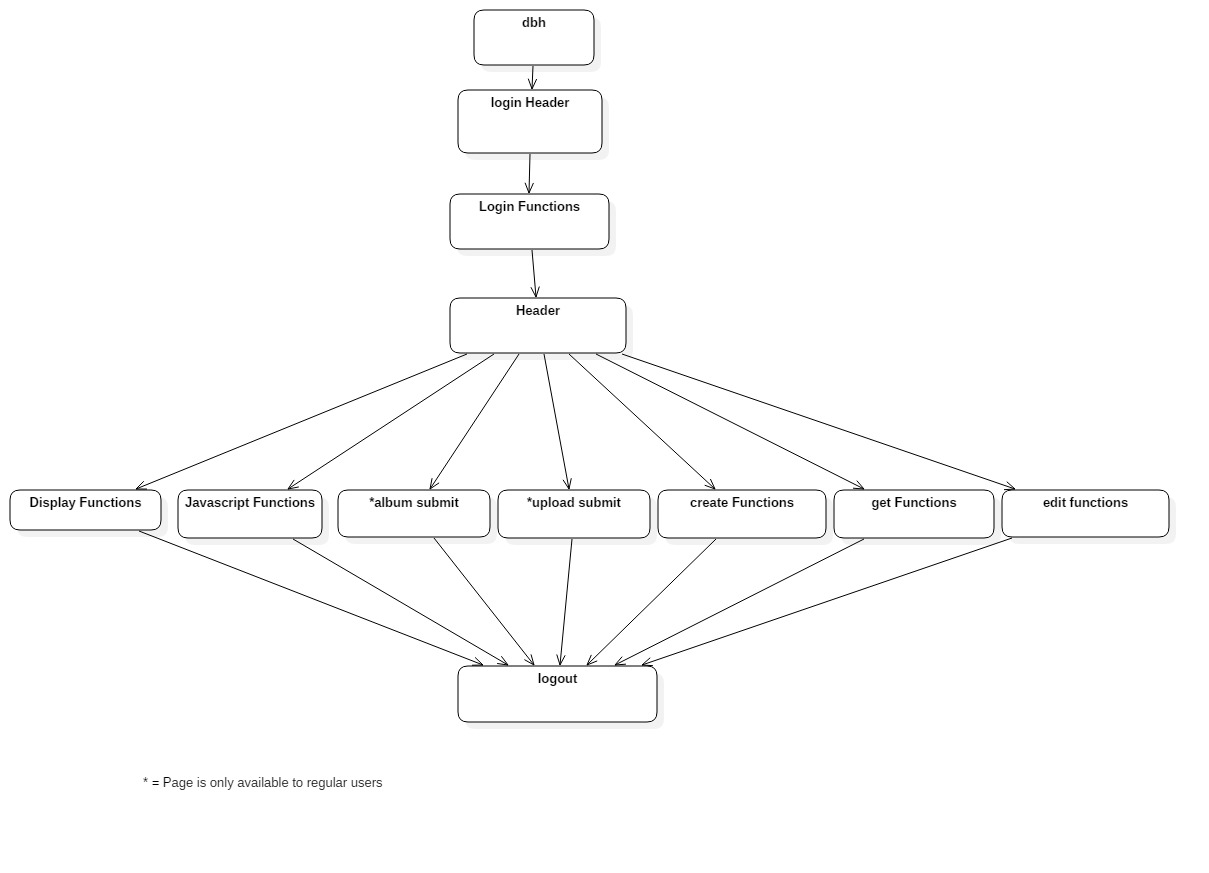
Admins can search any user’s photos. First in the search bar they type in a user’s username and they will be allowed to view all of that user’s pictures. If a picture is deemed inappropriate by the admin then the admin has a right to delete it. If a username is not found then the proper no results message will be displayed. If a user tries to access this page they will be prompted with an error. Just like homepage, this page includes the header page.



Figure 12: In admin view, instead of search albums, admins can search user’s and view or delete pictures

Finally once the user or admin is done with their session they can logout and be redirected to the login page. This destroys the session and any variables associated with the session (this will be discussed more in depth during the back end phase).

# Back End Scripts



## Dbh

The dbh.php page is a simple function page that connects to the qwickPic database. It is single handedly the most important script file in the whole project. Without this script you would not be able to connect to the database and get, add, delete or update the tables in the databases. Any other script that use of these actions from the database will fail if this file is not provided.

It is a rather simple file that needs only a connection method that takes four parameters to connect to the database. The first parameter is the type of connection to our database. Since we are using Xampp, all we need is a simple localhost connection. If we were using an actual server we would need a different connection, which varies. The next parameter is the username to access the database. This can be easily created in phpmyadmin and stored here. We want this account to be only able to change database info. The fourth parameter is the password of the 3rd parameter of the account. This must be the same as the password created for parameter 3 in phpmyadmin or the connection to the database will fail. The final parameter is the name of the database that you created in phpmyadmin. Finally we output if the connection was successful or not. All of this is put into a single function named db\_connect which we can call on any script that requires information in the database.

## Login Header & Login Functions

The login header as mentioned in front end is a preset script used for all front end login files to have the exact same look in between them. The login header has a certain width of a div tag so all files connected to it will have the same size and color that is set in the login header. The login header also checks if an admin or user is already logged in using Session variables (this will be discussed in login functions). If so, they will be redirected to the logged in homepage. This is because if a user is already logged in, we don’t want them to simply type the page and login again because that may cause errors on the site since, the session variables were never destroyed (this will be discussed in logout).

The next script is is the login functions which is a full script of user created functions that allow the user to login, create an account, email temporary password, change password, a generate temporary password for send the email and a send email function.

The function register, gets the information from the text field from CreateAccount page, checks if the username and email exists in the user and admin tables. If not it will add it to the database. This function has 5 parameters for the 5 different textfields to get the user generated text from. To get the info from the textfield, we use a POST, and pass it to a php page. We can then call a post variable and use that as a parameter. If the username or email already exists user will be forced to enter a new username or email. Instead of adding the regular password into the database, we hash the password and store that hashed password into the database instead for security. If an admin session variable is set then an admin is logged in and they can create other admin accounts. If not then only a regular user account can be created.

The next function is the login function which allows the user to login to their accounts. This function is activated in the index page once the user clicks the button. This function has 2 parameters, username and password. This is once again accessed via POST. First the function checks if the username exists in either the admins or regular users tables. If no username exists in either tables than it is an incorrect login. If a username was found, then it takes the password entered for parameter 2, does a password\_verify method which compares if parameter 2 matches the hashed password stored in the database. If the password\_verify returns false then it is an incorrect login. If password\_verify returns true than it is a correct login and we can redirect the user to the logged in homepage. We can also set our global session variables, so we can access them throughout the site. If a regular user logged in, then the user session variables will be set with their username, user id, first name and last name. If an admin logged in, them the admin session variables will be set with their username, user id, first name and last name.

The next function is the email password which sends an email if the user forgot their password. This function is used in the forgot\_password page when the submit button is clicked. This function has one parameter, email. This is accessed via POST. This function first checks if the parameter email is associated with an account. If not user is forced to enter their email again. If an email is found then it gets the password of that account and calls the reset\_password function to reset the password. It then calls the send\_email function and sends that new resseted password to the email that is provided with that account.

The next function is the reset\_password which resets the user’s password when the user forgets their password. This function is called in the email\_password function. It has not parameters because it just inserts the random password into the account that the email is provided with. First it declares a string of lowercase, uppercase and numbers. It then get the length of the entire and string and picks a random length from minimum 8 characters to 20 characters. Finally it goes through a loop until the length of the random string and adds a random character to the string until it reaches the length of that random string. It then hashes that string and adds it into the database for the new password. Finally, it returns the random generated string.

The next function is the send email function. This function actually sends an email. It is used in the email password function. It takes 2 parameters, who the email is sent to, and the message that is sent. This function needs an existing email to send emails. This function uses PHP mailer to send the email. We then use the two parameters provided to declare where the email will be send to and what message will be displayed.

The final function is the change password function. This is the only function that is declared where the user is already logged in. It is declared in the change password page. This function takes 3 parameters, the user’s current password, the user’s new password, and confirm that the user’s new password is the same (via POST). If a regular user is logged in (checked with session variable), then check the accounts table. If an admin is logged in then check the admin accounts table. First we get the hashed password of the account that is logged in. Use password\_verify to make sure that parameter 1 and hashed password is true. If false then the password is not changed. If true then check if parameter 2 and parameter 3 are the same. If so, hash the new password and add it to the database. Otherwise, the user cannot change their password.

## Header & Display Functions

The header script is accessed after a user has successfully logged into the site. Just like the login header it is a preset for all pages to have once you are logged in. Every page that includes this file will have a navbar, sidebar, etc. This page also checks if no user and no admin is logged in. If this is the case, then they will be redirected to the login page. We do not want users to access logged in pages without actually being logged in. If this was not included than any user without being logged in can simply type the url of a logged in page and access it freely.

The next script, is display function. This page is a list of user created functions that display different things when the user is logged in. This page has html\_url, display\_navbar, display\_sidebar, display\_homepage, display\_dropdown, display\_pictures, admin\_display\_pictures, display profile\_pic, search, sort\_album.

The first function is html\_url. This function returns a html string that returns a string of a link. This function has two parameters, the link and a title. The link is a simple link of where you want page to go. The title is the end of url which is added to the end of the link to make the new html url unique. This is important for pages like the pic profile page because we do not want to create many profile pages. Instead we make a unique url and generate each picture to have its own url.

The display\_navbar is a really simple function. It is called in header, so all logged in pages have a navbar. First using session variables we check what type of user is logged in. If an admin is logged in they will be generated with a seach bar for searching users, a create admin button in the nav bar and a button to open the sidebar. If a regular user is logged in they will be generated with a search bar for searching their albums, a upload button for pictures, a button to check all your pictures, and a button to expand the sidebar.

The display\_sidebar function is really similar to display\_navbar. It is called in header, so all logged in pages have a navbar. First using session variables we check what type of user is logged in. If an admin is logged in then the generated sidebar will have my profile, change password, create admin, logout, and a button to close the sidebar. If a user is logged in then the generated sidebar will have my profile, change password, upload, create an album, logout and close the sidebar.

The display\_homepage function is similar to the previous two functions. It is called in the homepage. This function two different homepages depending on which user is logged in. The image and the Text is slightly different and it gets the user’s First Name and Last Name and displays it on the screen.

The display\_dropdown function displays a dropdown of all the albums a particular user has created. It calls the get\_albums function from the getFunctions paged (will be discussed). When calling this function it gets the query of all of the user’s albums based on the session id of that user. It then places all these albums into a drop down. This function is used in the upload page where a user selects what album they want their picture to be put into when uploaded. This function is also called in the user’s my pictures page to sort the pictures based on the album clicked.

The display pictures page function displays all the pictures a particular user has uploaded based on their session user\_id. It calls the get\_pictures function from getFunctions page. When calling this function it gets all the pictures associated with your user\_id. It then puts it into an array and loops through the whole array printing all the pictures.

The next function admin\_display\_pictures is similar to display pictures except, this function only works for admins because this function is only applied after a username has been searched for and only admins can search for users. This function is called once a user clicks on the link of the username which they are searching for and all of that user’s pictures will be outputted because the admin\_get\_userid from get functions will be called, getting the user id of that user.

The next function display\_profile displays the profile based on the picture that was clicked on. This function is called in the profile page. It gets the picture id of the picture that was clicked on using the get\_url\_pic\_id function from get functions. Then it outputs the picture based on what the picture was clicked on with its caption.

The next function sort albums has 1 parameter, the name of the album. The goal of this function is get all the pictures that belong to that album name parameter. This function does 2 queries. First it needs to find the album id of parameter 1. The album id is then stored in a variable for the second query. For the second query, we simply output all pictures that have the same value as our album id. This function is called in our Javascript functions using Jquery and Ajax to sort an album dynamically in the pictures page with album that is selected in the drop down menu.

The last function search displays all the search results that were typed in. This function has 1 parameter, the text that was entered in the textfield via POST. First it checks whether the user is a admin or a regular user based on the session variable. If the user is an admin then they can search from users from the accounts table. It does a query with the parameter and compares the usernames that are similar to the parameter. If no results are found, then nothing gets outputted. If at least 1 result was found then print the number of results and print every result. If the user is a regular user they can search from the albums table. It does a query with parameter and compares album names similar to parameter. If no results found, no output. If at least 1 result found then print number of results and all of the album names with their cover photos.

## 

## 

## 

## 

## 

## Get Functions, Create Functions, Edit Functions

The get Functions page is simply a list of functions that return queries of the results of the tables. This page is simply shortcuts to avoid re writing the same code. The function names are get\_albums, get\_pictures, get\_url\_pic\_id, and admin\_get\_userid

The get\_albums and get\_pictures functions are essentially the same except one query returns all the albums that belong to that user based on their session user id and the other returns all pictures that belong to that user based on their session user id. The get\_albums function is used in the display\_dropdown function from display functions. While get\_pictures is used in display\_pictures function from display functions.

The get\_url\_pic\_id and admin\_get\_userid are really similar functions. They both return the id of the end of the url using the Server variable which returns the url. We then use the ltrim method and get rid of everything to the left of the actual id number in the url which was created using html\_url function. The only difference is ltrim for get\_url\_pic\_id trims the ?pictureid= and admin\_get\_userid trims the ?userid=. One returns the picture\_id which is used for getting an admin to view a particular user’s picture when clicked on or when a user wants to view their own picture. While, the other is when a admin clicks on a user’s link in the search, the user\_id allows them to view all their pictures.

The create functions is a list of functions where the user uploads their content to the website. This page only has 2 functions, create\_album, and upload\_picture.

The create\_album function is used called in the album page. It takes 2 parameters an album name, and the cover photo location. First it determines what user wants to create an album based on their session user id. It then takes the 2 parameters and inserts them into the database.

The next function upload picture the picture a user entered into the database. It is called in the upload page. It takes 3 parameters caption, location, and album id. First it determines what user is entering the picture into the database with the session variable user id. Then it does a query to get the album id of parameter 3. Finally, it does another query to insert parameter 1 and 2 and the album id of parameter 3.

The edit functions page is a list of functions and is used when on the picture profile page when the user wants to edit their picture. That page has 3 buttons that portray 3 functions here, delete photo, change caption, and move albums.

The delete photo function has no parameters and is used when the user clicks the delete button on the picture profile page. This function first gets the picture id from the url using the get\_url\_pic\_id function from get Functions. Then it does a delete query and deletes that picture from the database based on the picture id.

The change caption function has 1 parameters, the caption that will be changed via Post. This function changes the caption based on what the picture id is from the get\_url\_pic\_id function from get functions. It then does an update query caption to parameter 1 where the picture id is equal to the value in the url.

The last function move albums takes 1 parameter, album via Post. This function changes the album\_id of the picture currently displayed. Essentially, it moves a picture to another album. First it gets the picture id from url using get\_url\_pic\_id from get Functions. Then it needs to do a query of the parameter album name to find the album\_id in albums table. When we get the album\_id we can do another query and update the album id to album\_id that we found where the picture id is the same as the one in the URL.

## Album Submit & Upload Submit

The album submit page is accessed when the user submits the album page. Admins cannot access this page since, they do not have pictures, so give them an error. Next we check if everything in album page was filled in using Post and Files variables. If not redirect them back to album page. If so, create an array of allowed extension files, and use the explode method to get rid of the file name after the dot. The only thing left is the file extension in the variable. In the other variable get everything before the dot. If the extension is not in the array then the file type is not supported and give an error. If the extension is in the array then the file type is supported. Next we must check the file size of the cover photo using files variable size. If it is bigger than 10MB, give an error. Otherwise copy the current location of the cover photo to the new location. If the copy was successful then call the create\_album function from create Functions. If copy was not successful give an error.

The Upload submit page is nearly identical to the album submit page. It is accessed when the user hits submit in upload page Admins cannot access this page since they cannot upload pictures. Check if everything was filled out in upload page. Create an array of allowed file types. Get the extension using explode method. Compare if the file type matches the uploaded file type. If yes check the size of the file to make sure it is less than 10MB. If it is less than 10MB copy the uploaded file from one current location to new location. If the copy was successful then call the upload\_picture method from create Functions. If during any point the condition was not made then give an error.

## Javascript Functions

The javascript functions used in qwickPic are used for the visual representations on the site. They are dynamic, so the page does not need to be reloaded for them to work. The type of javascript framework used is Jquery to make things a bit easier. The functions included in this file is the sidebar open/close, sidebar close button and the sort pictures function.

The two functions sidebar open/close and sidebar close button are essentially the same function. Basically, sidebar open/close is a clickable span field that open and closes the sidebar. This span is located on the navigation bar throughout the site for all users. The sidebar close button is an actual button that simply closes the sidebar. Since, the button is located in the sidebar you can only close it because when you close the sidebar, the sidebar with the button is gone. These functions have a click event that is stored when a particular item is clicked (recognized with the id of that item. Once that item is clicked it simply turns on and off the class visible of the sidebar (in the css file. The default sidebar has a display of none, meaning it is not on the screen, but once it is given the visible class, the display is set to block meaning that it in block form and visible to the user.

The next function sort pictures first looks for a change of the selected value in the drop down menu in the pictures page. Once it registers the change. It gets the value of the album name selected and stores it in a variable. It then uses ajax to store a post method into the sort album function in display functions.It uses the stored variable as a parameter to call that function. Then it hides the original div that displayed all the original pictures. In the new div it stores all the html it received from the function, so that when an album is selected, it dynamically changes all the pictures that are in the selected album to be displayed on the screen without refreshing the page.

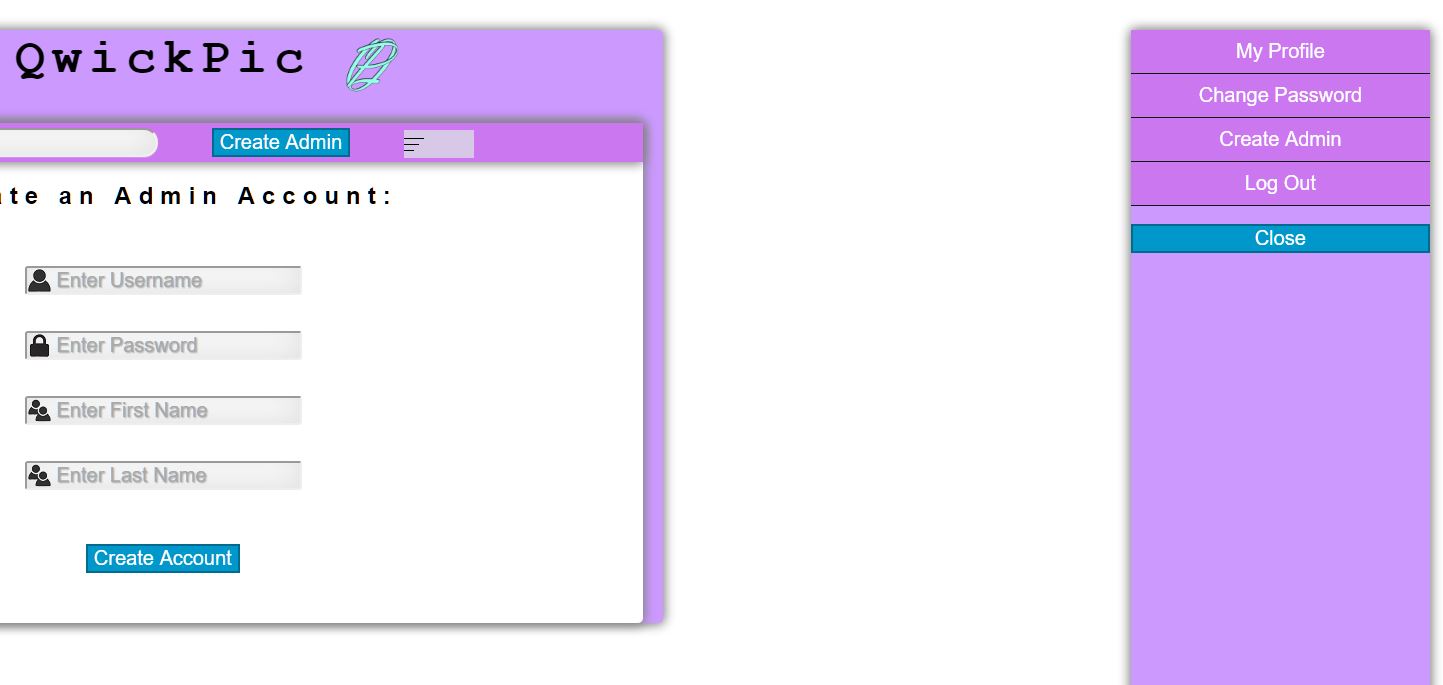


Figure 13: When the button is clicked it opens a sidebar



Figure 14: when album is clicked on. It only displays pictures in that album dynamically.

## Logout

The last script, but certainly not least is the logout script. When we called the login function and had a successful login, we set session variables to, so that we can call tasks easier since, session variables can be accessed throughout the pages. However, when we are done with our session we need to be able to log out. Logging out destroys all variables. If we do not destroy the variables then the user did not log out because they can still do things like logged in people can do. This is why we must destroy the variables. This page is called when a user hits the logout button in the sidebar or on the home page. First this page checks if a session variable exists for either a user or an admin. If a session exists then all session variables will be unset, the session will be destroyed, and the user will be redirected to the index page. As we said earlier, the login header checks if you are logged in with session variables. If you are logged in then you will be redirected back to the login home page, but since we were able to get back to the index page that means the session variables were destroyed and we have successfully logged out.

# Database Layout

## Accounts Table



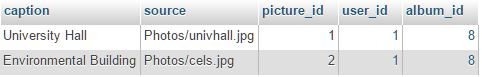
The accounts table is the table that stores all normal user’s accounts. Without this table we wouldn’t know how to identify a specific user. If a new user account is added, it will make a new entry in this table. The first attribute is username, which is the first identification the user must provide in order to login. Each username is unique meaning that there will not be two entries in the table with the same username. Each username is consisted of lower case letters and at least 5 characters in length. The next attribute is password. This attribute is important because it helps identify that the user trying to login is indeed that specific user. Notice that the password field is a combination of really long random letters. This is because it is unsafe to directly store the password string in the database if a hacker is able to get in. The password is encrypted using password hash (discussed earlier) to prevent this. The next attribute is the user\_id, a primary key that identifies each user. This is a unique number that increases by one for every new entry (account) created. This will be linked to the albums and pictures table to show which album and picture belong to which user using the user\_id. The next attribute is the email. This attribute is important for mailing the user a temporary password if they forget their current one. The final two attributes are first and last name which simply display the user’s actual name on the homepage when they log in.

## Album Table

album.JPG

The albums table displays all the albums that exist on the qwickPic website. The first attribute is the album name which helps a user search for a particular album in the search bar. Album names are not unique and there can be several entries in the table with the same album name. The next attribute is a cover photo which displays the image when an album is searched for. In the database table the cover photo displays the file path for the cover photo which when needed will be outputted to an the actual image. The next attribute is a the primary key which identifies the album, album id. It increments by one when a new entry is entered in the table. This will be used as a foreign key to in the pictures table to identify which picture belongs to which album. Finally, the last attribute is the user\_id foreign key from the accounts table. This attribute identifies which album belongs to which user based on the same user\_id. For example, the album Montclair belongs to the user whose user\_id is 1 in the accounts table. This happens to be shadowarty, so the album Montclair belongs to shadowarty. A user\_id can only be a number that exists in the accounts table. For example, the album Montclair can only have a user\_id of 1 or 3 because those are the only two accounts that exist in the accounts table at the moment.

## Pictures Table



The final table is the pictures table. This table stores all of the pictures that are uploaded to the qwickPic site for every user. The first attribute is the caption. This attribute is just a unique description of your picture. It will be displayed in the profile picture page. The next attribute is the source path of your file. This simply a path to eventually display the images on the My pictures page. The next attribute is the primary key, the picture\_id attribute. This attribute is important because it can uniquely identify each picture and access it when needed. The next attribute is the user\_id, which is the foreign key from the accounts table to specify which to which user the picture belongs to using the user\_id. The user\_id value can only be an id number that already exists in the accounts table. Finally, the last attribute is the album\_id attribute. This is another foreign key. It specifies which album that picture belongs to according to the same album\_id value in the albums table. The album\_id can only be a number that already exists in the album’s table for the album\_id.

## 

## Admin\_accounts Table



The admin\_accounts table is composed of all the admins accounts that exist in qwickPic. The reason why admin accounts needs its own table rather than combine the two account tables is because we need to be able to distinguish between the user and the admin, since the admin will have a different view when they log in. The first attribute is the admin username which is unique to both the admin and regular user’s usernames. This means no admins or users can have the same username. Admin username is the first credential a user will type in when they are trying to log in. It is their unique identifier. The admin password, just like the user’s password is there to make sure that the user accessing the account is indeed that user. The next attributes is the admin’s first and last name. This is simply needed to display a welcome message for the admin when they log in to the homepage. Finally, the last attribute is the admin\_id, a primary key which increments by 1 everytime a new admin account is created. Admin Id is needed to give every admin a unique identification.

# How to Upload QwickPic

* Create a database called “qwickpic” in PHPmyadmin
* Import the sql file, “qwickpic” in this folder to the database created above
* The sql file should already create a new user in PHPmyadmin, if not then create a user manually with the following, **username:** Arthur\_Rozenberg **password:** Chris\_Dahdouh
* You are now ready to use qwickPic, start up the php files and the database is now connected

# 