FURZE PLATT SENIOR SCHOOL

COMP4

Photography Management System

Myles Rankin

Candidate No. 7181
Centre No. 51519
Furze Platt Senior School

Contents

Analysis

Project Definition

Client: Furze Platt Photography Group

- Jake Smith (Primary contact, client)

Contact: smitj001@furzeplatt.com

+447868433563

Background to/identification of problem:

Furze Platt Senior School is the local school I attend to in Maidenhead. I have been approached by a group within the school that does all the photography for the school. They are responsible for going to events, taking pictures within school, etc. for publications and other uses. When talking to my user Jake Smith, he has described the current system and problems he feels needs to be addressed. The group currently takes a massive amount of photos from these events and such, but has no actual place they store and manage them. Currently, the system is relativity simple, inefficient and inadequate with photos being lost and not all in a centralised location. This causes problems when, for example, Mr Bown wishes to write a school publication and request for photos to use, because the photos are usually lost or on someone else's PC or even still on the memory card they were taken on. So really, it's very hard for people to access content the photography group has created, and this is the underlining problem.

Additionally, a few other sub issues have been brought up due to a lack of a formal system to store and manage the photography content. Mainly this resolves around parents, pupils and other outside users wishing to see photos taken at events, which currently is impossible because the only photos people are able to see are those that have been published in the school newsletter and weekly email update from parent mail. So an ideal system would be some sort of gallery that wold be situated on the photography management system, that would be accessible to parents, teachers and such.

My client is Jake Smith, a fellow student, who has a senior role in the photography group. His main duties involve organising events, collaborating/storing photos and then making photos available/distributing them, so they are used in school publications, etc.

Interview with Jake Smith about current problem and discussing about a solution:

Q = Question (me), A = Answer (Jake Smith)

Q: What current system does the group use currently, to store, manage and distribute photos from events, etc.?

> A: As of current there is no formal system the group uses to collaborate photos, and organise in a centralised place. Currently, the system consists of people exchanging photos after events by emailing, sending by memory stick or by just giving the memory card from the camera to whoever needs it.

Q: So the photos aren't actually stored anywhere that is safe or accessible by everyone in the group?

A: No, they are not. That's a huge problem I find, as none of the photos get backed up, they also end up anywhere on whoever's computer they get downloaded to. This means I can't get access to photos taken at an event, only If I find the person who took them and ask them to make a copy. I find this makes problems with data protection too, as some events involve school children which means the photographers shouldn't have the photos being stored on their personal computers in a possible unsecure environment where they can potentially be accessed by anyone.

Q: On the subject of drawbacks, what overall drawbacks does the current system have?

A: Well, the main problems were discussed previously, but in addition to that the current system doesn't allow for big files to be transferred by email, as we have limits in school. This has been a problem when we want to transfer a lot of photos at one time, so it means we are forced to use memory sticks and such, which usually get lost and are really inconvenient to use. Another thing I find really irritating is finding photos from past events, or specific sizes there is no way to search/filter the photos I do get, so I spend hours sifting through them finding for specific photos.

Q: Are there any actual benefits of the current system?

A: Not really, I suppose that the only benefit currently is that the system is mostly simple for people to use as it's only emailing around in reality, but this means it's very vulnerable and not robust.

Q: So, for a new system what critical features do you think would be need the most?

A: First of all, a place where all the photos can be stored and accessed centralised is the most crucial part of a solution to the problem we are having. Bridging off of this, I think the centralised system will need to be secure and 'locked' out so to speak from certain people, so a sort of private/public system would need to be made in order to allow access to photos depending on different group types. As mentioned before, a search system will be another vital tool the system will require, this will be a massive convenience to everyone using the system.

Q: After the vital features, what other things would you like to see the system have to make use of it enjoyable, user friendly and easy?

A: In addition to what is vital, I think having a public gallery for external people to view allowed photos would be great, also a comments system to go with this to allow for registered users to comment on each photos. The gallery would be sorted into albums of different events and would have a slideshow feature to allow ease of viewing of the photos. A blog system would also be great for posting news/announcements, along with a calendar for posting events that photographers will be able to see when they are needed. Lastly, some image processing tools would be great, but not vital, a way of automatically resizing photos and such would be great. I also think a help prompt, or a faq page would be an added bonus seeing that some of the photographers aren't necessarily computer competent, therefore it would help them use the system.

Q: How many people would you expect would be using the system?

A: The system would be used by the students in the photography club mainly, but also some teachers such as Mr Bown would use it for gaining access to the library of photos for school publications and other journalistic purposes.

Q: What kind of solutions would you be interested in? I.e. web or application based?

A: I'm interested in the solution that allows for the most compatibility and accessibility to my users, I can predict that a web based solution will be the most successful avenue to look into. I can see that an application based solution will be hard to distribute and will have compatibility issues within and out of school.

Description and observation of current system:

Currently the system is very simple and doesn't always function the same. Usually when a photographer has been to an event and taken pictures, they will store them on their own PC or even leave them on the memory stick. Sometimes the pictures, or some of them, are sent to the various teachers who are relevant via email or memory stick/card. For example (as seen on the right), Mr Bown will request to receive photos via email from the school celebration evenings in which, he uses for the school newsletter.

Email evidence – Mr Bown Photos for celebration evening

Glen Bown <bowng001@rbwm.org> 22 September 2013 12:39
To: smitj017@furzeplatt.com, hussh006@furzeplatt.com

Dear Both,

Can you help. I am just writing this week's parents newsletter and I don't have any photos of this weeks event. Any chance of just sending me one or two asap.
Thanks if you can help.
Glen Bown

Mr Glen Bown
Deputy Headteacher
01628 625308
Furze Platt Senior School : a company limited by guarantee
Registered in England: Company Number: 7834715
Registered Office: Furze Platt Road, Maidenhead, Berkshire SL6 7NQ

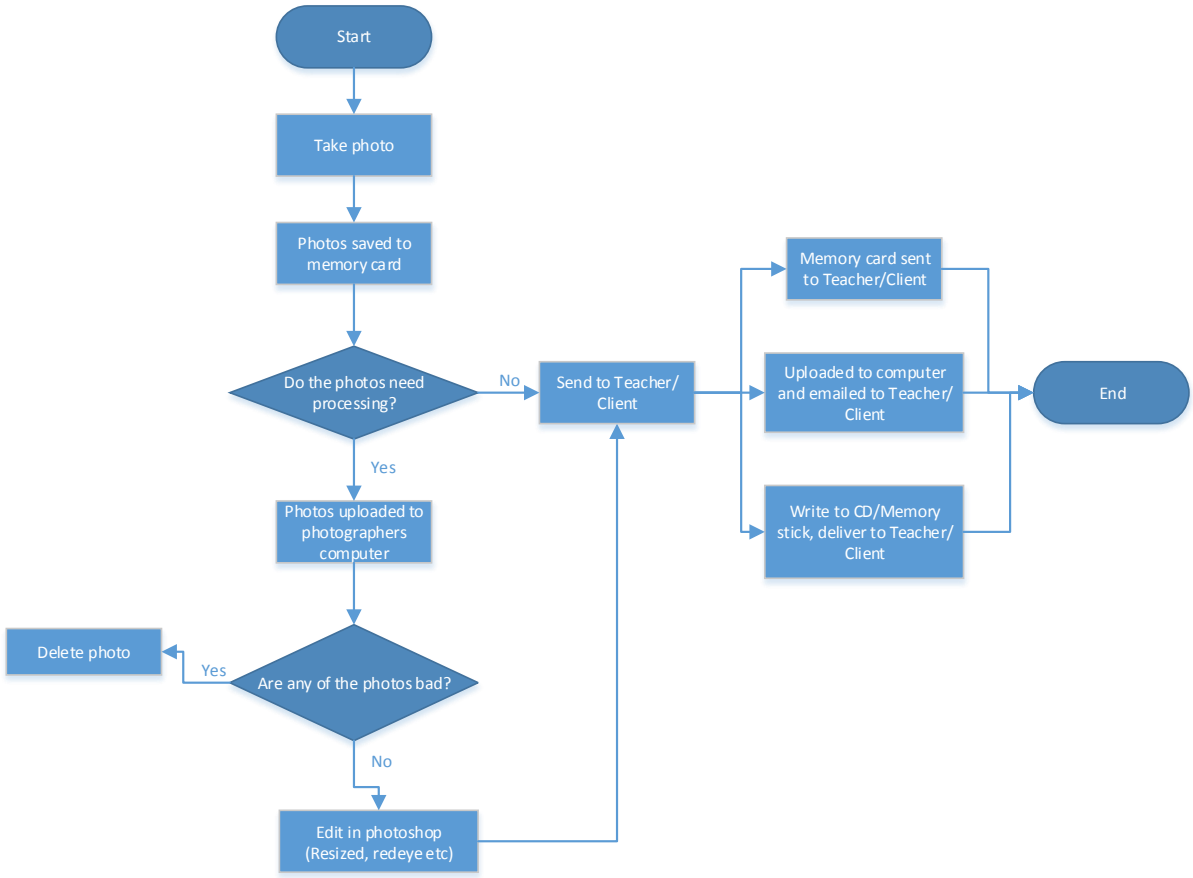
Jake Smith <smitj017@furzeplatt.com> 22 September 2013 12:48
To: Glen Bown <bowng001@rbwm.org>

Hamza currently has my memory card with photos on, so you would need to ask him. I'll send him a text, hopefully he'll be able to email you with the photos.

Current process broken down is as follows:

1. A Photographer is assigned to an event, takes pictures at event
   1.1. When the photographer is finished at the event he/she takes the camera home/to school
   1.2. The photos are then transferred to either:
      i) School computer
      ii) Personal computer/laptop
      iii) Teacher via the cameras removable memory card
   1.3. The photos are then either:
      i) Emailed to relevant persons
      ii) Stored where they are for later use
      iii) Transferred to memory stick or CD then given to relevant persons
      iv) Edited, processed, resized, etc… then either i , ii, iii

I made a flowchart when observing my client use the current system, to enable me to examine any inefficiencies:



As you can see in the flowchart, there are a few inefficiencies and problems. First of all, there are multiple methods of delivery to the end which makes the process complicated and unreliable. Additionally, there is at no point a centralised place of storage for the photos, this causes issues with data protection, backups of the files and it's very restrictive on who can actually use existing photos as they aren't physically accessible to all the photographers. Furthermore, the processing of the image from being taken to resizing to sending is done in too many steps in different places. This shows why a new system that is centralised will work more efficiently because everything can be done in one place, without having to move photos around and it allows for more accessibility and redundancy/protection of files.

Identification of prospective user(s):

- Photography group/Photographers
- Teachers
- Parents
- Pupils

Identification of user needs and acceptable limitations:

As part of identifying user needs, I decided to do research on any existing solutions to the current problem or similar systems. I found out the following key points of other systems:

- Searching – A field to filter photos was present on most of the systems
- Albums – All photos were organised into albums which were accessible on a single page, in which when you opened one it would open all pictures on a single page from that album.
- Statistics – Each photo seemed to have some stats varying per system, from amount of views to how many star rating each photo was.
- Comments - There were commenting capabilities on photos, which allowed you to leave a comment under your name by filling out a form underneath the photo.
- Resolution – Some systems had the ability to choose a different size resolution from the original image size.

After talking with my user he came up with the following needs:

- A permissions system for different levels of access when viewing/editing photos.
- A tagging system for photos, to allow users to identify/search for specific people/groups.
- Search function to filter photos by name, date, tags, location taken and resolution. (Make use of EXIF data)
- Booking system for photography events.
- Private/public photos (see permissions)
- Public gallery (see above)
- Comments system for photos
- Blog system for news posts
- Slideshow photos

Acceptable Limitations

- My user suggested making a more advanced tagging system that tags the actual photo with some sort of overlay, which means you could hover over someone's face and it would display a name. I believe this will be too complex for me to implement in the timeframe, as I have to learn PHP from scratch first.
- Mobile site functionality will problem be too hard to test and create within the timeframe I have been allocated, and has too many problems with compatibility issues with school infrastructure for testing and also different phone browsers may display differently.

The aim of this project will be to provide a clean, usable, function and robust system to store and organise content created by the photography group. At a core level, I hope to have a system them allows users to upload content freely, to also browse other peoples content on the system but based on a permission system to protect images that may need to be restricted due to data protection. Additionally, a registration system will be required to make the system secure and only accessible by those whom are approved.

Data source(s) and destinations:

Data source	Who provides it	Data destination	Who receives it/Where is it displayed
Photos	Photography users	Webserver/Processed	Users browsing

		to database	photos/Displayed in gallery
Events	Senior Photography users	Database events table	Users/Displayed on portal page in the calendar
User account details	User	Database user table	Database receives it, some of the user data is processed by the system and some displayed throughout the system.

Data volumes:

Types of files processed:

The system will have to process a large amount of image files (PNG, JPG, JPEG) this will probably accumulate in the 100's of Gb's of space over the extended use of the system in just raw images, they will also take up space being catalogued within the database.

By looking at my user's photo directory, I was able to extract some statistics on the volume of data each person would use on average. He had 9,014 photos, I used Windows PowerShell to calculate the average file size of all the files with the following script:

$foo = (Get-ChildItem -path "O:\Dropbox\Previews" -recurse | measure-object | select -expand Count)

$bar = ((Get-ChildItem -path "O:\Dropbox\Previews" -recurse | Measure-Object -property length -sum).sum /1MB)

$avrg = $bar / $foo

$avrg = 2.60273053510136

The average, as seen above worked out to 2.6MB (rounded) per photo. The collection was made over 3 years, so he took roughly 3,000 photos per year. That's 3000 * 2.6MB = 7.8GB for 1 user per year, there are currently 18 photographers so 18 * 7.8 = 140.4GB per year. Next you have to factor in database storage, to be conservative this will be 0.5GB. Additionally this system will be built with expansion and growth in mind, so 250GB to start off for the first year is a good estimate to give some space for expansion and potential growth of new photographers, it will also allow for the system to generate a compressed weekly backup of all photos and store it temporally on the server for download.

Number of users accessing the system:

The system will have to be able to cope with all the photographers, teachers who require content for publications, most If not all the senior staff and administrators to manage and maintain the systems.

There are 18 users in the photography group who require access, senior staff are around 10 and then I estimate 15 other users that the system will require to handle. So in total 45 users, which will all have data saved into the database on a daily basis. The system will also need to be designed with expansion/growth in mind, so initially I wish to allow the system to support/store data for up to 100 users.

Frequency of user access:

Users will most likely be accessing the system daily, mainly viewing photos and event timers. I expect that some users will be uploading content to the system weekly; all based on based on what current events are taking place.

Objectives for the Proposed System

From observing the current system, I have found out the inefficiencies, problems and upsides of it. When analysing it, making use of flowcharts and talking to my client I was able to collaborate an optimal set of features for my solution and I can now stipulate a set of objectives and sub-objectives for the solution.

1. The system must be able to store photos uploaded by photographers
    a. The system should record all data related to the photos during upload for use throughout the solution for searching, indexing albums, etc.
    b. It must also store the photos securely.
    c. Security must also contain a permissions system restricting access to the stored photos for different groups, a public and private gallery will be required.
    d. System will need to be able to store the amount of file space the photos will be taking up, the amounts are mentioned in data volumes analysis. So per user 3,000 photos at around 8 GB of file space per year, of course the system actually has no limit because it will be designed to be able to expand if storage is filled up.
    e. When uploading photos, duplicate photos (when filename is the same) need to be renamed.
    f. When uploading, the system will have to be able to cope with a limitless amount of files being uploaded and there will be no limit on the size of files being uploaded.
2. The system will need a login system where each user is given login details to access the system
    a. The login details will need an approval system to allow admins to control who can register and gain access.
    b. The login accounts should be tied to the permissions system, with perms being given by a per user basis requiring login to access their given perms.
    c. Login should be encrypted, passwords during login and registration should be stored hashed and not in their plaintext form, login will have to compare hashed passwords instead of plaintext when verifying login.
    d. The login system will need to be able to support the amount of photographers, teachers and other users that may need to use the system. So it must be able to support at least 45 people as mentioned in my analysis, but the system will ultimately be designed to support an infinite amount of users as the server space can be expanded to store more user data.
3. A registration system will be needed to go with the login system.
    a. Registration will need to be secure as mentioned in 2.c, passwords will need to be stored securely.
    b. Registrations will need to be restricted to only certain people, or If you register without being approved, you have no permission level to view photos/albums, only view public galleries.
4. Users will need to be able to manage their account.
    a. They will need to be able to:
        i. Change their password
        ii. Update their permission level with a key given out by administrators

iii. Upload/change avatar

5. A searching function will be required to allow users to search for photos based on information pulled during upload (see 1.a).
    a. EXIF data could be used to pull informations (consider for 1.a) for photos when searching.
    b. Searching parameters should include: resolution, tags, string search, date, album/event name, by photographer and room to add more.
        i. The tag searching would include a system to be in place already to tag photos with different words/phrases etc. during upload or when being managed.
6. As mentioned in 1.c and 2.b a permissions system will be required to underpin the system.
    a. Perms will be tied to each account.
    b. Photos and albums will have a required permission level to be able to view them.
    c. Additionally permission level will be used to restrict access to administrative privileges such as user management.
7. Event calendar to allow photographers to post events for other users to see where they are needed.
    a. Additionally add a system where photographers can tick the events they are able to attend/or are attending.
8. Image processing should be implemented into photo management
    a. Resizing is a key processing element, automatically giving the user multiple sizes that the system would resize when uploading.
9. User should be able to manage their photos they have uploaded.
    a. They will need to be able to:
        i. Delete
        ii. Make private
        iii. Image Process – resize, etc.
        iv. Have them put into their own user album
10. The solution will need a help/FAQ section, popup(s) or page to help the user use the system.
11. The system should have features to ease maintenance.
    a. A statistics page must be included to show administrators statistics such as, amount of files uploaded, amount of users and % of disk space left on the server for storing files.
    b. An automatic backup solution that backs all photos up periodically (i.e weekly) OR a sub program that an administrator can initiate that allows them to manually active a backup to download.

Potential solutions and justification of chosen solution:

Potential Solutions:

1. An application based solution, this would be a desktop application that the photographers would download and install onto their computer. It would allow for uploading via a UI the application running on their computer, which would then upload the photos to an external server where the files would be stored on. At the same time the application would make use of databases, inserting information about photos to a DB and also using a DB for a login system. This application I would most likely program in Java, mainly because of the cross compatibility it gives, it will run on all operating systems. Additionally, it's an easy language to pickup and a very commonly used one therefore it will have a lot of support and it does currently have a ton of libraries I would be able to use to create the system (i.e. MySQL library), which saves time and allows me to work on other aspects of the project.
    - Although Java is an easy language to pickup it will still take me a while to pick it up from scratch, so it might be wiser to use a language I am already familiar with.
    - Distribution may be problem, I would have to ensure all photographers and users are able to download, install and use the solution themselves which will introduce a whole range of problems. This would range from people not knowing how to download/install, to other issues such as restricted access running programs on computers at school or their home computers which could be, for example, restricted by parental controls or they may not have administrative access over their computer.
    - The system would also take up space on the user's computer, as it may have to cache photos they are viewing therefore using up HDD space which they may need or not have. It will also use a lot of bandwidth when viewing photos from other users, and taking up more space caching them.
2. Another application system I have considered is a tablet/mobile app solution. When looking into at some of the cameras that the photographers use, some of them are able to connect a tablet device to their camera and automatically download the photos to the device. I would be able to make an application that automatically fetches the photos being taken from the camera and saved to the device, then automatically sync them to a centralised server. The application could then be on any device that has the application and would be able to browse photos that have synced. If I was to go ahead with this solution, I would most likely use the android platform for development seeing as there is a lot of support for it and android is one of the most popular phone and tablet OS's on the market.
    - I found that not all the cameras were able to connect a tablet, due to compatibility issues and how new the camera was, so this would create a problem If the camera didn't support tablets.
    - Not all photographers would necessarily own a tablet or device/smart phone that would be able to install the application, therefore it would limit the amount of users able to access the system.
    - I would additionally have to develop the system for different operating systems, for example some people will own iOS devices and others Android, so this would create extra work having to learn two different development languages (so essentially I would have to create two different applications that do the same thing, but for different devices).
3. The other solution I have thought of would be a web based solution. This would comprise of a website with a login page that allows users to login to gain access to a photography management interface where users can upload photos to the site then manage, image process and share the

content they have made. This system would be the most optimal as the compatibility of websites are brilliant, as any device or computer that has a browser will be supported. I would be using PHP If I went with this solution, seeing as I have past experience with the language. It is additionally, the most popular and powerful web programming language that exists and with this comes a huge amount of resources online I can access to support me when making the program. Furthermore, there are plenty of libraries/functions already built into PHP that are designed for the processes I am thinking of using, for example there are already MySQL, uploading, image processing, hash and many more useful functions built in already.

- Only downside I could find would a webpage solution would have the images stored on an external server when hosting the website (this may cause data protection issues), but this can be counter acted by hosting the website on a server controlled by the school.

Chosen solution and justification:

My chosen solution is to make a web based program as described in part 3 of my Potential Solutions. After weighing the downsides and upsides of using each potential solution, and then comparing them I felt that firstly it would be easier for me to start developing a solution with a programming language I am familiar with, compared to starting with something I have no knowledge for (thus taking more time to learn).

Secondly, I believe that of all the inspected solutions, a web based project will be the most compatible of the three almost all devices these days have a browser and most If not all browsers will support HTML/CSS which PHP outputs in, as PHP is a server side programing language so the client does not have to worry about compatibility with the language being used.

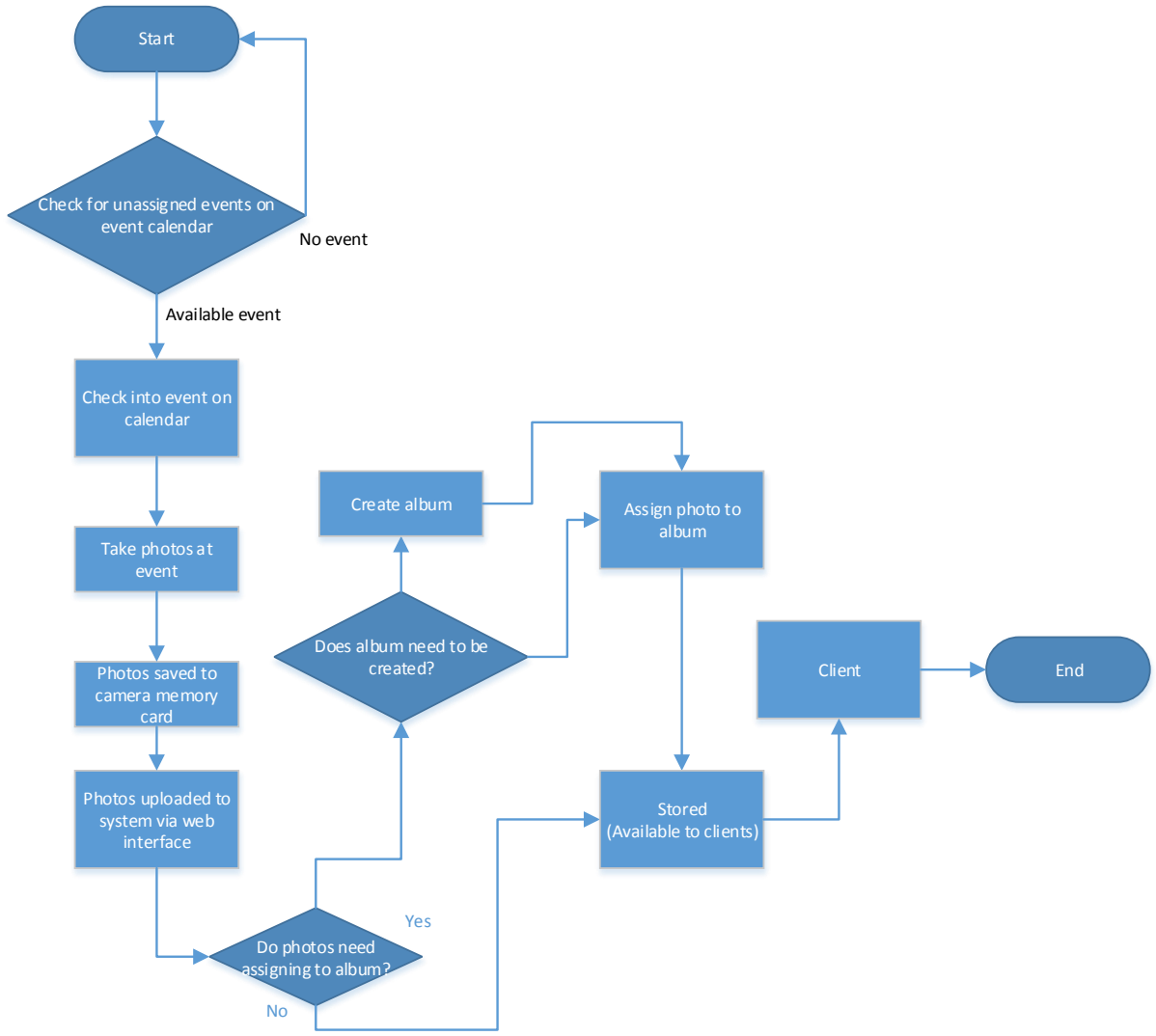
Lastly, comparing each language I have considered using I feel PHP is more powerful and fit for purpose for my given problem, as it has been tailored for similar projects involving login systems, uploads and is heavily supportive of the use of databases having native functions for MySQL.

Data Dictionary:

Data		Data type	Size/Length	Description
Photos		Image file (PNG, JPEG, JPG, GIF)	2.6MB (Average)	Photos that are uploaded from photographers which are stored on the webserver/
Photo data	File Name	Variable Character	Up to 1024	File Name taken from photo during upload then stored to database.
	File ID	Integer	Up to 256	File ID generated for photo during upload then stored in database.
	Path	Variable Character	Up to 1024	Path stored in database that points to the file on the server.
User data	Username	Variable Character	Up to 1024	Username chosen during registration that is then stored in database.
	Password	Variable Character	Up to 1024	Password chosen during registration that is then stored in database.
	First Name	Variable Character	Up to 1024	First Name entered during registration that is then stored in database.
	Last Name	Variable Character	Up to 1024	Last Name entered during

				registration that is then stored in database.
	User ID	Integer	Up to 256	ID Generated for user during registration which is stored in database.
Event dates		Integer Character		Date and times of events that are taking place all stored in database.
Event Name		Variable Character		Event names and uid which attaches to an event date all stored in database.
Event Content/Description		Variable Character	Up to 2048	Content/Description of event from event creation

Proposed System Flowchart



Inputs-outputs Chart

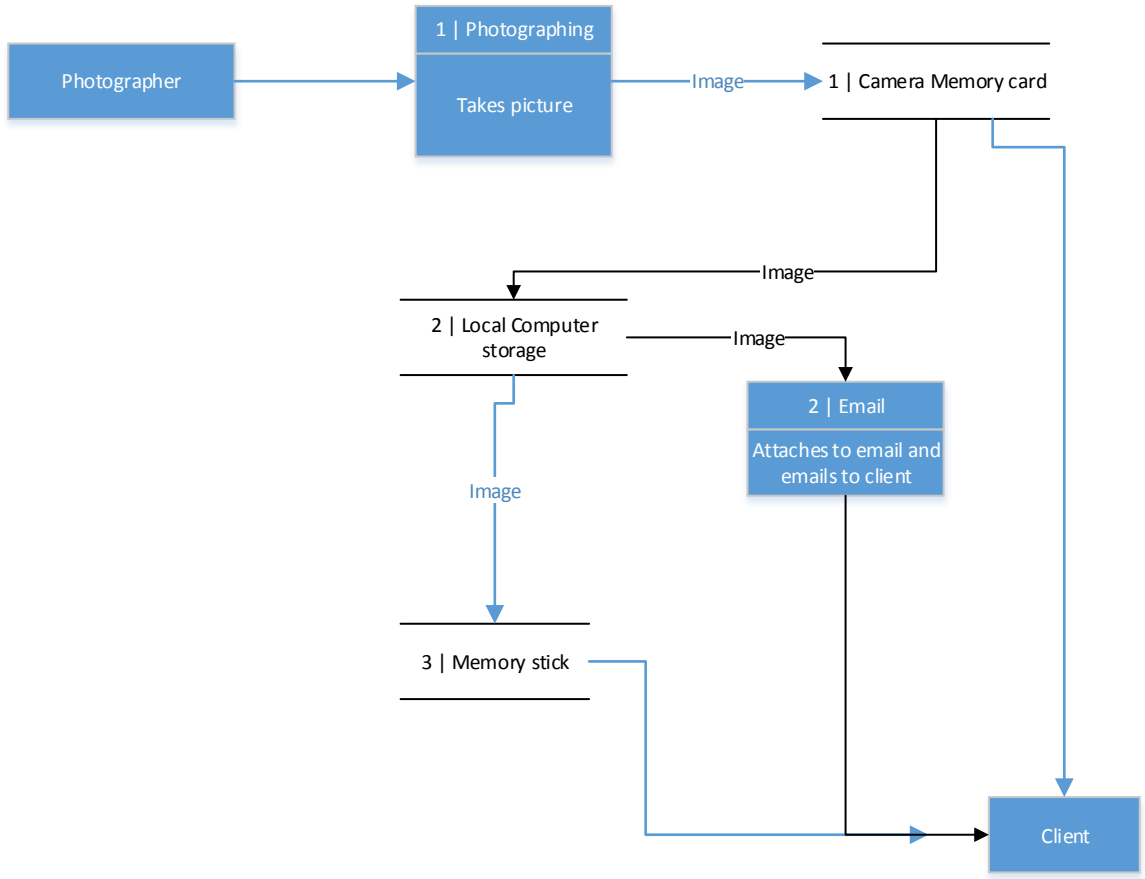
Making an IOP chart allows me to outline what is happening to data in the system on a very basic level in terms of inputs/outputs, processing and storage.

For registration:

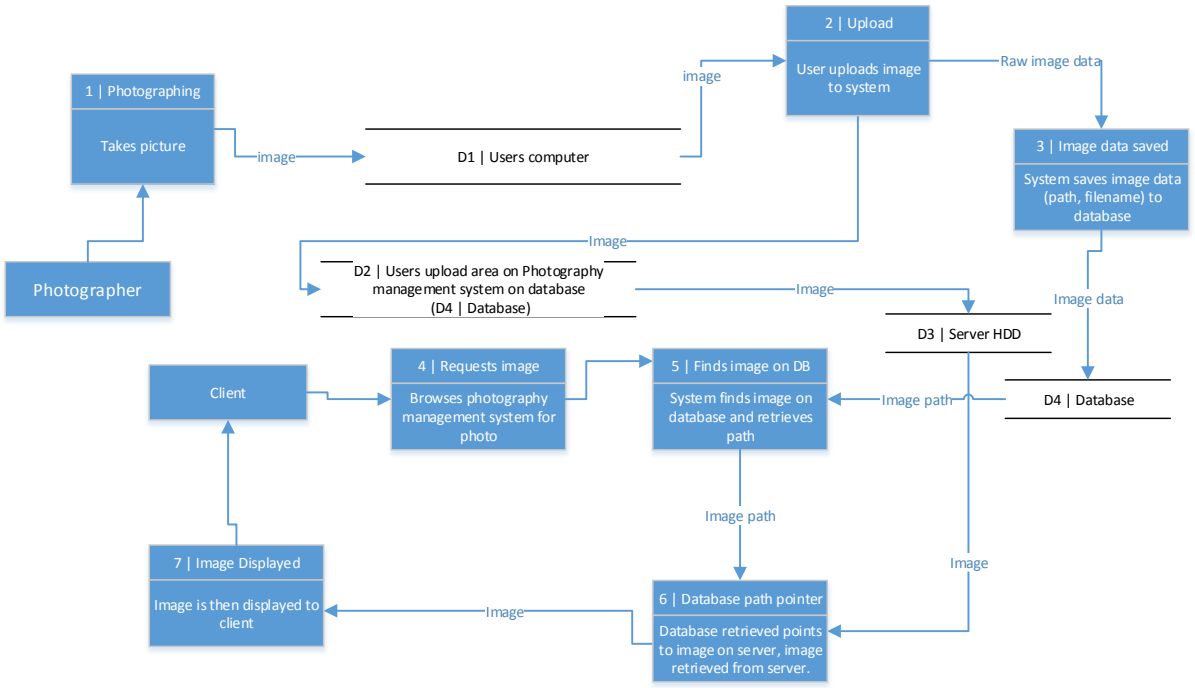
Inputs	Process
• Username (varchar) • Password (varchar) • First Name (varchar) • Last Name (varchar) • Reg. key (varchar)	• Password hashed with a generated salt unique to user. • Username, Password, First Name, Last Name, salt and Reg. key are passed to users database, a user unique ID is generated and stored as primary key for entry • Reg. Key is compared with Reg. Key db table, if it exists the permission level assigned to that Reg. Key is assigned to user and is stored in their row in users db.
Storage	**Output**
• userID (Primary key) -> Automatically incremented when below values are all store into same row • UserName -> Stored to users DB in row bound to userID • Password -> Stored to users DB in row bound to userID • First Name -> Stored to users DB in row bound to userID • Last Name -> Stored to users DB in row bound to userID	• Username displayed on pages when user is logged in. • Username displayed on user albums page as author to the album. • Username displayed on the admin management page.

Data Flow Diagrams (Exist and proposed system):

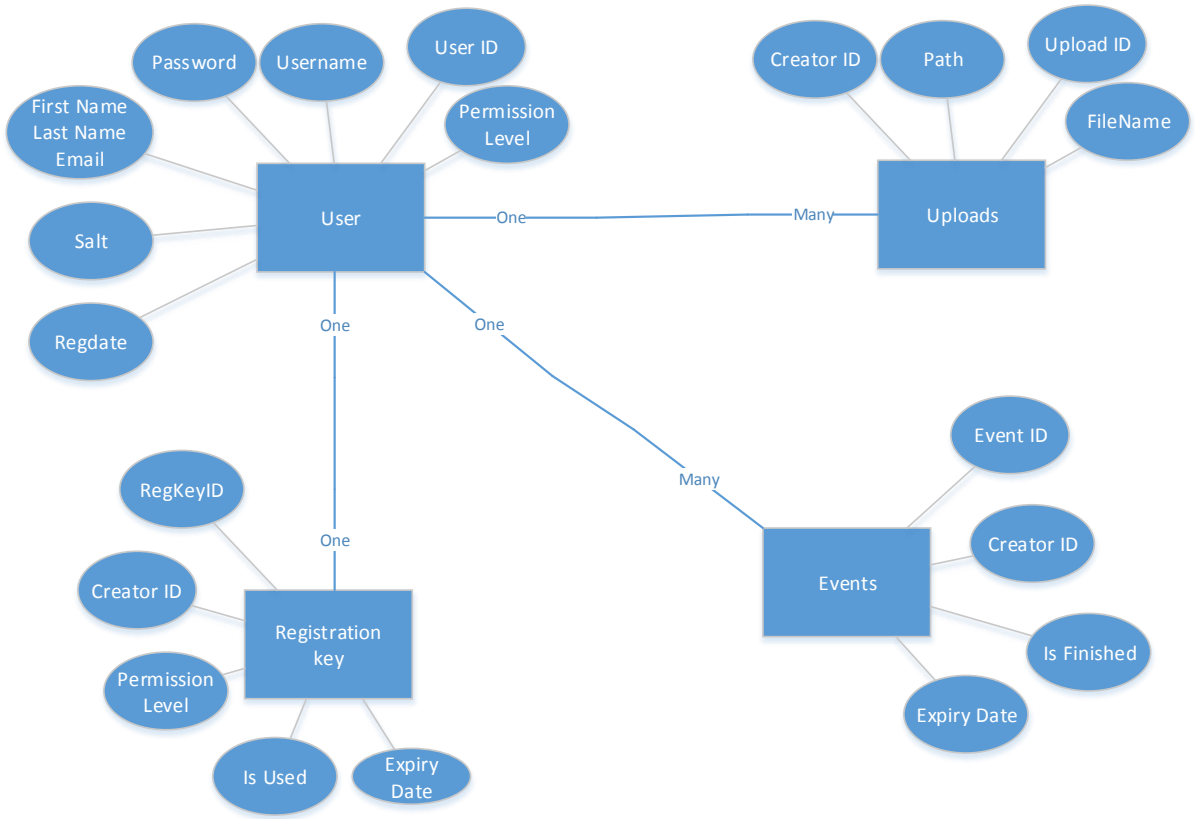
Current system DFD:



Proposed system:



Entity Relationship Model for main database



Entity Description:

Design

Website Index

To start designing the system, I needed to take my objectives, user needs and flow chart processes to create an index of the site to see what needs to be programmed and where. It allows me to assign the features and processes to certain parts of the site.

Page Name	Description
Login	Username/Password login page, where users log into the system.
Portal	News, index page, navigation to other pages, control panel, calendar page…
Photo	A template page that each individual photo will use and be displayed on when viewed.
Account settings	A page to change passwords and details for each user.
Upload	For uploading photos to the system
Search	A search page that allows searches of images by inputting specific search parameters i.e. by Album name – 'Christmas Photos' or even by other predefined parameters like search by resolution/image size, photo age, etc…
Gallery	A location that a whole album will be viewable, or random image viewing for users
Register Account	A page to allow users to register an account

Database planning

As I have planned to use MySQL databases to store user data and other data with the system, I need to design my database structure and plan how I will use it.

In my analysis, I identified what data and types I may need to store in the perspective of the end user. I broke this down to Photos/Uploads, User data, Event dates and Event information. With this in mind, I decided to create 4 tables. The 4th being involved with the RegKey system which wasn't foreseen in my original analysis:

- Members
- Uploads
- Events
- Regkeys

Users	Type

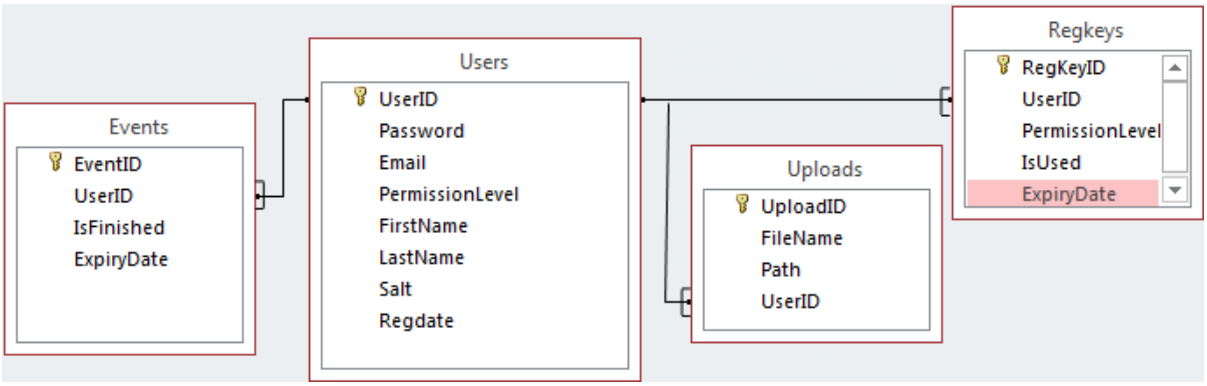
UserID (primary key) [Auto Increment]	Integer
UserName	Varchar
Password	Varchar
Email	Varchar
PermissionLevel	Integer
FirstName	Varchar
LastName	Varchar
Salt	Varchar
Regdate	DATE

Uploads	**Type**
UploadID [Auto Increment]	Integer
FileName	Varchar
Path	Varchar
UserID	Integer

Regkeys	**Type**
RegKeyID	Varchar
CreatorID (*UserID)*	Integer
PermissionLevel	Integer
IsUsed	Bool
ExpiryDate	DATE

Events	**Type**
EventID [Auto Increment]	Integer
CreatorID (*UserID)*	Integer
IsFinished	Bool
ExpiryDate	DATE

This diagram shows the relationship between tables:

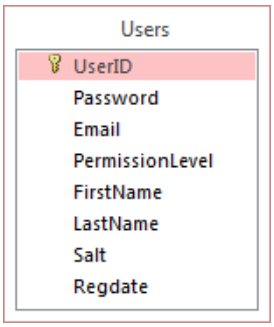


SQL database design

From the plans I made in my database planning, I will need to create queries to submit to my MySQL server to create the tables required for my system. Each query will have to specify all fields/columns needed in each table then each of them will need to be assigned a specific data type, which I have identified for each field in my database design stage previously.

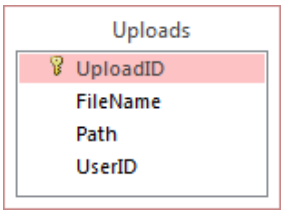
Users table

SQL Query
CREATE TABLE 'Users' ('UserID' int(64) AUTO_INCREMENT, 'UserName' varchar(256) NOT NULL, 'Password' varchar(256) NOT NULL, 'FirstName' varchar(256) NOT NULL, 'LastName' varchar(256) NOT NULL, 'PermissionLevel' int(64) DEFAULT '0', 'Salt' varchar(32) NOT NULL, 'Regdate' date DEFAULT NULL, PRIMARY KEY ('UserID'))



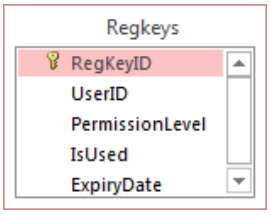
Uploads

SQL Query
CREATE TABLE 'Uploads' ('UploadID' int(64) AUTO_INCREMENT, 'FileName' varchar(1024) NOT NULL, 'Path' varchar(2048) NOT NULL, '*UserID*' int(64) NOT NULL, PRIMARY KEY ('UploadID'))



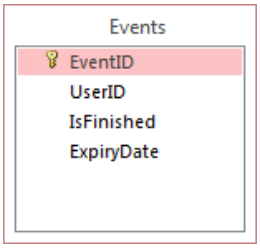
Regkeys

SQL Query
CREATE TABLE 'Regkeys' ('RegKeyID' varchar(32) NOT NULL, 'UserID' int(64) AUTO_INCREMENT, 'PermissionLevel' varchar(64) NOT NULL, 'IsUsed' BOOLEAN, 'ExpiryDate' datetime NOT NULL, PRIMARY KEY ('RegKeyID'))



Events

SQL Query
CREATE TABLE 'Events' ('EventID' varchar(64) NOT NULL, 'UserID' int(64) AUTO_INCREMENT, 'IsFinished' BOOLEAN, 'ExpiryDate' datetime NOT NULL, PRIMARY KEY ('EventID'))



Samples of possible SQL queries to databases

Fetching a list of all members registered
`SELECT UserName FROM users`

Fetching a list of upload names from a user with ID of '5'
`SELECT FileName FROM Uploads WHERE UserID ='5'`

Fetching a list of Event ID's in order of date descending
`SELECT EventID FROM Events ORDER BY ExpiryDate DESC`

Adding a new user into the users table
`INSERT INTO Users (UserName, Password, FirstName, Email, PermissionLevel, FirstName, LastName, Salt, Regdate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`

Searching query for fetching a list of file names from a search input
`SELECT FileName FROM Uploads WHERE FileName LIKE '%search_input'`

Updating a user password with new password input
`UPDATE Users SET Password ='newpassword_input'`

Further SQL Database Design

During the development of the system, two more additional databases were created. These are 'filesizer' and 'posts' tables. The posts table is used in storing administrator posts on the front page of the system, and has one relational key to the userID in users table. The other table is the filesizer table that counts file space used by each user for use in systems maintenance, this table again only has one relational key from the users table being the userID key.

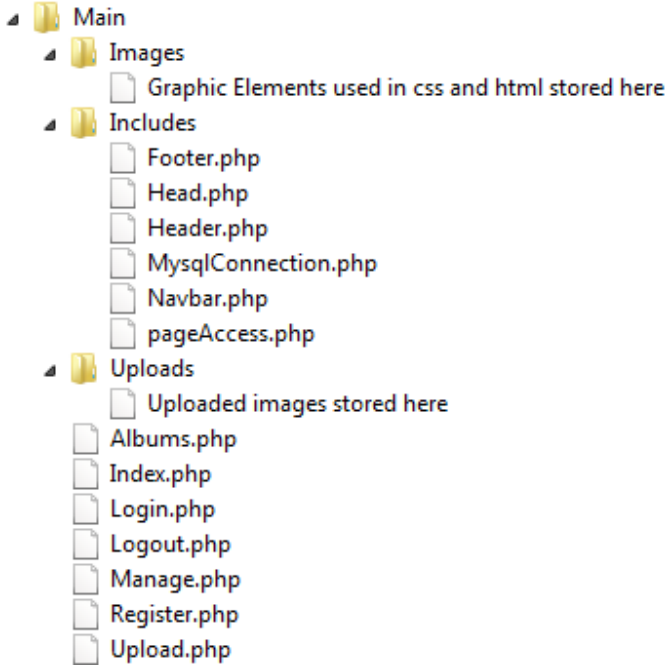
Full diagrams of the final SQL tables can be seen in the systems maintenance section.

System structure design

After doing some preliminary research and from past experience of web programming/design, I wanted to create the system with an easy way to replicate code used to generate the GUI for the pages and also different parts of code that would be reused throughout the system.

With this in mind, I looked into using the php include function that allows me to include an external file into said page. When including another php script, it basically adds the code in that script onto the page you are currently coding on, adding all variables/functions/etc to it. This allows me to, for example add a MySQL connection module that will do connection to the database and will let me include this in each page as it doesn't change.

System file tree



Additionally, using includes means I can easily create new pages by just including the main elements of the page such as a header, content and footer include that will render all the GUI for the page. It also means, when I want to change some of the design or a module, it means when I change the include file, it

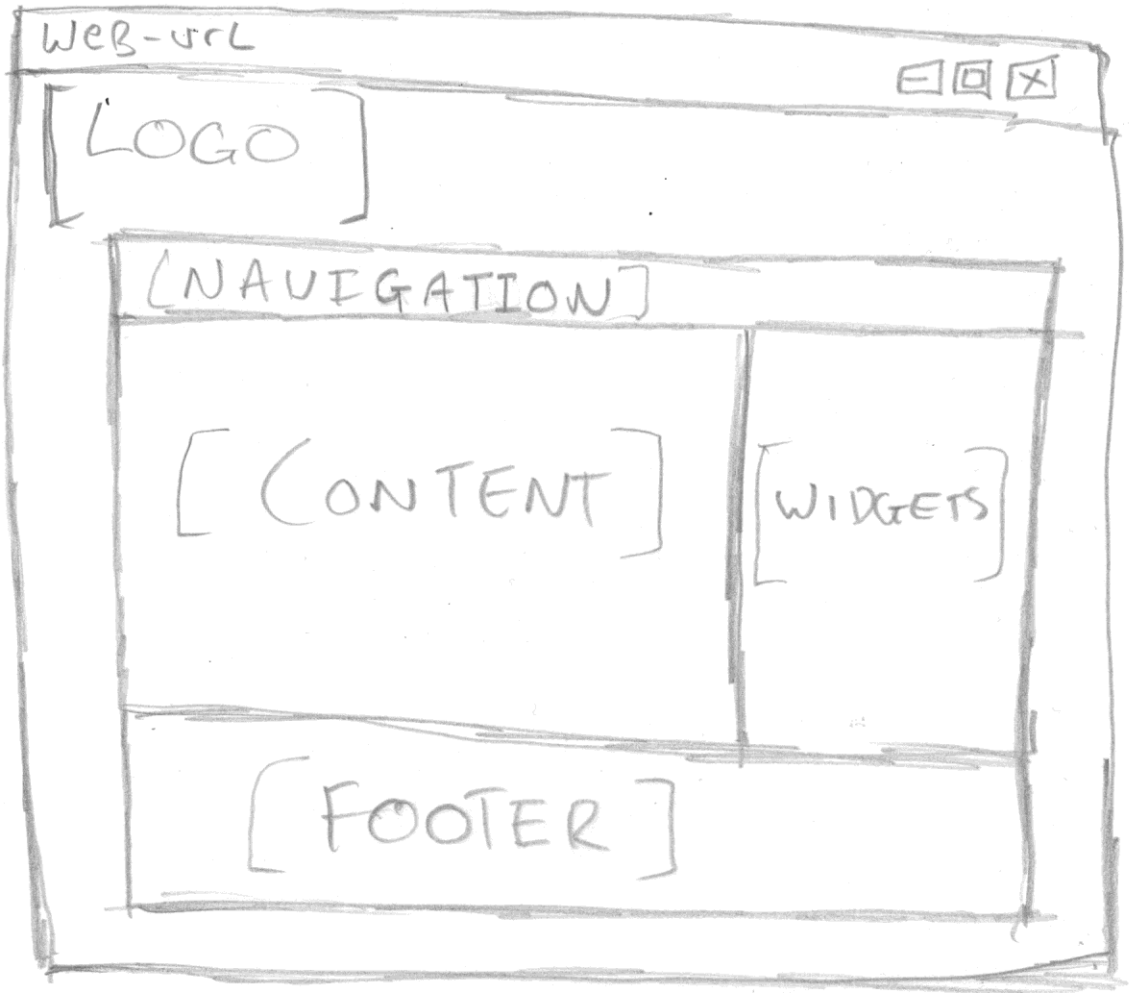
will automatically be pushed to the whole site. This is better than having the same elements in each page, then having to change each page individually, as this takes time and effort.

User interface design

Initial drawings

I first started designing the templates of my web based solution on paper, fleshing features out and seeing where I would position everything to look professional but very functional on a browser.

Firstly after looking at some existing websites to get an idea of some basic layouts, I sketched out a brief page breaking down a webpage to its main elements:



When sketching the layout I broke down the website into 5 distinct elements:

- Logo
- Navigation
- Content
- Widgets
- Footer

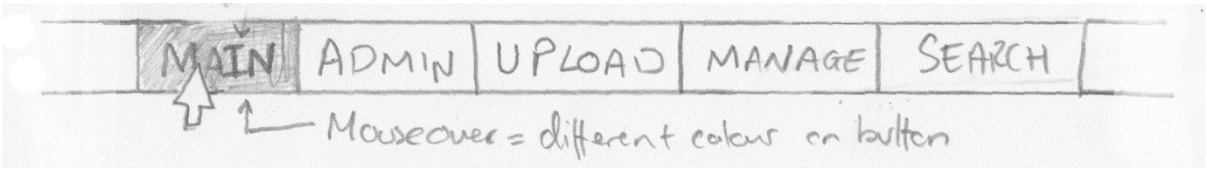
After defining the elements I started to design each individual part, now I felt that separately designing the content and footer wouldn't be needed as they either change on each page (so they will be designed

in each specific page design) and in the case of the footer, it's just a copyright string (i.e Copyright - FP Photography 2014).

Logo design:

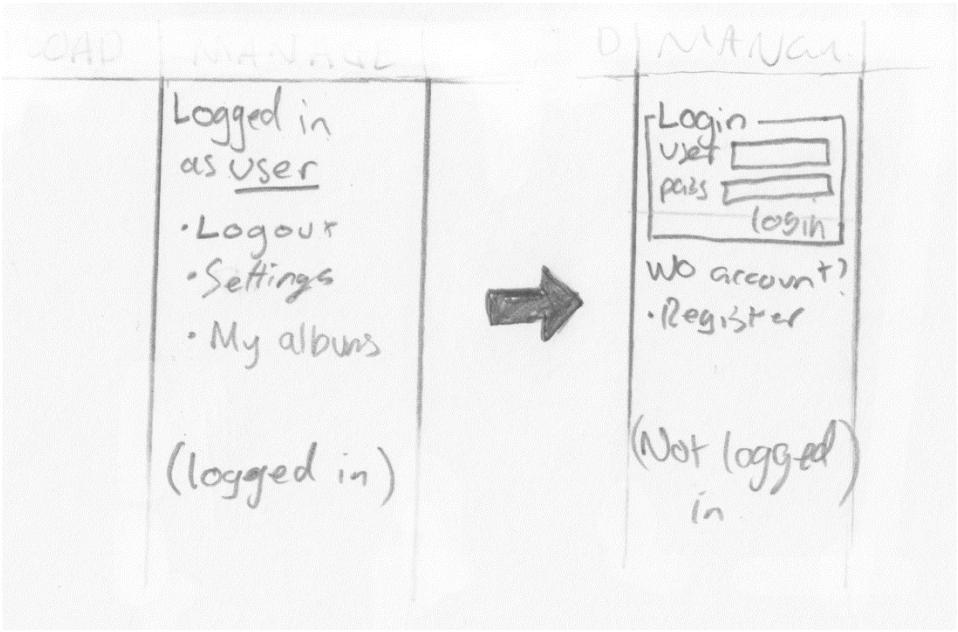
Simple design, a camera icon on the left with the FP characters inside which stand for 'Furze Platt', then the word photography on the right.

Navigation bar:



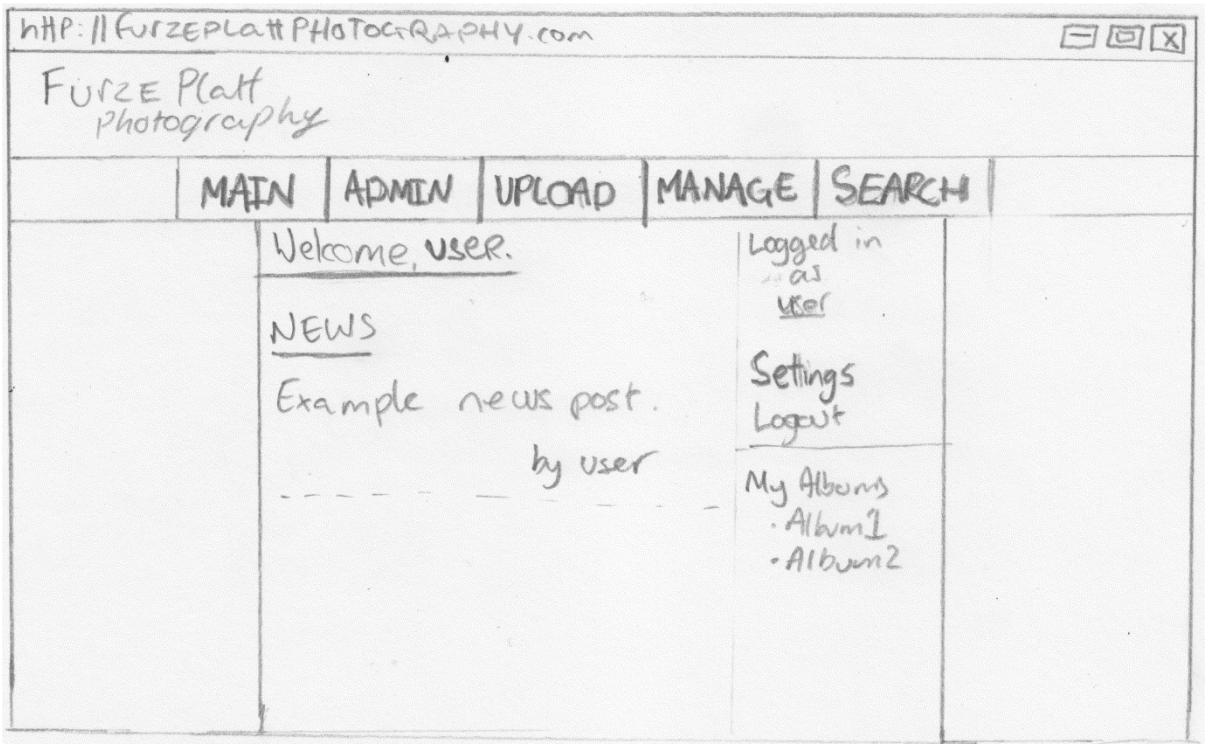
The navigation bar is built for ease of use, I thought that having a mouse over colour change would be beneficial to the user as it allows them to easily identify which tab they are hovering over. The mouse over colour change will just be a darker colour of the navigation bars native colour. Additionally, I have only included a small amount of tabs, to avoid any confusion for users when browsing. The main tab is first to fit it's identify as the main page, and then from there the tabs were randomly positioned apart from the search tab which I feel should be last after seeing/experiencing this in other websites.

Widget area:



Each page will have a widget area to the right of the content box which will be consistent throughout the site and be unique for each user. As seen in the design, when logged in it will show who the user is logged in as, a logout & settings button and also where the 'My albums' button is I plan to have a brief listing of their albums underneath. Moving to the right, if you are logged out the widget will display a login box to allow the user to login. It will also provide a link to the registration page in the case that the user has not registered to the system.

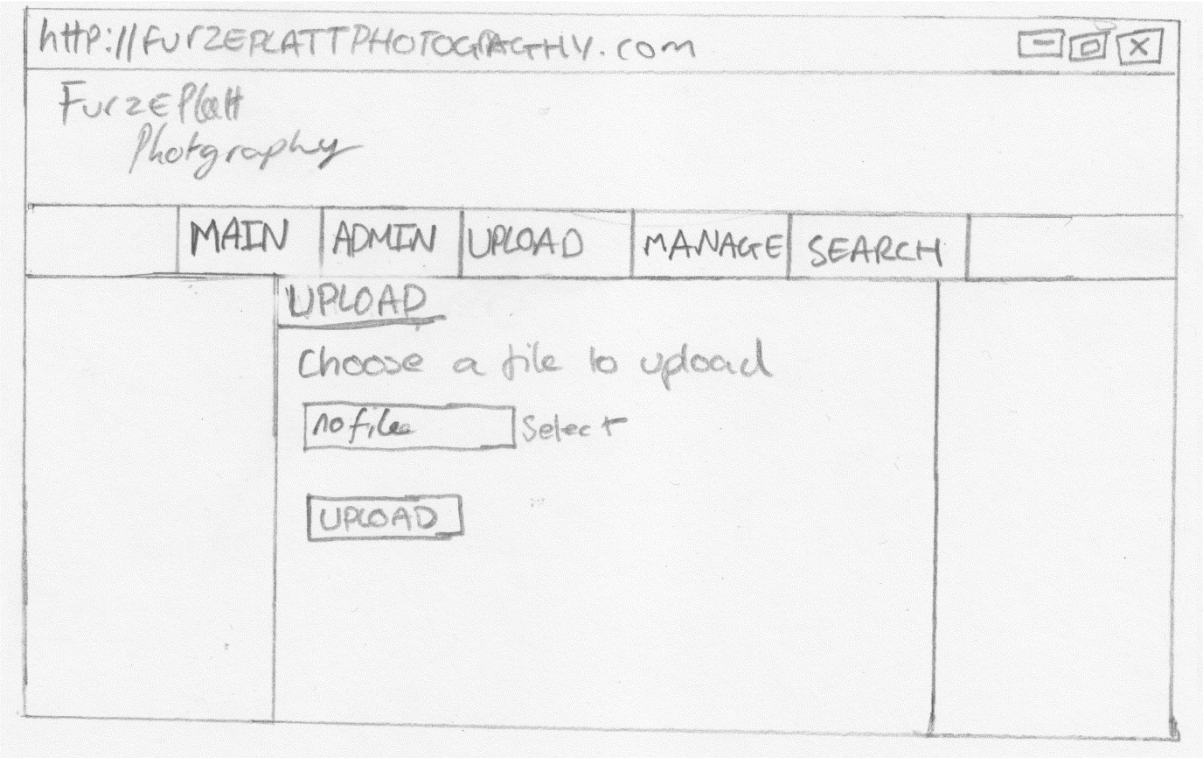
Main index page:



This would be the main page where users would go to after they have logged in, users are able to access all other aspects of the site from this page. It is also the design that all my other pages are based off of to keep consistency across the system.

Additionally, when users aren't logged in, where it has the "logged in as user" part, there will be a login form/widget where users can login from, as mentioned and designed previously where I broke the design down into its separate parts.

Photo upload page:



This will be the upload page where users go to upload images to the system. It uses the same template as the main index page.

Colour schemes:

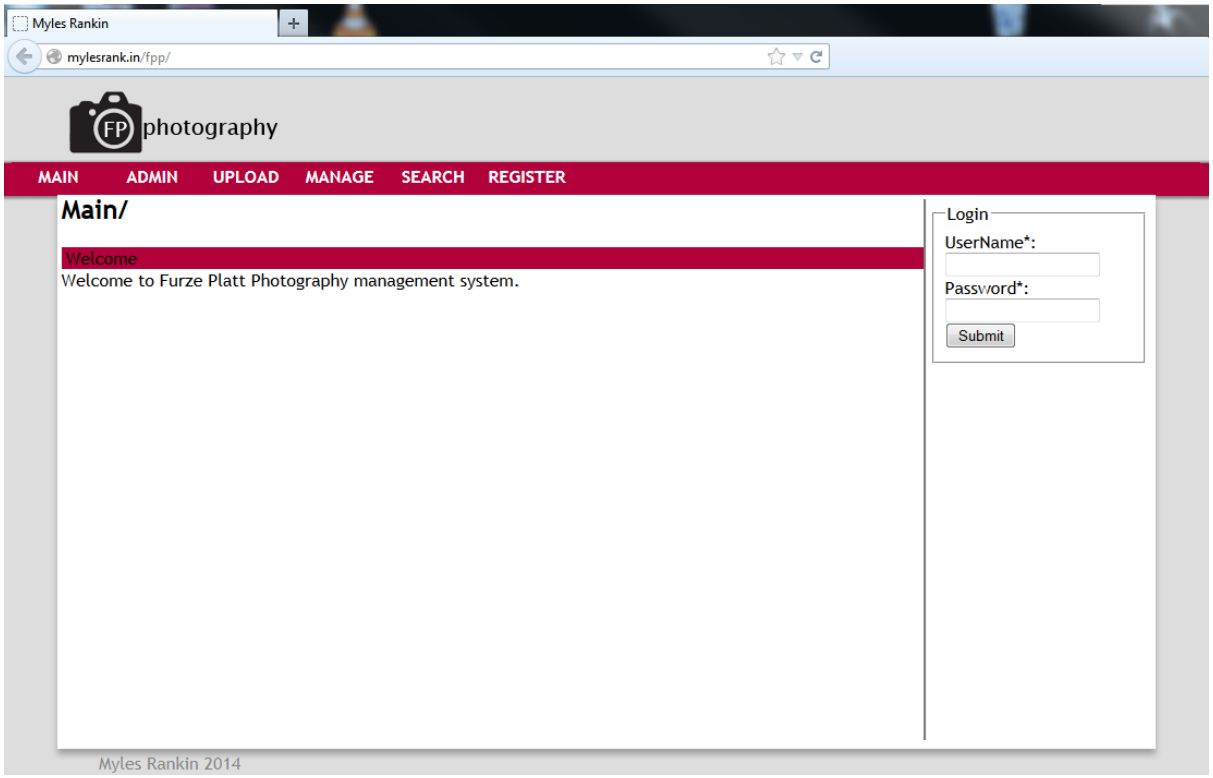
For the first pass of the design, I decided to use the following colour scheme:

Colours are defined as hex codes

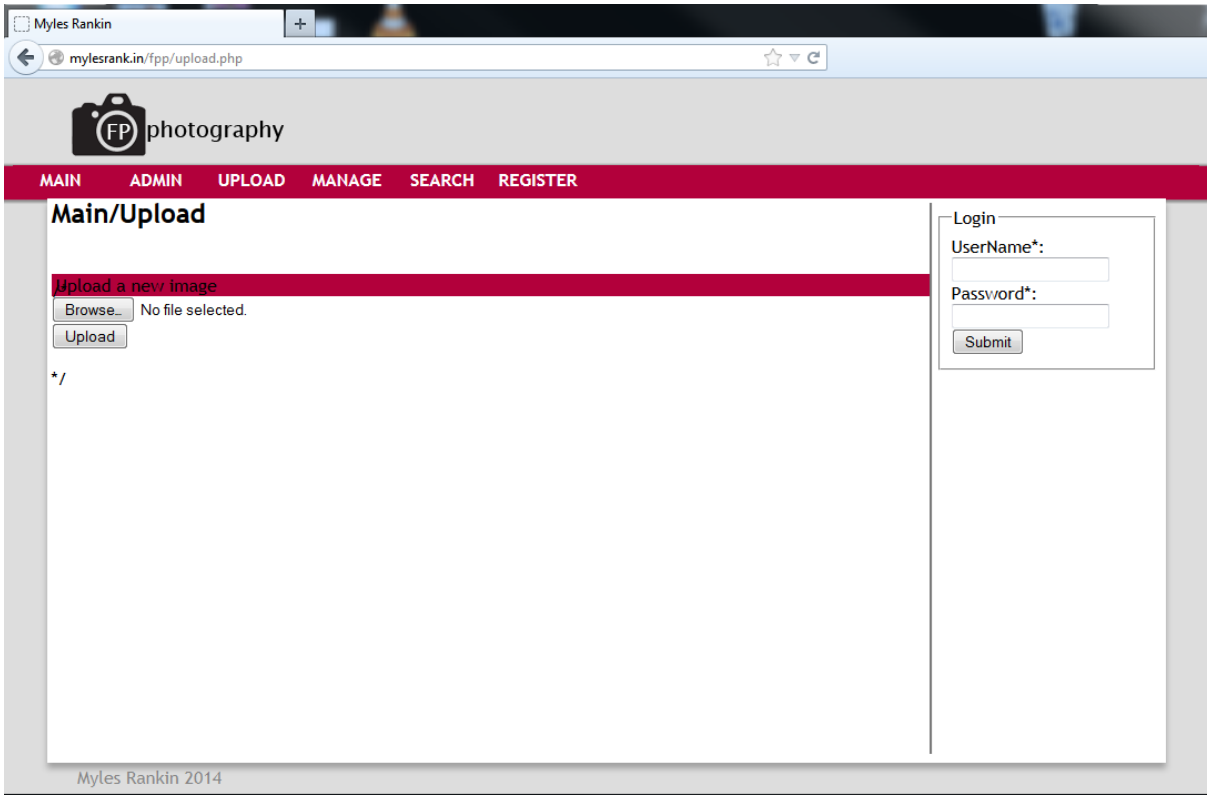
```
Background-colour: #dddddd (Light grey)
Navbar-colour: #b2003b (Dark pink)
Contentbox-colour: #FFFFFF (White)
Navbar-text: #FFFFFF (White)
Content-text: #000000 (Black)
```

First design screenshots:

Main index page:



Upload page:



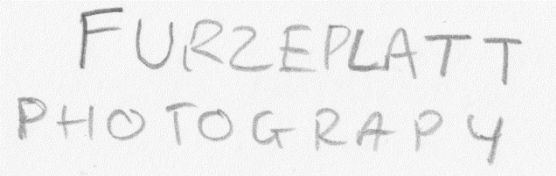
I felt that from the preliminary designs, the aside login widget on the right was inadequate and that I needed a standalone login page to separate unauthorised users from the main system. Therefore I

started a design on a new login system. I also showed my progress of the design to my user, he agreed with changing the login area but also gave some other comments and changes he would like to see.

Along with the login system, he wanted the colour scheme changed to more darker colours and the icon on the logo needed to go so I made a second pass a the sketching designs.

New logo design:

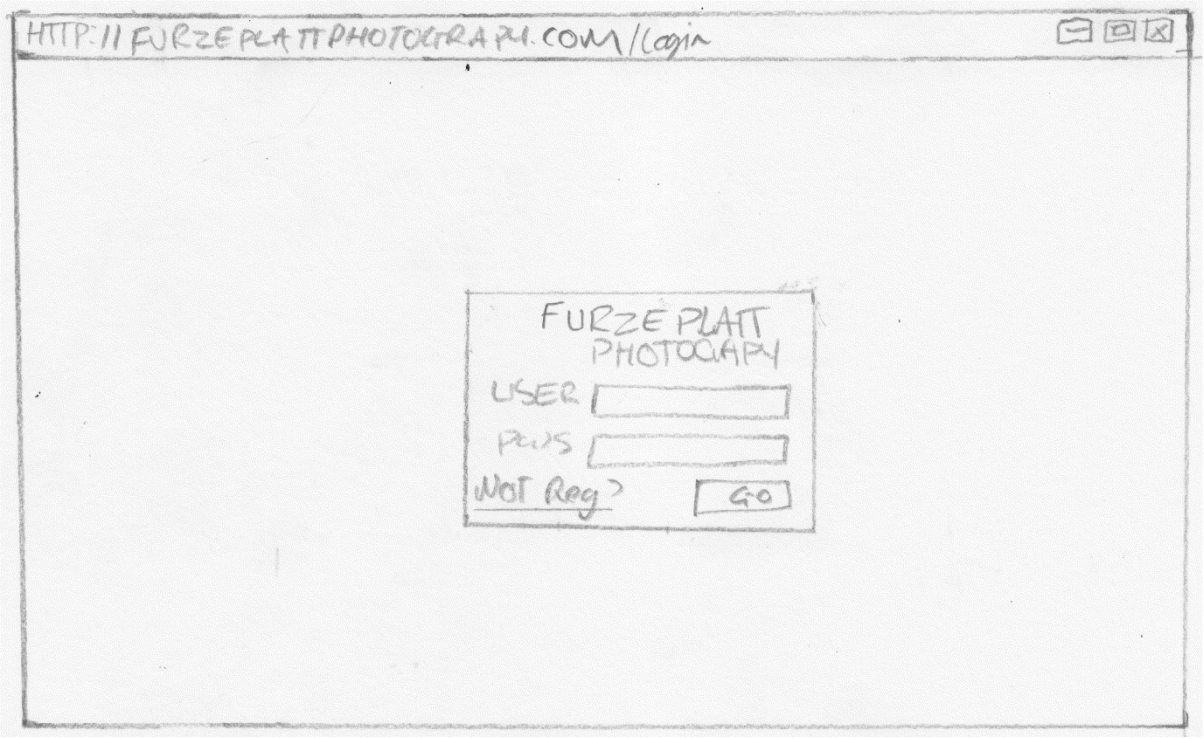
(sketch)



(photoshop result)

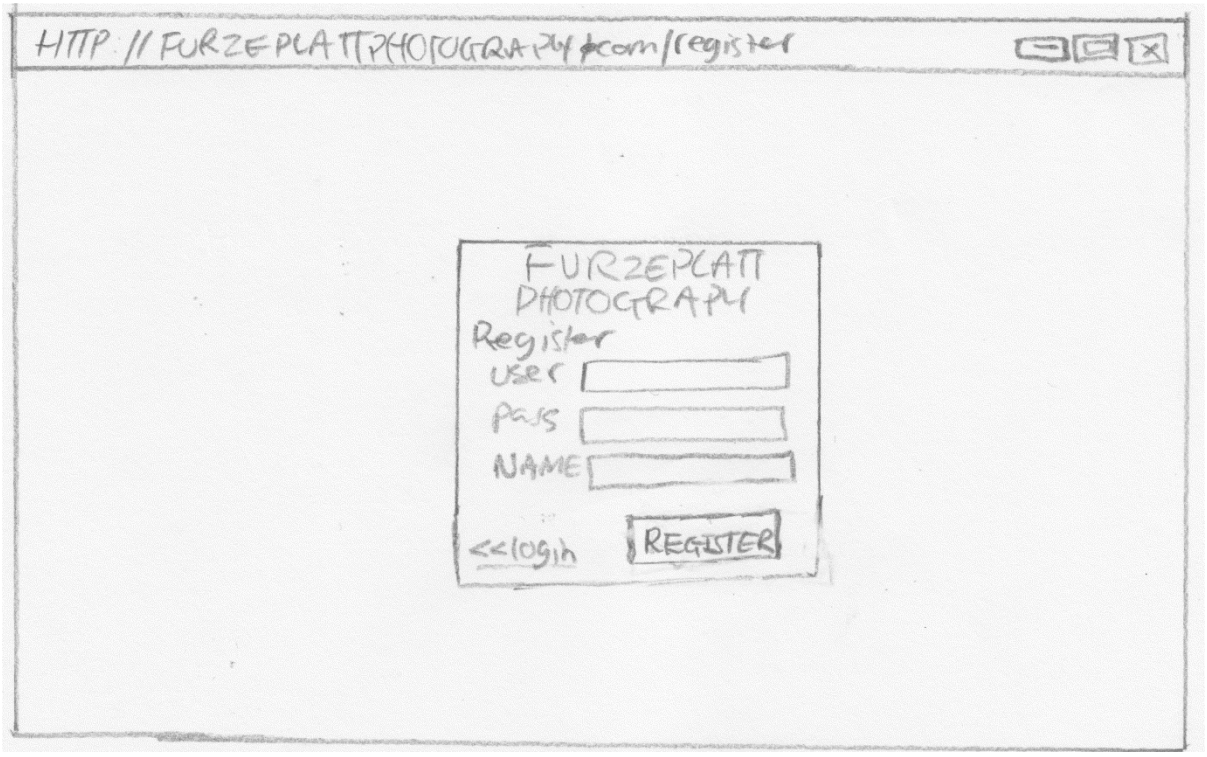
The logo rendered logo is shown on a black background as the text is in white, because the background on the new colour scheme is much darker so the white allows the logo to be clearly seen.

New login page design:



This is the new login system where users will first land before being able to enter the system, it will also be the place where users are directed to when they are not logged in. It now uses a different template from the main index as myself and my user wanted something that looked simplistic & separate where people first see the system.

Registration page design:



The registration page is where people will be able to navigate to from the login, to register an account to be able to login then access the system. It uses the same elements as the login page, to make a smooth transition between the two.

New colour scheme:

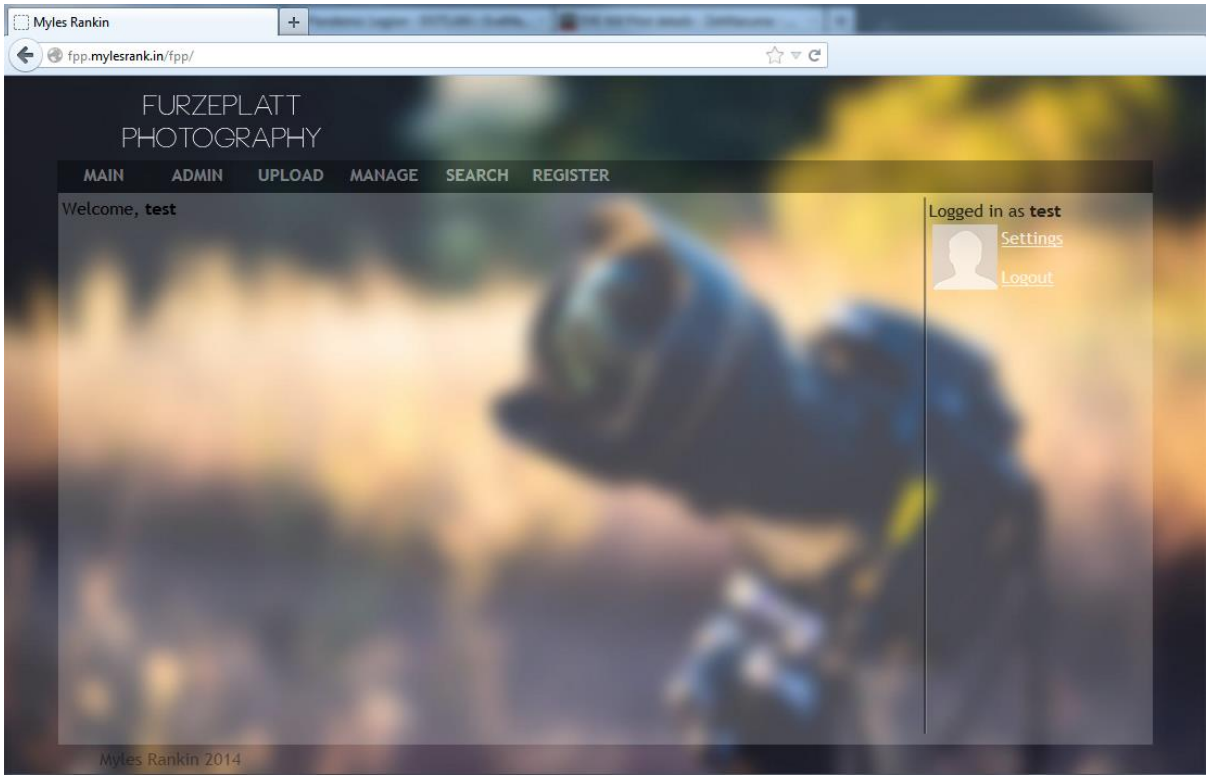
Colours are defined as hex codes

```
Background: blurred image of camera (See second pass screenshots)
Navbar-colour: #000000 (black) with opacity 0.8
Contentbox-colour: #FFFFFF (White)with opacity 0.2
Navbar-text: #FFFFFF (White)with opacity 0.8
Content-text: #000000 (Black)
```

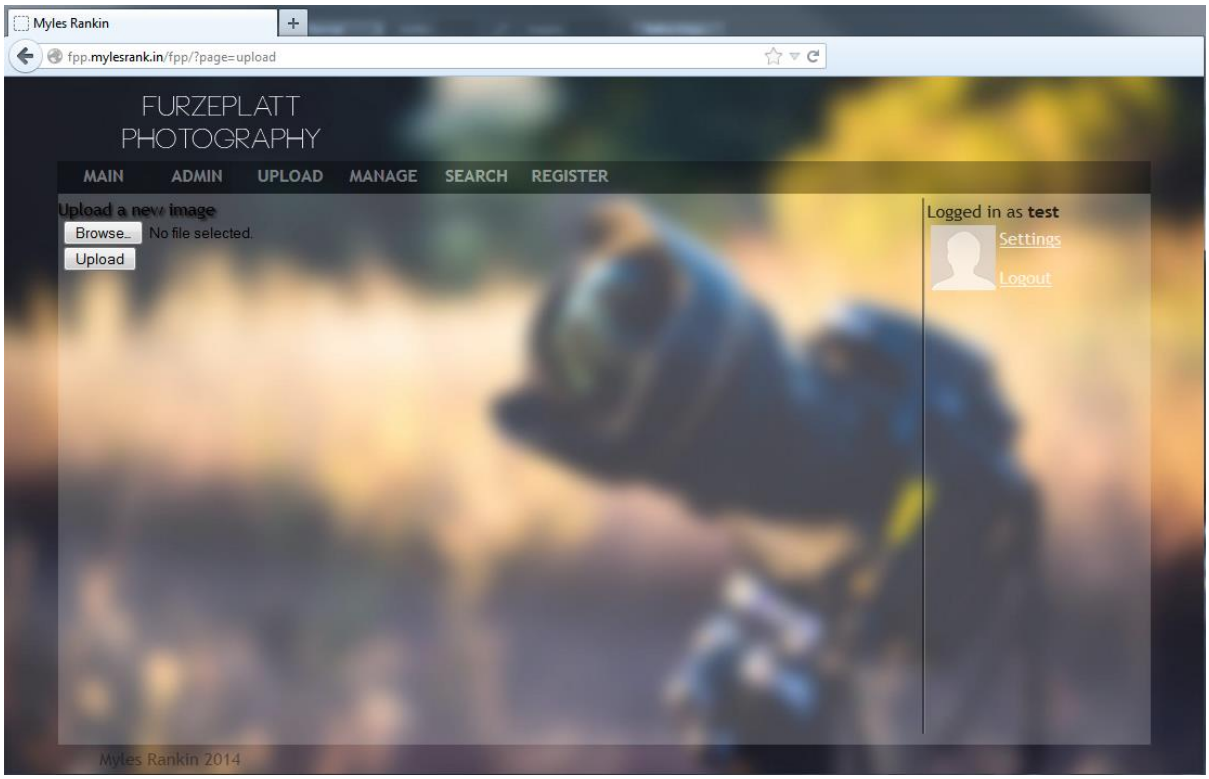
Second pass design:

On my second pass of the design, I decided to change a few things. I felt that the colour scheme could be improved along with the logo and that the navigation bar could be shortened.

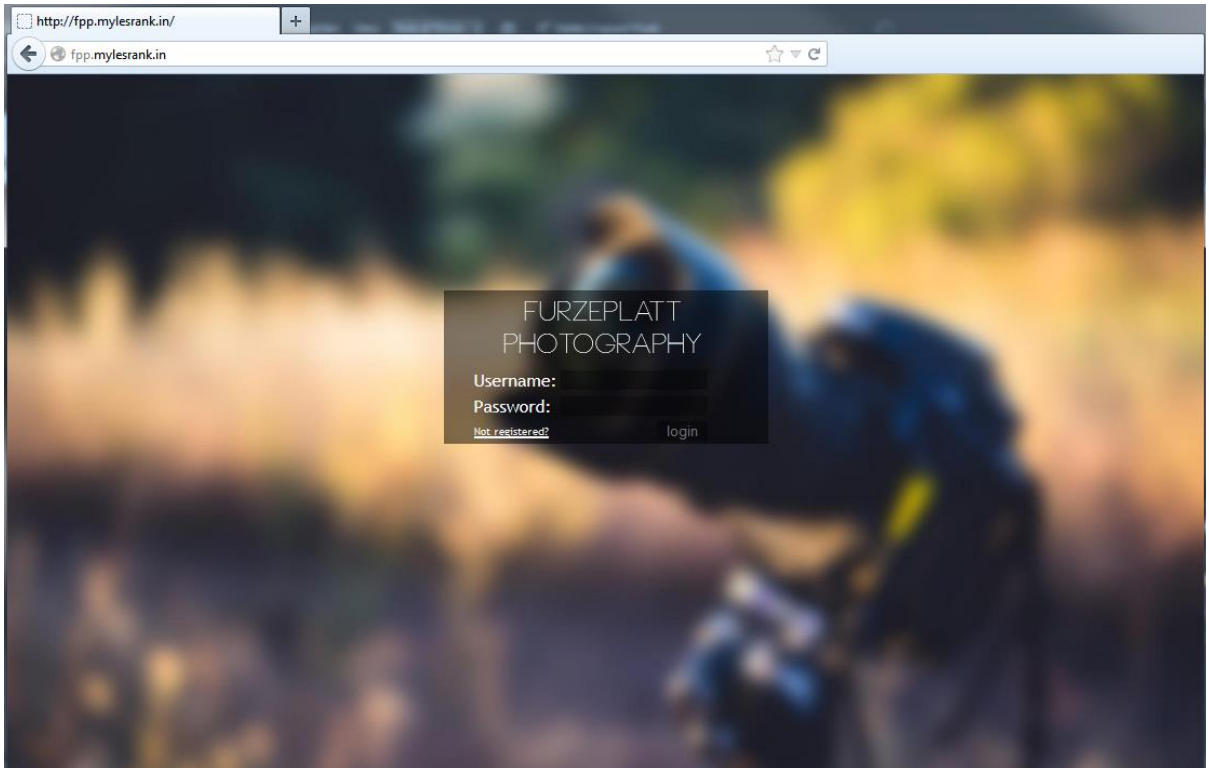
Main index page:



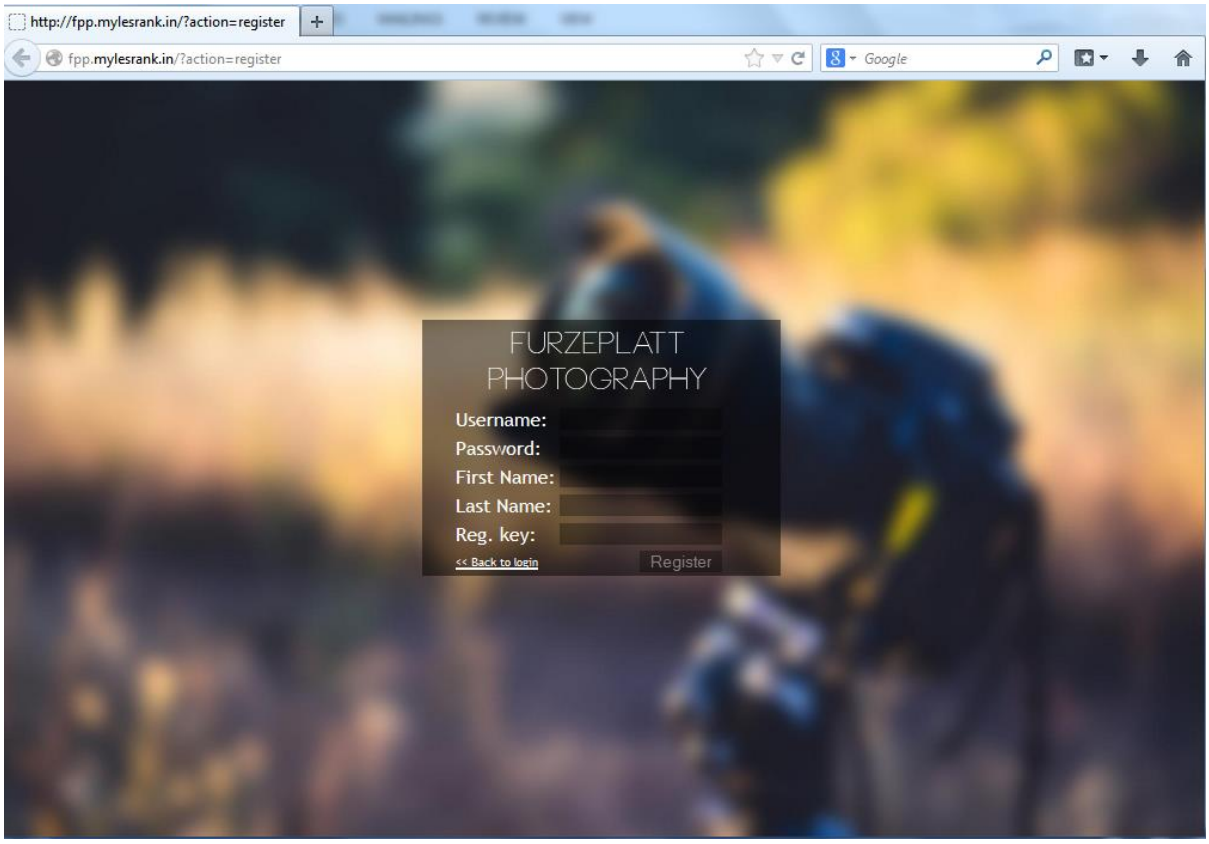
Upload page:



Login page:



Register page:



From the main index page, I can break down the elements of the page into an include system to easily be able to recreate all my other pages and to allow me to duplicate a theme perfectly throughout the

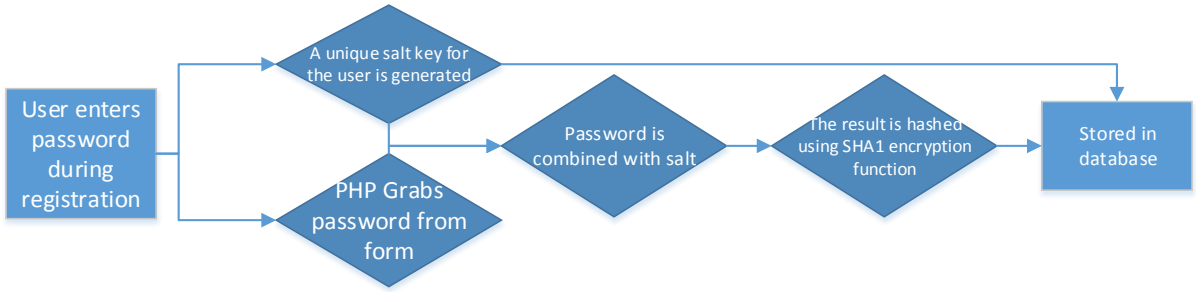
system. Due to this, I haven't seen the need to design all the pages before creating them just plan what the contents will be.

Security and Integrity of data

As mentioned in analysis and objectives for the system, it is required to be secure and the data maintained within must be kept restricted, as there will be pictures of students, etc. This branches from the login/registration that is a solution to keeping the pages secure and restricted. Each page will require a permission level to access and view, this is given when a user registers. This meets the requirement of restricting access to the system, as all users will be required to login to actually view or do anything.

The second problem that has been encountered was the need to safely store passwords securely. Obviously, as people will be registering there will be sensitive information such as passwords which will need to be stored somewhere to be used when a user logs in. I plan to use MySQL databases to store all the information used in the registration and login, but as they can possibly be compromised I don't want to store the passwords in plaintext.

Therefore, I have created a process to secure store passwords:



As seen in my flowchart, the password is never stored as plaintext but always as a hashed string. The user has a salt generated when registering using the openssl_random_pseudo_bytes() function (It generates a secure string of random characters/symbols), this is stored in the database and is assigned to the user for later use. The password that the user has entered is then stored in a variable, then combined with the salt. The result of this is then used using the sha1() encryption function built into php, this will then create a hash that is practically impossible to reverse to its plaintext form, therefore it's secure because even if the system was compromised you can't do anything with the hashed password. This hash is then stored on the server in the users password field. Now, you may be thinking how a login works if a user is using a plaintext password but their password is actually stored as a hash password. Well when logging in, the login system uses the same process to hash the password given for login then compares this hashed password to the stored hash, if they are the same then it allows access.

Algorithm Design

MySQL Connection module

Description
The module that allows for communication between the program and the database server, this will be seen included into pages throughout the program.
Pseudo-code
`VAR host = localhost`

```
VAR dbUser = root
VAR dbPass = password123
VAR dbName = fpp_main

VAR connection = MYSQLI_CONNECT(host,dbUser,dbPass,dbName)

If MYSQLI_CONNECT_ERRNO(connection)
   PRINT "Failed to connect to MySQL db:"
   PRINT MYSQLI_CONNECT_ERROR(connection)

END
```

Login system

Description

The system requires a login system to restrict access to the system, the login is required to check If in the form that is being entered If the username exists and If the password they have entered is correct.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE

VAR username = POST['username']
VAR password = POST['password']
VAR SQL usernamerequest =
SELECT UserName, Password, salt, id FROM members WHERE UserName =
"username"
VAR Object = QUERY->FETCH_OBJECT[MYSQL_QUERY(SQL)]
VAR dbUsername = Object->UserName
VAR dbPassword = Object->Password
VAR dbSalt     = Object->Salt
VAR UniqueID   = Object->UniqueID

VAR password_hashed = SHA1_ENCYRPT(password+Salt)

If Object IS SET = TRUE
   IF password_hashed == dbPassword
      SESSION_START
      VAR SESSION['USERNAME'] = dbUsername
      VAR SESSION['UNIQUEID'] = dbUniqueID
      PRINT "Login success"
      END(REDIRECT_SECUREINDEX)
   ELSE
      PRINT "Incorrect password"
      END(REFRESH_PAGE)
ELSE
   PRINT "Username invalid"
   END(REFRESH_PAGE)
```

Register system

Description

The registration script is required to register new users to the system. It is required to check for existing usernames, and also store information gathered from html form securely via hashing.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE

VAR postUsername  = POST['USERNAME']
VAR postFirstname = POST['PASSWORD']
```

```
VAR postLastname  = POST['LASTNAME']
VAR postRegkey    = POST['REGKEY']
VAR postPassword  = POST['PASSWORD']

VAR salt = OPENSSL_RANDOM_PSEUDO_BYTES(32)
VAR password = SHA1(postPassword+salt)


IF EMPTY(postUsername) = TRUE
   PRINT "UserName field required"
   END
IF EMPTY(postPassword) = TRUE
   PRINT "Password field required"
   END
IF EMPTY(postFirstname) = TRUE
   PRINT "First name field required"
   END
IF EMPTY(postLastname) = TRUE
   PRINT "Last name field required"
   END

VAR userCheckQuery = MYSQL_QUERY(connection,SELECT UserName FROM
members WHERE UserName = 'postUsername')

VAR userCheck = userCheckQuery->FETCH_OBJECT

IF userCheck IS SET = TRUE
   PRINT "Username taken"
   END

VAR regkeyCheckQuery = MYSQL_QUERY(connection, SELECT PermissionLevel
FROM regkeys WHERE RegKey = 'postRegkey')

VAR regkeyCheck = regkeyCheckQuery->FETCH_OBJECT
VAR permissionLevel = regkeyCheck->PermissionLevel

IF regkeyCheck IS SET = TRUE
   MYSQL_QUERY(connection,UPDATE members SET PermissionLevel =
   'permissionLevel' WHERE UserName = 'postUsername')
ELSE
   PRINT "No valid registration key was found, you have been
registered without permissions"

VAR sqlQuery1_insertdata =
INSERT INTO members (UserName, Password, Firstname, Lastname, salt,
regdate) VALUES (postUsername, password, postFirstname, postLastname,
salt)

IF MYSQL_QUERY(connection,sqlQuery1) RETURNS ERROR
   PRINT "ERROR: " + MYSQL_ERROR(connection)
   END
ELSE
   PRINT "Success, you have been registered"
```

Page session required module

Description

This module adds protection to pages only allowing those who have a verified session to be able to access it.

Psuedo-code

```
SESSION_START

VAR pageAccess = FALSE
VAR SessionID = SESSION['uid']

IF SessionID IS SET
    pageAccess = TRUE
ELSE
    pageAccess = FALSE
    PRINT "You are not logged in, you will now be redirected to login
page"
    END(REDIRECT_INDEX)
```

Upload system

Description

The upload system is where users will go to upload images to the system which will then be bound to their account.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE
INCLUDE PAGE_SESSION_REQUIRED MODULE

VAR uid = SESSION['uid']

VAR target = CurrentDir + "/Uploads/"
ARRAY whitelist = ARRAY('jpg','jpeg','png','gif')
FETCH FILES_UPLOAD from FORM[FILES]

IF FILES_UPLOAD = TRUE
   ARRAY file_ary = arrayFiles(FILES_UPLOAD)
   Foreach (File_ary as VAR FILE)
          VAR FileOriginal = File['name']
          VAR FileExtension = PathInfo(Extension) of File['name']
          VAR FileName = PathInfo(FileName) of File['name']
          IF FileExtension IS IN_ARRAY whitelist = TRUE
             END("One or move files have unallowed extensions")
             REDIRECT_UPLOAD
          VAR i = 1
          While File_exists (target + FileOriginal)
                FileOriginal = FileName + "_" + i + FileExtension
                VAR i = INCREASE BY 1
      IF UploadedFile_moved(FILE['name'] TO Target + FileOriginal) =
      TRUE
          VAR databasePath = CurrentDir + FileOriginal
          QUERY (STORE databasePath, FileName, uid IN database)
          PRINT FileOriginal + " has been uploaded."
      ELSE
          PRINT "Error during upload, retry."
END
```

Manage system

Description

The manage system is for managing photos uploaded by the user, i.e. deleting photos and viewing links for them.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE
INCLUDE PAGE_SESSION_REQUIRED MODULE

VAR sessionUID = SESSION['uid']

VAR sql = "SELECT UploadID, FileName, Path FROM uploads WHERE uid =
sessionUID"

VAR query = MYSQL_QUERY(connection,sql)

ARRAY row = empty

WHILE row = MYSQL_FETCH_ARRAY(query)
    VAR id = row['UploadID']
    PRINT TABLE
           Row['UploadID']
           Row['FileName']
           Row['Path']
 PRINT "<a href='delete.php?id=$id' class='confirmation'>delete</a>"
 PRINT URL: url + Row['FileName']


SCRIPT javascript
     //Confirmation of deletion
    VAR elems = GetElementsByClassName('confirmation')
    VAR confirmIt = function(e)
Function(e){
    IF confirm = TRUE {
         PRINT 'ARE YOU SURE YOU WANT TO DELETE IMAGE'
         e.PreventDefault()
         }
    FOR VAR i = 0, l = elems.length: i < 1: i++ {
         Elems[i].addEventListener('click, confirmIt, False')
         }
```

Gallery system

Description

The gallery system will display images from each users uploaded images in a grid format.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE

SQL UserData = "SELECT * FROM users"
VAR UserDataQuery = SQL_QUERY(UserData)

VAR UserID = GET['ID'] FROM URL

IF UserID IS SET
   SQL UserGalleryData = "SELECT * FROM users WHERE id=UserID
   VAR UserGalleryDataQuery = SQL_QUERY(UserGalleryData)
   ARRAY UserArr = FETCH UserGalleryDataQuery AS ARRAY
   SQL FindPaths = "SELECT PathResized PathFullSize FROM Uploads
WHERE UploaderID =UserID
    VAR Paths = SQL_QUERY(FindPaths)
    VAR ThumbNailArr = ARRAY[]
```

```
    VAR FullImageArr = ARRAY[]

    WHILE row1 = FETCH_ROW(FindPaths)
          INSERT INTO ARRAY(ThumbNailArr, row1[ThumbNail][0]

    WHILE row2 = FETCH_ROW(FindPaths)
          INSERT INTO ARRAY(FullImageArr, row2[FullImage][0]

   VAR PicturesPerRow = 10
   VAR NumRowsFromQuery = COUNT_NUM_ROWS(UserGalleryDataQuery)

TABLE OPENED

   FOR a=0,a<NumRowsFromQuery, a++
       PRINT ROW START
           FOR b=0,b<PicturesPerRow,b++
               PRINT TABLE DATA START
                     DISPLAY ThumbNailArr[(a*PicturesPerRow)+b]
                     ON CLICK ON THUMBNAIL
                        DISPLAY FullImageArr[(a*PicturesPerRow)+b]
               PRINT TABLE DATA END
           END FOR
     PRINT ROW END
  FOR END

TABLE CLOSED
```

Identification of Storage Media

As the whole system is web based, the system will need to be stored on a server remote from the client. The client will simply just be accessing the system remotely from a browser. The photos being uploaded will accumulate a lot of space, so will require sufficient storage area on a server to be hosted.

The only viable storage media to have my system on is a server in one of the Furze Platt Senior School server rooms. This means it will be easily accessible by my users in mind, enabling them to add more storage onto it when needed as the system will be quite dynamic in the sense where uploads are stored, a system administrator will be able to simply add a bigger drive to the server to increase space.

Additionally, having the server within the school premises creates two upsides. The first is speed, as it will be on the school internal network users will have the max bandwidth speed the schools network can handle, which is 100Mbit/s in most rooms. At this speed, the average file will be uploaded in 0.2 seconds, which is more than adequately fast. The second upside, is to do with security. If the system were to be hosted outside the school, there may be problems with data protection of storing photos of pupils outside the school network, but inside the school network where the server will be hosted means that the IT Technicians will have direct control over the server and will be able to ensure files are kept securely.

Systems Testing

To make sure the system is able to correctly perform and process from all user inputs (both correct and incorrect), and also correctly allow the user to navigate across the whole system I planned several ways to test my system.

Testing plan

User input and Output Testing plan

This table is a plan for testing each input and output given to the user on the system. I have used TEX which is data type being used, Typical, Erroneous and Extreme.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Exepected Outcome	Actual Outcome	Comments
1	Logging in	T: Logging in with correct details	Redirection to main index of system, with displayed message “Login successful. Welcome, <username>.”		
		E: Logging in with incorrect details	No redirection and display of message “Username invalid.”		
		E: Logging in with some login details	No redirection and display of message(s): “Password missing.” “Username missing.”		
		X: Logging in with no details	No redirection and display of message “Username missing.”		
2	Registering	T: Registering with all correct fields (including valid regkey)	Registration of user and message displayed “Success: You have been registered”.		

		E: Registering with some incorrect/missing required fields	No registration and display of message(s): “Username field empty”. “Password field empty” “First Name field empty”. “Last Name field empty”.		
		E: Registering with correct fields but with invalid RegKey	Registration successful and display of message: “No valid RegKey was found during registration, you have been registered without permissions.” And “Success: You have been registered"”		
		E: Registering with correct fields but with a username that already exists.	No registration and display of message “Username taken”.		
		X: Registering with all empty fields	No registration and display of message “Username field empty”.		
3	Uploading files	T: Upload allowed/normal file type	File will upload and display of message “File <filename> has been uploaded.”		
		E: Upload non allowed file type	File will fail to upload and display of message “One or more files had unallowed extensions during upload, please retry”		
		X: Upload multiple non allowed file types	All files will fail to upload and display of message “One or more files had unallowed extensions during upload, please retry”		
4	Removing a file	T: Delete one of own images	File successfully deleted and message displayed “<filename> has been successfully deleted.”		
		X: Navigating delete id to delete someone else’s image	No file deleted and message displayed “That is not your file to delete!”		
5	Searching for file(s)	T: Search for existing file(s)	Successful search and table displayed with FileNames and URL’s beside them		
		X: Search for non-existent file(s)	Unsuccessful search with empty table displayed (No rows of filenames, just		

			table headings.)		
6	View events table	T: Navigate to events page	Full table of events displayed with name, dates, notes and event ETA		
7	View a user gallery	T: Navigate to an existing user gallery	Full display of that users photos in a grid		
		E: Navigate to a non-existent user gallery by user ID	Empty page with no images displayed		
8	Create a new post	T: All fields filled correctly	Message “Post successfully created!” displayed then redirection to admin page. New post visible in manage posts table.		
		E: Some empty fields	No post created and message(s) “"Post title is missing." Or "Post content is missing." displayed.		
		X: All fields empty	No post created and message “"Post title is missing." displayed.		
9	Create a new event	T: All fields filled correctly	Event created, redirect to admin panel with new event seen in manage events table		
		E: Some required fields empty	No event created, no redirection and display of either message: "Event name is missing." Or "Event date is missing."		
		X: All fields empty	No event created, no redirection and display of message: "Event name is missing."		
		X: Non date input text filled into Date field	Unable to input letters into date field.		
10	Create a new registration key	T: All fields filled correctly	Registration key created, redirect to admin page and message displayed: "Success!"		
		T: Already existing Regkey name entered in key name field	No Registration key created, no redirect and message displayed: "That key name is already taken, please enter a new one!"		
		E: Some required fields empty	No Registration key created, no redirected and message(s) displayed: "Permission Level is missing." Or "Keyname is		

			missing." Or "Expiry date is missing."		
		X: All fields empty	No Registration key created, no redirect and message displayed: "Permission Level is missing."		
		X: Letters attempted input into Permission Level input	No Registration key created, no redirect and message displayed: "Please enter a number."		
		X: Non date input text filled into Date field	Unable to input letters into date field.		
11	Delete user	T: Press user delete button (admin page)	User deleted from database/table on admin page and message "User succesfully deleted." displayed.		
12	Delete registration key	T: Press Regkey delete button (admin page)	Regkey deleted from database/table on admin page and message "Regkey succesfully deleted." displayed.		
13	Delete post	T: Press Post delete button (admin page)	Post deleted from database/table on admin page and message "Post succesfully deleted." displayed.		
14	Delete event	T: Press Event delete button (admin page)	Event deleted from database/table on admin page and message "Event succesfully deleted." displayed.		
15	Change user permission level	T: Fill out new permission number field correctly with number	User permission level changed, redirect back to admin page and message displaying “User permission level successfully changed!"		
		E: Leave permission number field empty	No user permission level changed, no redirect and message displaying "Permission field is empty"		
		X: Input letters into new permission number field	No user permission level changed, no redirect and message displayed: "Please enter a number.”		
16	Add tags to image	T: Tags field filled out	Tags added to image, viewable in table.		
		E: Tags field empty	No tags added and message displayed "Tags		

			field is empty."		

Testing results

User input and Output Testing plan

This table is the results for testing each input and output given to the user on the system. Results have been screen captured, referenced in each test (In table) and are located at the end of this document.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments/ Corrective actions
1	Logging in	T: Logging in with correct details	Redirection to main index of system, with displayed message “Login successful. Welcome, <username>.”	As expected (Screenshot Test – 1)	None required.
		E: Logging in with incorrect details	No redirection and display of message “Username invalid.”	As expected (Screenshot Test – 2)	None required.
		E: Logging in with some login details	No redirection and display of message(s): “Password missing.” “Username missing.”	As expected (Screenshot Test – 3a, 3b)	None required.
		X: Logging in with no details	No redirection and display of message “Username missing.”	As expected (Screenshot Test – 4)	None required.
2	Registering	T: Registering with all correct fields (including valid regkey)	Registration of user and message displayed “Success: You have been registered”.	As expected (Screenshot Test – 5)	None required.
		E: Registering with some incorrect/missing required fields	No registration and display of message(s): “Username field empty”. “Password field empty” “First Name field empty”. “Last Name field empty”.	As expected (Screenshot Test – 6a, 6b, 6c, 6d)	None required.
		E: Registering with correct fields but with invalid RegKey	Registration successful and display of message: “No valid RegKey was found during registration, you have been registered without permissions.” And	As expected (Screenshot Test – 7)	None required.

			"Success: You have been registered""		
		E: Registering with correct fields but with a username that already exists.	No registration and display of message "Username taken".	As expected (Screenshot Test – 8)	None required.
		X: Registering with all empty fields	No registration and display of message "Username field empty".	As expected (Screenshot Test – 9)	None required.
3	Uploading files	T: Upload allowed/normal file type	File will upload and display of message "File <filename> has been uploaded."	As expected (Screenshot Test – 10)	None required.
		E: Upload non allowed file type	File will fail to upload and display of message "One or more files had unallowed extensions during upload, please retry"	As expected (Screenshot Test – 11)	None required.
		X: Upload multiple non allowed file types	All files will fail to upload and display of message "One or more files had unallowed extensions during upload, please retry"	As expected (Screenshot Test – 12)	None required.
4	Removing a file	T: Delete one of own images	File successfully deleted and message displayed "<filename> has been successfully deleted."	As expected (Screenshot Test – 13)	None required.
		X: Navigating delete id to delete someone else's image	No file deleted and message displayed "That is not your file to delete!"	As expected (Screenshot Test – 14)	None required.
5	Searching for file(s)	T: Search for existing file(s)	Successful search and table displayed with FileNames and URL's beside them	As expected (Screenshot Test – 15)	None required.
		X: Search for non-existent file(s)	Unsuccessful search with empty table displayed (No rows of filenames, just table headings.)	As expected (Screenshot Test – 16)	None required.
6	View events table	T: Navigate to events page	Full table of events displayed with name, dates, notes and event ETA	As expected (Screenshot Test – 17)	None required.
7	View a user gallery	T: Navigate to an existing user gallery	Full display of that users photos in a grid	As expected (Screenshot Test – 18)	None required.
		E: Navigate to a non-existent user gallery by user ID	Empty page with no images displayed	As expected (Screenshot Test – 19)	
8	Create a	T: All fields filled	Message "Post	As expected	None

	new post	correctly	successfully created!" displayed then redirection to admin page. New post visible in manage posts table.	(Screenshot Test – 20a, 20b)	required.
		E: Some empty fields	No post created and message(s) ""Post title is missing." Or "Post content is missing." displayed.	As expected (Screenshot Test – 21a, 21b)	None required.
		X: All fields empty	No post created and message ""Post title is missing." displayed.	As expected (Screenshot Test – 22)	None required.
9	Create a new event	T: All fields filled correctly	Event created, redirect to admin panel with new event seen in manage events table	As expected (Screenshot Test – 23a, 23b)	None required.
		E: Some required fields empty	No event created, no redirection and display of either message: "Event name is missing." Or "Event date is missing."	As expected (Screenshot Test – 24a, 24b)	None required.
		X: All fields empty	No event created, no redirection and display of message: "Event name is missing."	As expected (Screenshot Test – 25)	None required.
		X: Non date input text filled into Date field	Unable to input letters into date field.	As expected (Screenshot Test – 26)	None required.
10	Create a new registration key	T: All fields filled correctly	Registration key created, redirect to admin page and message displayed: "Success!"	As expected (Screenshot Test – 27)	None required.
		T: Already existing Regkey name entered in key name field	No Registration key created, no redirect and message displayed: "That key name is already taken, please enter a new one!"	As expected (Screenshot Test – 28)	None required.
		E: Some required fields empty	No Registration key created, no redirected and message(s) displayed: "Permission Level is missing." Or "Keyname is missing." Or "Expiry date is missing."	As expected (Screenshot Test – 29a, 29b, 29c)	None required.
		X: All fields empty	No Registration key created, no redirect and message displayed: "Permission Level is missing."	As expected (Screenshot Test – 30)	None required.
		X: Letters attempted input into Permission Level input	No Registration key created, no redirect and message displayed: "Please enter a number."	As expected (Screenshot Test – 31)	None required.

		X: Non date input text filled into Date field	Unable to input letters into date field.	As expected (Screenshot Test – 32)	None required.
11	Delete user	T: Press user delete button (admin page)	User deleted from database/table on admin page and message "User successfully deleted." displayed.	As expected (Screenshot Test – 33)	None required.
12	Delete registration key	T: Press Regkey delete button (admin page)	Regkey deleted from database/table on admin page and message "Regkey successfully deleted." displayed.	As expected (Screenshot Test – 34)	None required.
13	Delete post	T: Press Post delete button (admin page)	Post deleted from database/table on admin page and message "Post successfully deleted." displayed.	As expected (Screenshot Test – 35)	None required.
14	Delete event	T: Press Event delete button (admin page)	Event deleted from database/table on admin page and message "Event successfully deleted." displayed.	As expected (Screenshot Test – 36)	None required.
15	Change user permission level	T: Fill out new permission number field correctly with number	User permission level changed, redirect back to admin page and message displaying “User permission level successfully changed!"	As expected (Screenshot Test – 37)	None required.
		E: Leave permission number field empty	No user permission level changed, no redirect and message displaying "Permission field is empty"	As expected (Screenshot Test – 38)	None required.
		X: Input letters into new permission number field	No user permission level changed, no redirect and message displayed: "Please enter a number.”	As expected (Screenshot Test – 39)	None required.
16	Add tags to image	T: Tags field filled out	Tags added to image, viewable in table.	As expected (Screenshot Test – 40)	None required.
		E: Tags field empty	No tags added and message displayed "Tags field is empty."	As expected (Screenshot Test – 41)	

System Maintenance

Introduction

The Photography Management system for Furze Platt Photography has been designed to be a lightweight, responsive and powerful tool for the photography club to use frequently. It's been made to have maintenance functions built in accessible to the user client side without needing access to any of the server side files.

There are two distinguished types of user, an Administrator and a plain user. They are based off of a permission level given to them through registration to the system. A user can upload files to the system, view them, add tags, access paths to share uploads and also delete them. They have only view access to the news posts and events system. An Admin is completely different with a higher permission level, they are able to delete and create both posts and events that the users view.

The system backend is powered by PHP, which is a server side programming language. All client side output is handled by HTML, CSS and JavaScript. As you might be able to tell from the languages used, the system is web based, which makes it very flexible and accessible from a multitude of computers and devices. It also means that it is compatible with any computer/device that can run a browser, so there is no actual requirement for specification or operating system of the computer, only the need of a browser and internet connection.

Installation

As mentioned previously, this system is a web based solution, in the nature of this there is no installation required by the user apart from a browser that most operating systems already have installed. However, there is installation required by the system administrator to the server.

These installation steps presume the system administrator has a web server, FTP server and MySQL database already installed, as these steps are not covered.

The system will come packaged in a ZIP file will all server side php files included. All you have to do is simply upload these files to the web server via FTP to a location of your choosing. Once installed, you must open the includes/mysqlconnection.php file and fill out your MySQL sever connection information there (an example configuration will already be there, this can be seen Appendix 1 code listings). Once this has been done, there is only one more step which is creating the MySQL databases.

This part assumes you have adequate knowledge on how to submit SQL commands to your database to create tables.

To create the tables needed in your database submit the following creation queries to your MySQL server:

```
--
-- Table structure for table `events`
--

CREATE TABLE IF NOT EXISTS `events` (
  `eventID` int(64) NOT NULL AUTO_INCREMENT,
  `eventName` varchar(1024) NOT NULL,
  `eventDate` date NOT NULL,
  `eventFinished` int(1) NOT NULL DEFAULT '0',
  `eventCreatorID` int(64) NOT NULL,
  `eventNote` varchar(1024) NOT NULL,
  PRIMARY KEY (`eventID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=71 ;

--
-- Table structure for table `filesizer`
--

CREATE TABLE IF NOT EXISTS `filesizer` (
  `id` int(64) NOT NULL,
  `UserName` varchar(2056) NOT NULL,
  `fileUsed` int(255) NOT NULL,
  `unqiueID` int(64) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`unqiueID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=10 ;


--
-- Table structure for table `members`
--

CREATE TABLE IF NOT EXISTS `members` (
  `id` int(64) NOT NULL AUTO_INCREMENT,
  `UserName` varchar(256) NOT NULL,
  `Password` varchar(256) NOT NULL,
  `FirstName` varchar(256) NOT NULL,
  `LastName` varchar(256) NOT NULL,
  `PermissionLevel` int(64) NOT NULL DEFAULT '0',
  `salt` varchar(32) NOT NULL,
  `regdate` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=216 ;

--
-- Table structure for table `posts`
--

CREATE TABLE IF NOT EXISTS `posts` (
  `postID` int(64) NOT NULL AUTO_INCREMENT,
  `postTitle` varchar(1024) NOT NULL,
  `postContent` varchar(5000) NOT NULL,
  `postAuthorID` int(64) NOT NULL,
```

```
  `postDate` datetime NOT NULL,
  PRIMARY KEY (`postID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=18 ;

-- --------------------------------------------------------

--
-- Table structure for table `regkeys`
--

CREATE TABLE IF NOT EXISTS `regkeys` (
  `CreatorID` int(64) NOT NULL,
  `RegKey` varchar(32) NOT NULL,
  `PermissionLevel` int(64) NOT NULL,
  `IsUsed` tinyint(1) NOT NULL DEFAULT '0',
  `ExpiryDate` datetime NOT NULL,
  `id` int(64) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=15 ;


--
-- Table structure for table `uploads`
--

CREATE TABLE IF NOT EXISTS `uploads` (
  `UploadID` int(64) NOT NULL AUTO_INCREMENT,
  `FileName` varchar(1024) NOT NULL,
  `Path` varchar(2048) NOT NULL,
  `uid` int(64) NOT NULL,
  `PathResized` varchar(2048) NOT NULL,
  `tags` varchar(2048) NOT NULL,
  PRIMARY KEY (`UploadID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=224 ;
```

Once all tables have been created in the database, and correct MySQL configuration details have been put into the mysqlconnection.php file, then the system will be fully operational.

Final SQL Table structure

Here are final tables of each table in the database, it shows data type, default values, column names and any indexes/primary keys in each database. Column names have changed since design, due to easier naming schemes and consistency across all tables.

‘events’ table

Column	Type	Null	Default
eventID	int(64)	No	
eventName	varchar(1024)	No	
eventDate	date	No	
eventFinished	int(1)	No	0
eventCreatorID	int(64)	No	
eventNote	varchar(1024)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	eventID	2	A	No

‘filesizer’ table

Column	Type	Null	Default
id	int(64)	No	
UserName	varchar(2056)	No	
fileUsed	int(255)	No	
unqiueID	int(64)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	unqiueID	9	A	No

‘members’ table

Column	Type	Null	Default
id	int(64)	No	
UserName	varchar(256)	No	
Password	varchar(256)	No	
FirstName	varchar(256)	No	
LastName	varchar(256)	No	
PermissionLevel	int(64)	No	0
salt	varchar(32)	No	
regdate	date	Yes	*NULL*

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	9	A	No

‘posts’ table

Column	Type	Null	Default
postID	int(64)	No	
postTitle	varchar(1024)	No	
postContent	varchar(5000)	No	
postAuthorID	int(64)	No	
postDate	datetime	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	postID	0	A	No

‘regkeys’ table

Column	Type	Null	Default
CreatorID	int(64)	No	
RegKey	varchar(32)	No	
PermissionLevel	int(64)	No	
IsUsed	tinyint(1)	No	0
ExpiryDate	datetime	No	
id	int(64)	No	

Indexes

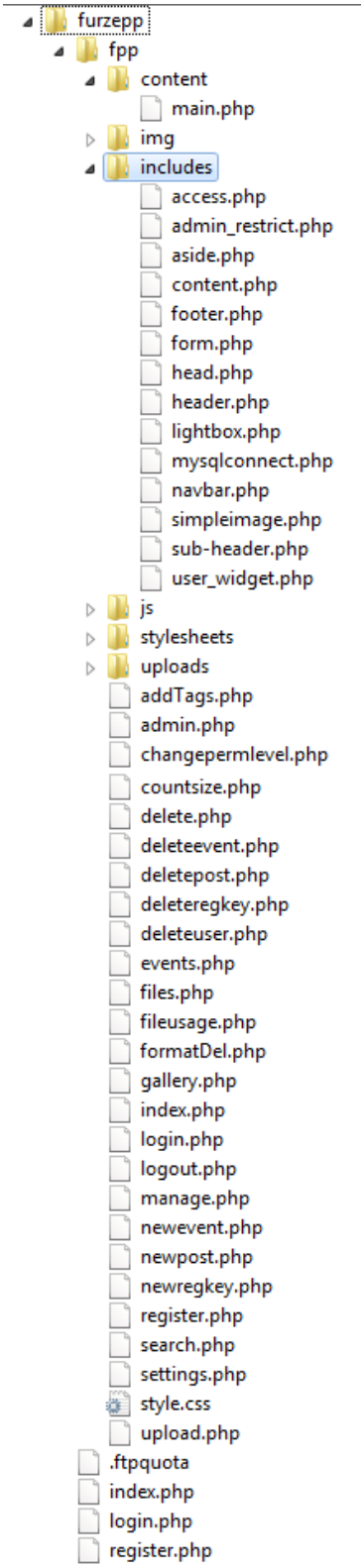
Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	5	A	No

‘uploads’ table

Column	Type	Null	Default
UploadID	int(64)	No	
FileName	varchar(1024)	No	
Path	varchar(2048)	No	
uid	int(64)	No	
PathResized	varchar(2048)	No	
tags	varchar(2048)	No	

Indexes

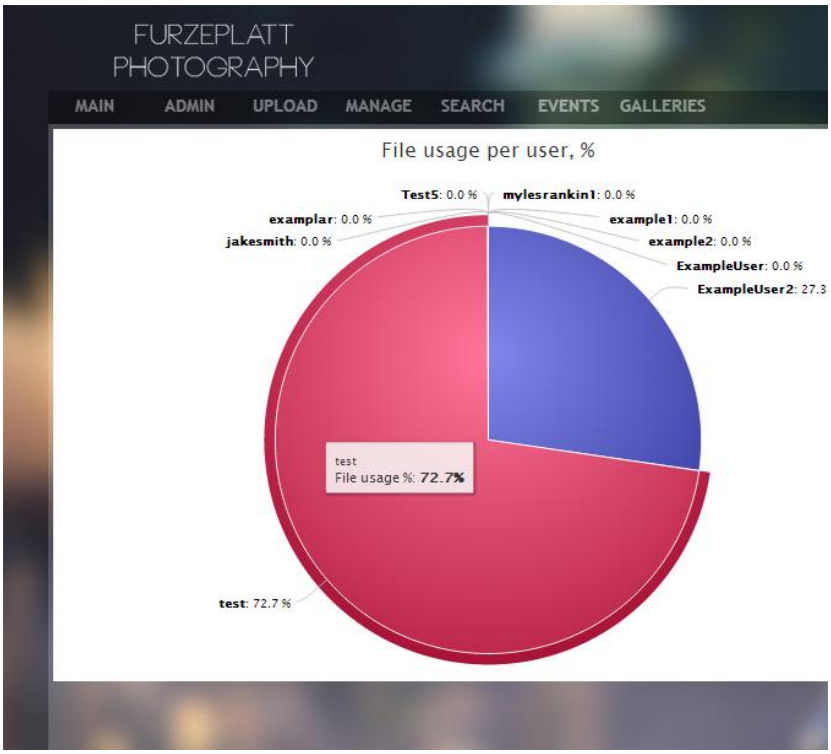
Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	UploadID	35	A	No



Checking file usage

For system maintenance, a system administrator will want to view the amount of file space being used. It is assumed a system administrator knows how to find out the size of a folder through their given server operating system, because this varies between each operating system, but what If a system administrator wants to work out which user is taking up what space

For this purpose I designed a stats page to show a graph of all users on a pie chart. It shows each member as a slice with a % number of the amount of the total space they are using. This can be found by first updating the 'filesizer' table by navigating to countsize.php file on your website. Once executed it will display a counted number in bytes for each user and will be queried to the database. After this has been done, you can navigate to the fileusage.php file and you will see a pie chart with data which will look like the screenshot shown below:



As seen each user is displayed, even ones who aren't using any space up (0%). From this administrator can find those users who are using the most space, and prune their files/warn them if necessary.

**This graph is generated using HighCharts library*

File tree

As Seen in the file tree there are many files, and a few sub folders that contain different files. Each file in both the root directory /furzepp and /furzepp/fpp are constructed around an include system. This means

the visual elements of the page have been split up into individual include files as seen in the includes folder /furzepp/fpp/includes. So when making a new page with the default template you simply just have to open the header include and then end at the bottom of the page with the footer include.

Example:

```
<?php
include 'includes/header.php';

// Content/alorgithms to output data would go here //

include 'includes/footer.php';
?>
```

Doing this will fully render the template of the page. The system has been designed like this for many reasons. Firstly, it saves on code as this is able to be easily replicated and put into new pages instead of having to copy and paste out the same style template into each page you simply include a file. Secondly, by doing this the styling of pages will be consistent across the system as all styling pulls from these two master files. And lastly, if you wish to make any changes to the templates across the site, you only have to edit these two files and the changes will be applied to every page.

Additionally I have added other include files that perform actions that are required to happened multiple times across different pages. For example, mysqlconnect.php contains all the MySQL connection details and stores them into a $con variable that is used in any query in the system, it also initializes connection with the database and prints any connection errors If they occur.

I have also put any libraries into include files, such as the simple image resizing library for resizing images for thumbnails (as seen in includes/simple-image.php). Once this file is included in the page I can simply call the functions from the include file to manipulate images.

Navigation Overview

Main navigation

Here is the main navigation overview, it shows how all the pages/php files are linked together on a primary navigational level.

Form overview

In the system there 8 input forms for users to submit data. They are located in the following files:

- /login.php
- /register.php
- /fpp/upload.php
- /fpp/settings.php
- /fpp/newregkey.php
- /fpp/newpost.php
- /fpp/newevent.php

- /fpp/changepermlevel.php

All data retrieved from the client is stored via HTTP POST method, when is it needed by the system PHP calls it via $_POST.

/Login.php

The form located in the login page handles and stores input for user logons. All data from this form is retrieved by /fpp/login.php.

Variables gathered from form:

Name	Description
$username	Grabs username field from the form via HTTP POST Method
$password	Grabs password field from the form via HTTP POST Method

/Register.php

The form located in the registration page handles and stores input for user registrations. All data from this form is retrieved by /fpp/register.php.

Variables gathered from form:

Name	Description
$pusername	Grabs username field from the form via HTTP POST Method
$pfirstname	Grabs firstname field from the form via HTTP POST Method
$plastname	Grabs lastname field from the form via HTTP POST Method
$pregkey	Grabs registration key field from the form via HTTP POST Method
$ppassword	Grabs password field from the form via HTTP POST Method

/fpp/Upload.php

The form located in the upload page handles and temporally stores files uploaded from user uploads. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$files['upload']	Grabs upload data from HTTP POST FILES Method

/fpp/Settings.php

The form located in the settings page handles and stores input for user password changes. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$newpassword	Grabs new password field from the form via HTTP POST Method

/fpp/newregkey.php

The form located in the 'create new registration' key page handles and stores input for new registration keys. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$permissionlevel	Grabs permission level field from the form via HTTP POST Method
$keyname	Grabs new key name field from the form via HTTP POST Method
$expirydate	Grabs expiry date field from the form via HTTP POST Method

/fpp/newpost.php

The form located in the 'create new post page' handles and stores input for new posts. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$postTitle	Grabs post title field from the form via HTTP POST Method
$postContent	Grabs post content field from the form via HTTP POST Method

/fpp/newevent.php

The form located in the 'create new event page' handles and stores input for new events. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$eventName	Grabs event name field from the form via HTTP POST Method
$eventDate	Grabs event date field from the form via HTTP POST Method
$eventNote	Grabs event note field from the form via HTTP POST Method

/fpp/changepermlevel.php

The form located in the 'change user permission level page' handles and stores input for changing user permissions levels. All data from this form is retrieved by itself.

Variables gathered from form:

Name	Description
$newperm	Grabs new permission level field from the form via HTTP POST Method

File and Variable Overview

In the table below describes what each php file in the system does

File	Description
Access.php	Restricts a non-registered user from accessing page it's included on
Admin_restrict.php	Restricts registered users that don't have the required permission level from accessing the page
Aside.php	Renders aside widget bar
Footer.php	Renders footer and closes div tags for page
Head.php	Has html head styling inside, included in header.php
Header.php	Renders header of each page, opens and includes all styling for system ui
Lightbox.php	Includes preview library for image viewing
Mysqlconnect.php	Has all MySQL connection/configuration details in and initiates MySQL connection. Is included in any page that has MySQL queries.
Navbar.php	Renders Navigation bar ui for user navigation across the site, is included in header.php
Simpleimage.php	Includes library functions for resizing images
Sub-header.php	Contains html/css <style> and <meta> data includes. Is included in header.php
User_widget.php	User widget on side of each page that welcomes <username> and has a users navigation bar to personalised pages such as settings and my albums.
addTags.php	Enables user to add tags to an image
Admin.php	All administrative features are done on this page. (manage: users, events, posts and regkeys)
Changepermlevel.php	Enables admin to change a users permission level
Countsize.php	Counts all users file usage in bytes and submits to 'filesizer' table in database.
Delete.php	Enables user to delete own images
Deleteevent.php	Enables admin to delete an event
Deletepost.php	Enables admin to delete a post
Deleteregkey.php	Enables admin to delete a regkey
Deleteuser.php	Enables admin to delete a user
Events.php	Displays all events from 'events' table in database
Files.php	Displays a users file in a list

Fileusage.php	Displays all users file usage in % in a pie chart
formatDel.php	Exports users file usage and name in a format highcharts library can use
Gallery.php	Displays grid of each users images in a gallery with clickable images which brings up a preview
Index.php	Main index page, shows posts from admins and manages page requests from HTTP GET method for pages. (i.e. ?page=admin redirects to admin.php)
Login.php	Enables user to login to the system
Logout.php	Enables user to logout of the system
Manage.php	Enables user to manage photos such as delete and preview
Newevent.php	Enables admin to create an event
Newpost.php	Enables admin to create a new post
Newregkey.php	Enables admin to create a new registration key
Register.php	Enables an unregistered user to register to the system
Search.php	Enables a user to search for existing and tagged files
Settings.php	Enables a user to change password details
Style.css	Style sheet for all elements in UI
Upload.php	Enables user to upload files
Main.php	Content of main page included in if statement inside index.php

Variable listing

Below is a table of variables from all the code contained within the system.

Variable Name	Contained within	Description
$_username	Login.php	Stores username from form via POST method
$action	Index.php	Stores GET variable from action via GET method
$con	mysqlconnect.php	Defines full connection parameters for SQL querying
$currentdate	Newpost.php, register.php	Used to store current date today using date() function
$dataArray	Fileusage.php	Used to store mysql query data as a normal array
$dbname	Mysqlconnect.php	Defines MySQL database name for use in query db selection (configuration variable)
$dbpassword	Login.php	Used to store password from stored password in members database
$eventDate	Newevent.php	Used to store the event date from FORM via HTTP POST Method
$eventName	Newevent.php	Used to store the event name from FORM via HTTP POST

		Method
$eventNote	Newevent.php	Used to store the event note from FORM via HTTP POST Method
$expiryDate	Newregkey.php	Used to store the regkey expiry date from FORM via HTTP POST Method
$fetch, $fetch2, …	Delete.php, settings.php	Used to store a fetched associative array from a query
$getID	Gallery.php	Used to store ID from page url via HTTP GET (i.e. url?get=1)
$host	Mysqlconnect.php	Defines hostname of MySQL server (configuration variable)
$id	Changepermlevel.php, delete.php, deleteevent.php, deletepost.php, deleteregkey.php, deleteuser.php	Used to store a user's ID from a query
$imageID	addTags.php	Used to store an image ID from a query
$keyName	Newregkey.php	Used to store the regkey keyname from FORM via HTTP POST Method
$newpassword	Settings.php	Used to store the new password plaintext from FORM via HTTP POST Method
$newpasswordhashed	Settings.php	Used to store the result of hasing $newpassword with the system hashing method
$newpasswordQuery	Settings.php	Stores sql query for inserting new password into users member row
$newPerm	Changepermlevel.php	Used to store the new permission level from FORM via HTTP POST Method
$now	Events.php	Used to store current time using time() function
$numrows	Newregkey.php	Used to store the result of counting number of rows from a query
$obj	Login.php	Used to store a query as an object
$pageaccess	Access.php	Defines whether the page is accessible or not (TRUE/FALSE)
$pagename	Index.php	Used to store grabbed page name request from url via HTTP GET Method (i.e. url?page=example)
$pass	Mysqlconnect.php	Defines password used in MySQL connection to server
$password	Login.php, register.php	Used to store password from FORM via HTTP POST Method

$passwordhash	Login.php	Stores result of hashing $password variable retrieved from FORM
$path	Delete.php	Used to define full server path for use in deleting or unlinking a file using unlink() function
$permArr	Admin_restrict.php	Used to store result of querying a user permission level
$permissionLevel, $permLevel	Newregkey.php, register.php, admin_restrict.php	
$pf	Delete.php	Fetches file path from associate array
$pfirstname	Register.php	Used to store the first name from register FORM via HTTP POST Method
$plastname	Register.php	Used to store the first name from register FORM via HTTP POST Method
$postContent	Newpost.php	Used to store the new post content from FORM via HTTP POST Method
$postTitle	Newpost.php	Used to store the new post title from FORM via HTTP POST Method
$ppassword	Register.php	Used to store the password from register FORM via HTTP POST Method
$pregkey	Register.php	Used to store the regkey from register FORM via HTTP POST Method
$pusername	Register.php	Used to store the username from register FORM via HTTP POST Method
$query, $query2, $query3, …	formatDel.php, gallery.php, admin_restrict.php, login.php, manage.php, newregkey.php, search.php, addTags.php, admin.php, main.php, countsize.php, events.php, files.php, fileusage.php	Used to store result of a query
$regkeycheck	Register.php	Stores result of regkey check query
$regkeycheck_obj	Register.php	Stores the fetched object of the $regkeycheck query result using fetch_object() function
$restrictionlist	Index.php	Defines pages as an array that will have page restriction module included
$salt	Settings.php, login.php, register.php	Used to store a user salt value
$searchinput	Search.php	Used to store the search input from search FORM via HTTP

		POST Method
$sql, $sql2, $sql3,…	deletepost.php, register.php, manage.php, addTags.php, changepermlevel.php, main.php, countsize.php, files.php, deleteuser.php, gallery.php, deleteevent.php, deleteregkey.php, events.php, delete.php, newevent.php, newpost.php, newregkey.php, search.php, admin.php	Used to store SQL query command to be used in a query execution
$tags	addTags.php	Used to store the tags from add tags FORM via HTTP POST Method
$target	Upload.php	Defines upload target directory for uploading a file to
$uid	settings.php, addTags.php, upload.php, manage.php, newevent.php, admin_restrict.php, login.php, deletepost.php, deleteuser.php, changepermlevel.php, files.php, newregkey.php, admin.php, newpost.php, deleteevent.php, deleteregkey.php, delete.php	Used to store the current users id via $_SESSION['uid'] stored in their session.
$uploaderID	Delete.php	Used to store uploaderID when querying data for an upload
$user	Mysqlconnect.php	Defines username to use in MySQL connection
$usercheck	Register.php	Stores result of usercheck query
$usercheck_obj	Register.php	Used to store the object result of the usercheck query result using fetch_object() function
$username	Login.php, formatDel.php, fileusage.php	Used to store a user's username either from a form via HTTP POST Method, or from a MySQL query
$whitelist	Upload.php, index.php	Defines all pages in an array that a user is allowed to access when using $_GET
$_SESSION	All pages	Contains all current user's session data
$_GET	All pages	Used to get a value from a url via GET Method (i.e. url?get=value)
$_POST	All form pages	Used to grab form data via HTTP POST Methods

Detailed Algorithm Design

There are no comments included in the tables below, to find the commented code please refer to Appendix 1 at the end of this document. Use the location: box for the filename of each detailed algorithm code and refer to the Appendix 1 contents page.

Algorithm: Upload files	**Location:** /fpp/upload.php

Description

This algorithm in the upload files page handles all file uploads from the user. It first grabs all files from HTTP POST FILE method array and stores them into a new indexed array using an arrayFiles function. I originally thought there would be a function to do this, so in my pseudo I have just called the function arrayFiles, it turns out it doesn't exist so I had to get my own as seen in the code listing at the bottom of this table. Once in an array, each entry in array is looped in a foreach statement. Each time it loops though the image is checked for already existing, If it does exist then it is renamed with _ and a number corresponding with the number of duplicates existing using an incremental variable. The statement also stores all file info for each file looped into the 'uploads' table in the database.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE
INCLUDE PAGE_SESSION_REQUIRED MODULE

VAR uid = SESSION['uid']

VAR target = CurrentDir + "/Uploads/"
ARRAY whitelist = ARRAY('jpg','jpeg','png','gif')
FETCH FILES_UPLOAD from FORM[FILES]

IF FILES_UPLOAD = TRUE
   ARRAY file_ary = arrayFiles(FILES_UPLOAD)
   Foreach (File_ary as VAR FILE)
          VAR FileOriginal = File['name']
          VAR FileExtension = PathInfo(Extension) of File['name']
          VAR FileName = PathInfo(FileName) of File['name']
          IF FileExtension IS IN_ARRAY whitelist = TRUE
             END("One or move files have unallowed extensions")
             REDIRECT_UPLOAD
          VAR i = 1
          While File_exists (target + FileOriginal)
                FileOriginal = FileName + "_" + i + FileExtension
                VAR i = INCREASE BY 1
      IF UploadedFile_moved(FILE['name'] TO Target + FileOriginal) =
      TRUE
          VAR databasePath = CurrentDir + FileOriginal
          QUERY (STORE databasePath, FileName, uid IN database)
          PRINT FileOriginal + " has been uploaded."
      ELSE
          PRINT "Error during upload, retry."
END
```

Code

```
include 'includes/mysqlconnect.php';

function reArrayFiles(&$file_post) {
    $file_ary = array();
    $file_count = count($file_post['name']);
    $file_keys = array_keys($file_post);

    for ($i=0; $i<$file_count; $i++) {
        foreach ($file_keys as $key) {
```

```
            $file_ary[$i][$key] = $file_post[$key][$i];
        }
    }

    return $file_ary;
}

$uid = $_SESSION['uid'];
$target = getcwd() . "/uploads/";
$whitelist = array('jpg', 'jpeg', 'png', 'gif');

if ($_FILES['upload']) {
    $file_ary = reArrayFiles($_FILES['upload']);
    foreach ($file_ary as $file) {
            $fileOriginal = $file['name'];
            $fileExtension = pathinfo($file['name'], PATHINFO_EXTENSION);
            $FileName = pathinfo($file['name'], PATHINFO_FILENAME);
            if(in_array(strtolower($fileExtension),$whitelist) === FALSE){
                  echo '<meta http-equiv="refresh" content="1.5;
url=http://fpp.mylesrank.in/fpp/index.php?page=upload" />';
                  die("One or more files had unallowed extensions during upload, please
retry.");
                  }
            $i = 1;
            while(file_exists($target . $fileOriginal)){
                  $fileOriginal = $FileName . "_" . $i . "." . $fileExtension;
                  $i++;
                  }
            if(move_uploaded_file($file['tmp_name'],$target . $fileOriginal)){
                  $dbpath = "/fpp/uploads/" . $fileOriginal;
                  $resizeImage = new SimpleImage();
                  $resizeImage->load('uploads/' . $fileOriginal);
                  $resizeImage->resize(150,150);
                  $resizeImage->save(getcwd() . "/uploads/resized/" . $fileOriginal);
                  $resizedImagePath = "/fpp/uploads/resized/" . $fileOriginal;
                  $sql = "INSERT INTO uploads (FileName, Path, uid, PathResized) VALUES
('$fileOriginal','$dbpath','$uid','$resizedImagePath')";
                  mysqli_query($con,$sql);
                  echo "File " . $fileOriginal . " has been uploaded.<br>";
                  }else{
                        echo "Something went wrong when uploading, please retry!";
                        }
    }
}
```

Algorithm: Rendering grid of images for gallery	**Location:** /fpp/gallery.php
Description	
The algorithm in the gallery page is used to generate a grid of all the images that are owned by a specific user defined in the url via HTTP GET Method. The algorithm generates a table based on the number of images found in an SQL query to the 'uploads' table in the database. It selects all of a specific user images, counts the number of rows from the results which is the number of images. It uses this number to then loop through two for statements to define the number of rows the html table needs to display the images. The first for statement does this, then the second inputs 5 images per row, also making the images preview able by using Lightbox library.	
Pseudo-code	
INCLUDE MYSQL_CONNECTION MODULE SQL UserData = "SELECT * FROM users"	

```
VAR UserDataQuery = SQL_QUERY(UserData)

VAR UserID = GET['ID'] FROM URL

IF UserID IS SET
   SQL UserGalleryData = "SELECT * FROM users WHERE id=UserID
   VAR UserGalleryDataQuery = SQL_QUERY(UserGalleryData)
   ARRAY UserArr = FETCH UserGalleryDataQuery AS ARRAY
   SQL FindPaths = "SELECT PathResized PathFullSize FROM Uploads              WHERE
UploaderID =UserID
    VAR Paths = SQL_QUERY(FindPaths)
    VAR ThumbNailArr = ARRAY[]
    VAR FullImageArr = ARRAY[]

    WHILE row1 = FETCH_ROW(FindPaths)
          INSERT INTO ARRAY(ThumbNailArr, row1[ThumbNail][0]

    WHILE row2 = FETCH_ROW(FindPaths)
          INSERT INTO ARRAY(FullImageArr, row2[FullImage][0]

   VAR PicturesPerRow = 10
   VAR NumRowsFromQuery = COUNT_NUM_ROWS(UserGalleryDataQuery)

TABLE OPENED

   FOR a=0,a<NumRowsFromQuery, a++
       PRINT ROW START
           FOR b=0,b<PicturesPerRow,b++
               PRINT TABLE DATA START
                     DISPLAY ThumbNailArr[(a*PicturesPerRow)+b]
                     ON CLICK ON THUMBNAIL
                        DISPLAY FullImageArr[(a*PicturesPerRow)+b]
               PRINT TABLE DATA END
           END FOR
     PRINT ROW END
  FOR END

TABLE CLOSED
```

Code

```
include 'includes/mysqlconnect.php';

$sql =  "SELECT * FROM members";
$query = mysqli_query($con,$sql);
$getID = $_GET['id'];
if(isset($getID)){
      echo "<script src='https://ajax.googleapis.com/ajax/libs/jquery/2.0.2/jquery.min.js'
type='text/javascript'></script>";
      echo "<script src='js/jquery.lighter.js' type='text/javascript'></script>";
      echo "<link href='stylesheets/jquery.lighter.css' rel='stylesheet'
type='text/css'/>";
            echo $getID . " Gallery->";
            $uid = $_GET['id'];
            $sql2 = "SELECT * FROM members WHERE id='$uid'";
            $query2 = mysqli_query($con,$sql2);
            $userArr = mysqli_fetch_assoc($query2);
            echo $userArr['UserName'];
            $sql3 = "SELECT PathResized FROM uploads WHERE uid='$uid'";
            $sql4 = "SELECT FileName FROM uploads WHERE uid='$uid'";
```

```
            $query3 = mysqli_query($con,$sql3);
            $query4 = mysqli_query($con,$sql4);
            $imgArray = array();
            while($row = mysqli_fetch_row($query3)){
                  array_push($imgArray,$row[0]);
            }
            $imgFileNameArray = array();
            while($row = mysqli_fetch_row($query4)){
                  array_push($imgFileNameArray,$row[0]);

            }
            $picsPerRow = 5;
            $numRows = ceil(mysqli_num_rows($query3)/$picsPerRow);
            echo "<table border='1' width='700'>";
            for($j=0;$j<$numRows;$j++){
                  echo "<tr>";
                  for($i=0;$i<$picsPerRow;$i++){
                        echo "<td>";
                        echo " <a href=' http://fpp.mylesrank.in/fpp/uploads/" .
$imgFileNameArray[($j*$picsPerRow)+$i] . "' data-lighter>";
                        echo "<img src='" . $imgArray[($j*$picsPerRow)+$i] . "'/>";
                        echo "</a>";
                        echo "</td>";
                  }
                  echo "</tr>";
            }
            echo "</table>";
}else{
      echo "<b>User Galleries:</b></br>";
      while($row = mysqli_fetch_array($query))
   {
      $id = $row['id'];
      echo "<a href='?id=" . $row['id'] . "'>" . $row['UserName'] . "</a>";
      echo "<br>";
      }
}
```

Algorithm: Registering new users	**Location:** /fpp/register.php

Description

The code in register.php handles all new user registrations. It grabs data from the HTML registration form and parses it through validation before being stored into a members table in the database. It also handles hashing and generates a salt for each new user registering. The hashing works by first combining the salt generated for the user when registering, then combines this with the users password. The result of appending the salt and username is then hashed using the PHP SHA1() function which creates a near irreversible hash so that the passwords are stored securely. When the password needs to be checked with a say, login the salt is grabbed from the database again and the password that needs to be checked is then hashed using the same process. Once it has been hashed, it then compares with the hashed password generated by this algorithm to confirm they are the same password.

Pseudo-code

```
INCLUDE MYSQL_CONNECTION MODULE

VAR postUsername  = POST['USERNAME']
VAR postFirstname = POST['PASSWORD']
VAR postLastname  = POST['LASTNAME']
VAR postRegkey    = POST['REGKEY']
VAR postPassword  = POST['PASSWORD']

VAR salt = OPENSSL_RANDOM_PSEUDO_BYTES(32)
```

```
VAR password = SHA1(postPassword+salt)


IF EMPTY(postUsername) = TRUE
   PRINT "UserName field required"
   END
IF EMPTY(postPassword) = TRUE
   PRINT "Password field required"
   END
IF EMPTY(postFirstname) = TRUE
   PRINT "First name field required"
   END
IF EMPTY(postLastname) = TRUE
   PRINT "Last name field required"
   END

VAR userCheckQuery = MYSQL_QUERY(connection,SELECT UserName FROM members WHERE
UserName = 'postUsername')

VAR userCheck = userCheckQuery->FETCH_OBJECT

IF userCheck IS SET = TRUE
   PRINT "Username taken"
   END

VAR regkeyCheckQuery = MYSQL_QUERY(connection, SELECT PermissionLevel FROM regkeys
WHERE RegKey = 'postRegkey')

VAR regkeyCheck = regkeyCheckQuery->FETCH_OBJECT
VAR permissionLevel = regkeyCheck->PermissionLevel

IF regkeyCheck IS SET = TRUE
   MYSQL_QUERY(connection,UPDATE members SET PermissionLevel =
   'permissionLevel' WHERE UserName = 'postUsername')
ELSE
   PRINT "No valid registration key was found, you have been
registered without permissions"

VAR sqlQuery1_insertdata =
INSERT INTO members (UserName, Password, Firstname, Lastname, salt, regdate) VALUES
(postUsername, password, postFirstname, postLastname, salt)

IF MYSQL_QUERY(connection,sqlQuery1) RETURNS ERROR
   PRINT "ERROR: " + MYSQL_ERROR(connection)
   END
ELSE
   PRINT "Success, you have been registered"
```

Code

```
include 'includes/mysqlconnect.php';

$salt = openssl_random_pseudo_bytes(32);
$password = sha1($_POST[password] . $salt);

$currentdate = date("Y-m-d");

$pusername = $_POST['username'];
$pfirstname = $_POST['firstname'];
$plastname = $_POST['lastname'];
```

```
$pregkey = $_POST['regkey'];

$sql="INSERT INTO members (UserName, Password, Firstname, Lastname, salt, regdate)
VALUES
('" . mysqli_real_escape_string($con, $pusername) . "','" .
mysqli_real_escape_string($con, $password) . "','" . mysqli_real_escape_string($con,
$pfirstname) . "','" . mysqli_real_escape_string($con, $plastname) .
"','$salt','$currentdate')";

$ppassword = $_POST['password'];
if(empty($pusername)){
      die("Username field empty");
}
if(empty($ppassword)){
      die("Password field empty");
}
if(empty($pfirstname)){
      die("First Name field empty");
}
if(empty($plastname)){
      die("Last name field empty");
}

$usercheck = mysqli_query($con, "SELECT UserName FROM members WHERE UserName = '" .
mysqli_real_escape_string($con, $pusername) . "'");
$usercheck_obj = $usercheck->fetch_object();

if (isset($usercheck_obj)){
      die("Username taken");
}

$regkeycheck = mysqli_query($con, "SELECT PermissionLevel FROM regkeys WHERE RegKey = '" .
mysqli_real_escape_string($con, $pregkey) . "'");
$regkeycheck_obj = $regkeycheck->fetch_object();
$permissionLevel = $regkeycheck_obj->PermissionLevel;

if(isset($regkeycheck_obj)){
      mysqli_query($con, "UPDATE members SET PermissionLevel = '$permissionLevel' WHERE
UserName = '" . mysqli_real_escape_string($con, $pusername) . "'");
      echo "A valid RegKey was found during registration, you have been registered with
permissions.";
}
else{
      echo "<p>No valid RegKey was found during registration, you have been registered
without permissions.</p>";
}

if (!mysqli_query($con,$sql)){
      die('Error: ' . mysqli_error($con));
}
      echo "<h2>Success: You have been registered";
```

Complete code listings

Complete code listings can be found at the end of this document in Appendix 1.

List of libraries used in system

1. **Lightbox image preview library (Javascript) -** http://lokeshdhakar.com/projects/lightbox2/ **-** Used for creating image preview popups on the manage and gallery pages.
2. **Highcharts graph rendering library (Javascript/HTML5) -** http://www.highcharts.com/ **-** Used in generating a pie chart in the fileusage page as seen at the beginning of the systems maintenance section.
3. **Simple image manipulation library (PHP Function) -** https://github.com/claviska/SimpleImage **-** Used for resizing images during upload for thumbnails used in the gallery page.

User Guide

Introduction to Furze Platt Photography Management System

FPPMS is a web program designed to allow the Furze Platt Photography group to store, manage and distribute photographs taken by the group from events and other activities. The system is coded in PHP, which is a web programming language, you access the system by navigating to it through a browser. Each user has details to login to the system, with each user having varying permission levels to view each page.

System Requirements

In order to access the system you will need the following:

- A computer that is able to run a browser.
- An internet connection.
- An internet browser that fully supports HTML5/CSS3 – *Chrome, Firefox, Safari, Opera…*

Peripherals:

- Keyboard
- Mouse
- Monitor
- (Optional) Camera – To take photos to put onto the system.
- (Optional) SD Card reader – For transferring files from cameras.

Compatibility top 5 browsers based on html5.com and css3test.com scores

Browser	*CSS3*	*HTML5*
Google Chrome (Build 33)	✓	✓
Opera (Build 20)	✓	✓
Firefox (Build 28)	✓	✓
Safari 7.0	✓	✗
Internet Explorer 11	✓	✗

**Disclaimer: Internet Explorer is not recommended to use for this system as it does not have the necessary support for elements used throughout the system.*

Accessing the system

First of all, to access the system you will be required to navigate to the URL of website through the browser of your choice. You need to open your browser and type http://fpp.mylesrank.in/ into your web address bar as pictured in Fig. 1.

This will land you on the login page which looks like the picture in Fig. 2 below.

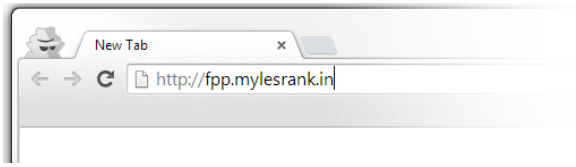


Fig. 1

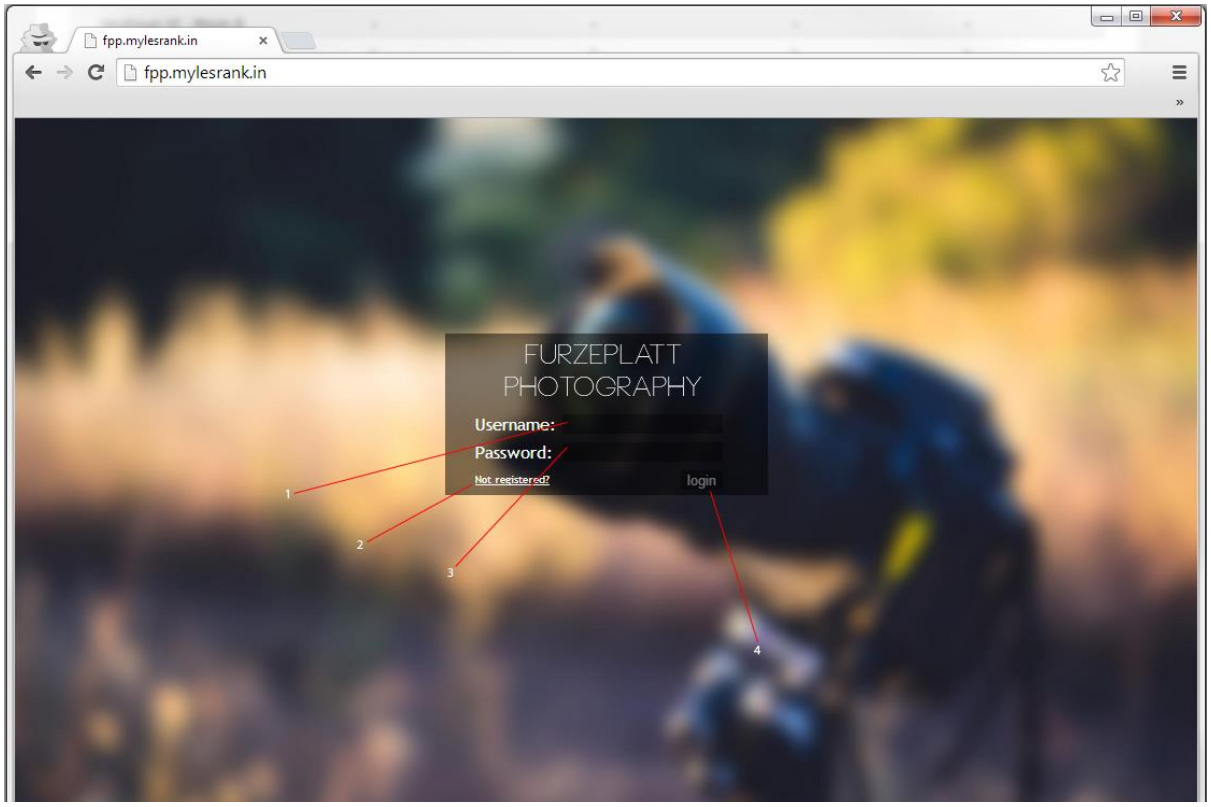


Fig. 2

1- **Username login form box –** User is required to enter username here.
2- **Redirection to registration page button –** This button will take you to the login page.
3- **Password login form box –** User is required to enter password here.
4- **Login form button –** This button will login user once login details are filled in.

Once you are on the login page, If you already have details you simply just fill out your username and password (labelled as 1 and 3 in Fig. 2) then press the login button (label 4 Fig. 2), you will be forwared to the main UI seen in Fig. 4.

If you enter the wrong username/password details then you will see error messages saying "Incorrect Username" and "Incorrect Password", once these are displayed you can just simply re-enter different details and try again.

If you do not have login details, you can press the "Not registered?" button (label 2 Fig. 2), which will direct you to the registration page pictured in Fig. 3 below.

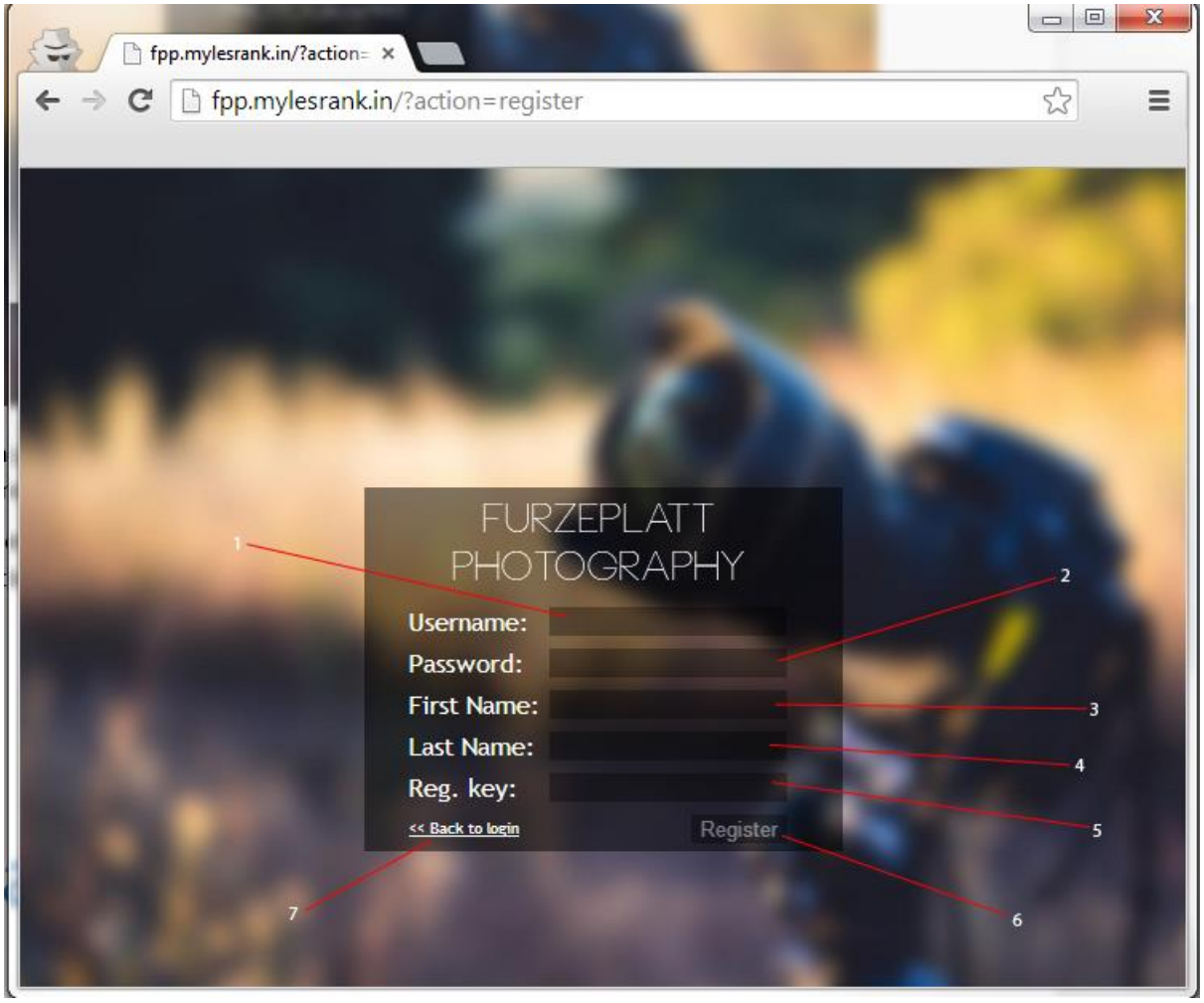


Fig. 3

1- **Username registration form box –** This is where the user registering enters a username to use.
2- **Password registration form box –** This is where the user registering enters a password to use.
3- **First Name Registration form box –** This is where the user registering enters their First Name.
4- **Last Name Registration form box –** This is where the user registering enters their Last Name.

Once directed to this page if you are not registered, fill out your desired Username[1] and Password[2], followed by your First[3] and Last[4] Name. If you have been given a Registration Key[5] from an Admin to get higher than normal permission level, then enter that into the Reg. key box. All these fields are labelled in parts 1-5 in Fig. 3. When all fields have been filled out, hit the Register button[6] and you will be registered and forwarded to a page confirming your registration as seen in Fig. 3.1 on the next page.

If you have accidently pressed the Not Registered button and want to go back to the login, simply hit the Back to login button/text[7] and this will redirect you to the login page.

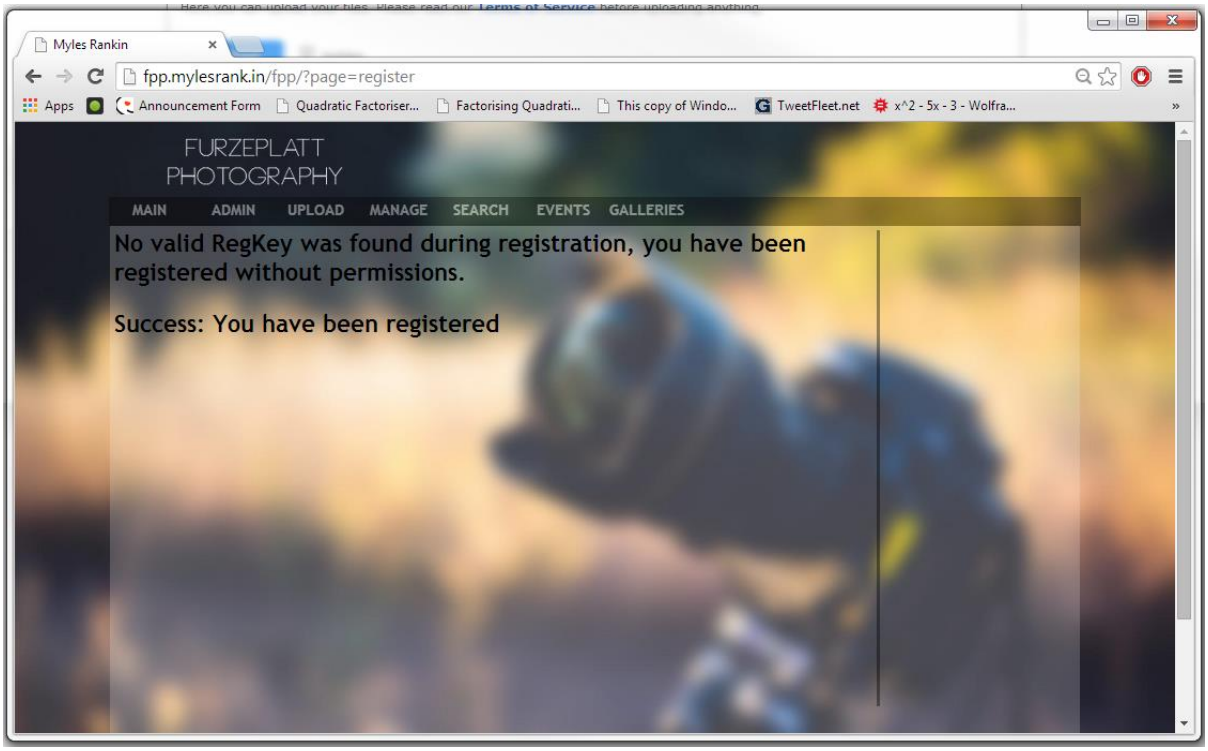


Fig. 3.1

If you have not entered a Registration Key then you will see that you will be registered without a permission level. If you have entered a valid Registration Key, then you will see a different message as shown in Fig. 3.1.1 and will be registered with permissions.

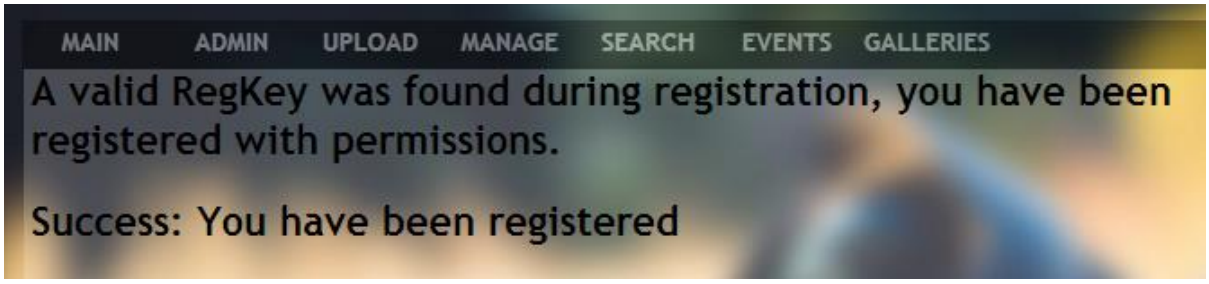


Fig. 3.1.1

Using the system

Index page – How to use the main UI

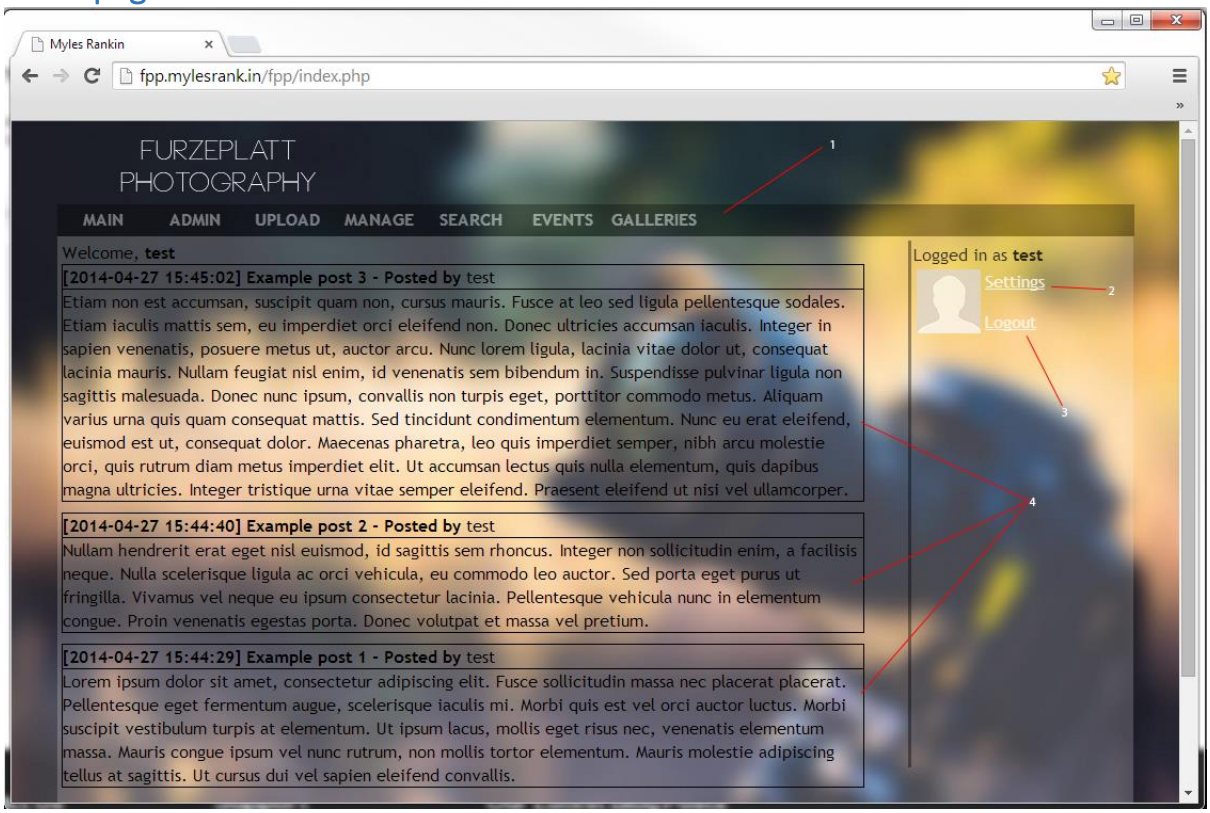


Fig. 4

1- **Navigation bar –** From here you can navigate to any page in the system.
2- **Settings page button –** Clicking this will take you to the settings page where you can change your password.
3- **Logout page button –** Clicking this will log you out ending your session.
4- **New Posts –** These are news posts created by admins which are displayed on the main page.

This is where you are directed to once logged into the system (You can also access and view this page when you are not logged in, but there will be a message prompting you to go and login). To navigate the site, you use the navigation bar labeled 1 in Fig. 4 at the top of the page underneath the FPP logo. Each page has a button in the navigation bar which will redirect you to it. There are also news posts displayed on the main page which admins are able to post, these are labaled 4 in Fig. 4. Each post has a post date, title, poster name and content of the post. Additonally across the site, there is an aside bar which is seen on the right of the page. This has a message displaying who you are logged in as then two buttons below it which take you to the settings (label 2 Fig. 4) page and a logout button (label 3 Fig. 4) which will end your current session.

Upload page – How to upload files

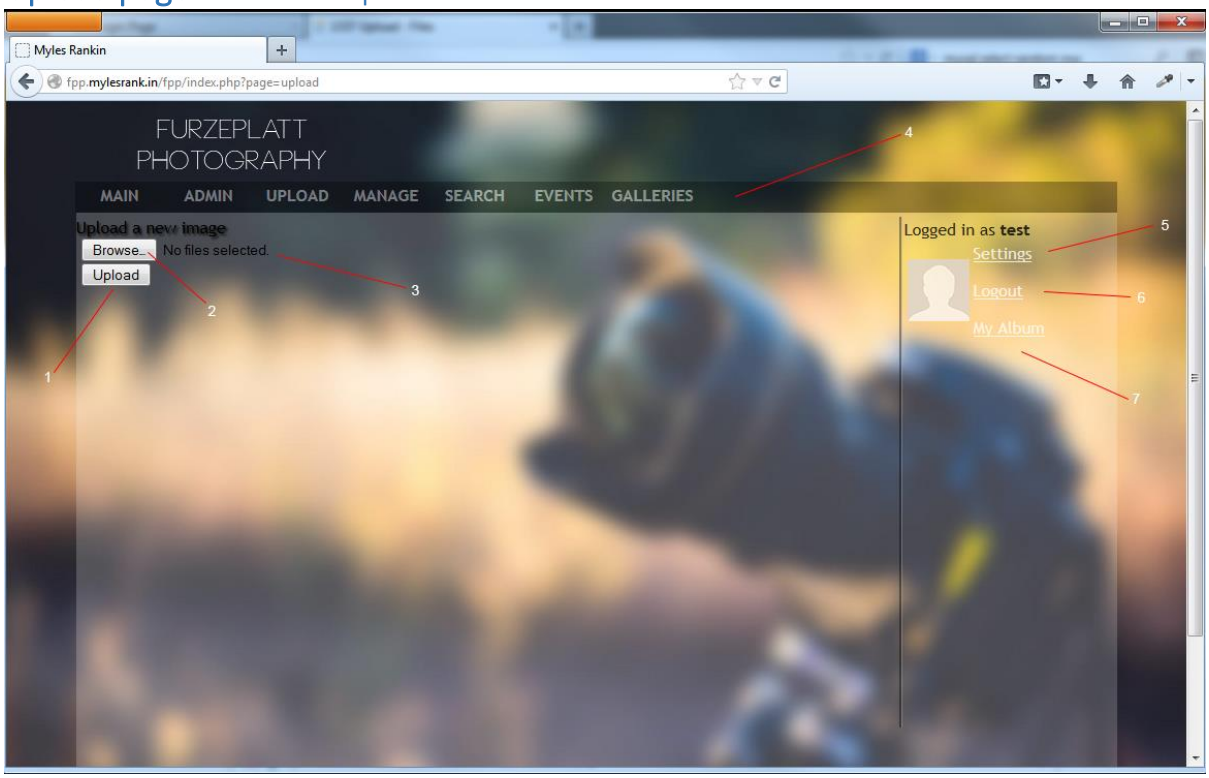


Fig. 5

1- **Upload button –** After selecting photos to upload by first pressing the browse button (labelled 2 in Fig. 5)
2- **Browse button –** The button you press before going to the upload button. By pressing this it will open a browse window to select image files as seen in Fig. 5.1 below.
3- **Selected files output –** This will display the number of files that have been selected to be uploaded from the browse window.
4- **Navigation bar –** From here you can navigate to any page in the system.
5- **Settings page button –** Clicking this will take you to the settings page where you can change your password.
6- **Logout page button –** Clicking this will log you out ending your session.
7- **My albums** – This button will take you to your current user album of uploaded files.

This page (Fig. 5) is where you go to upload images to the website. To start the process of uploading an image, first navigate to this page using the navigation bar when you login, the button for this upload page is conveniently named “Upload”. Once here, you will see a few buttons. To select an image to upload press the browse button labelled 2 in Fig. 5. This will open a window shown as Fig.5.1 on the right.

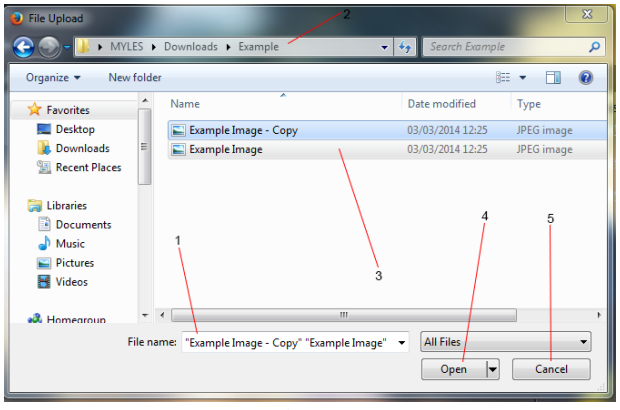


Fig. 5.1

1- **Current images selected**
2- **Current folder path**
3- **Images to select**
4- **Open images into uploader**
5- **Cancel selection**

To use the uploader as seen in Fig. 5.1, simply hold Ctrl on your keyboard and then select or drag a box over the images you wish to select for upload. Once the images are highlighted as seen labelled 3 in Fig. 5.1 and after checking which images are selected both through looking at highlighted ones and the box labelled 1 in Fig. 5.1 which shows current selected images, you can press the open button labelled 4 in Fig. 5.1. Once the open button is pressed, the browse window will close and you will see that the upload page has x number of files selected ready for upload as seen in Fig. 5.2 labelled 2.

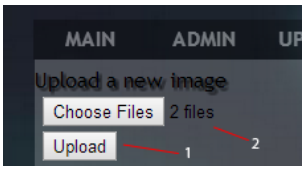


Fig. 5.2

Fig. 5.3

Once the files are selected you can hit the upload button (labelled 1 in Fig. 5.2). After uploading, the page will print a confirmation message for each file you selected for upload in the format "File *filename* has been uploaded." This can be seen on the left in Fig. 5.3 as label 2.

Manage page – How to manage your uploaded photos

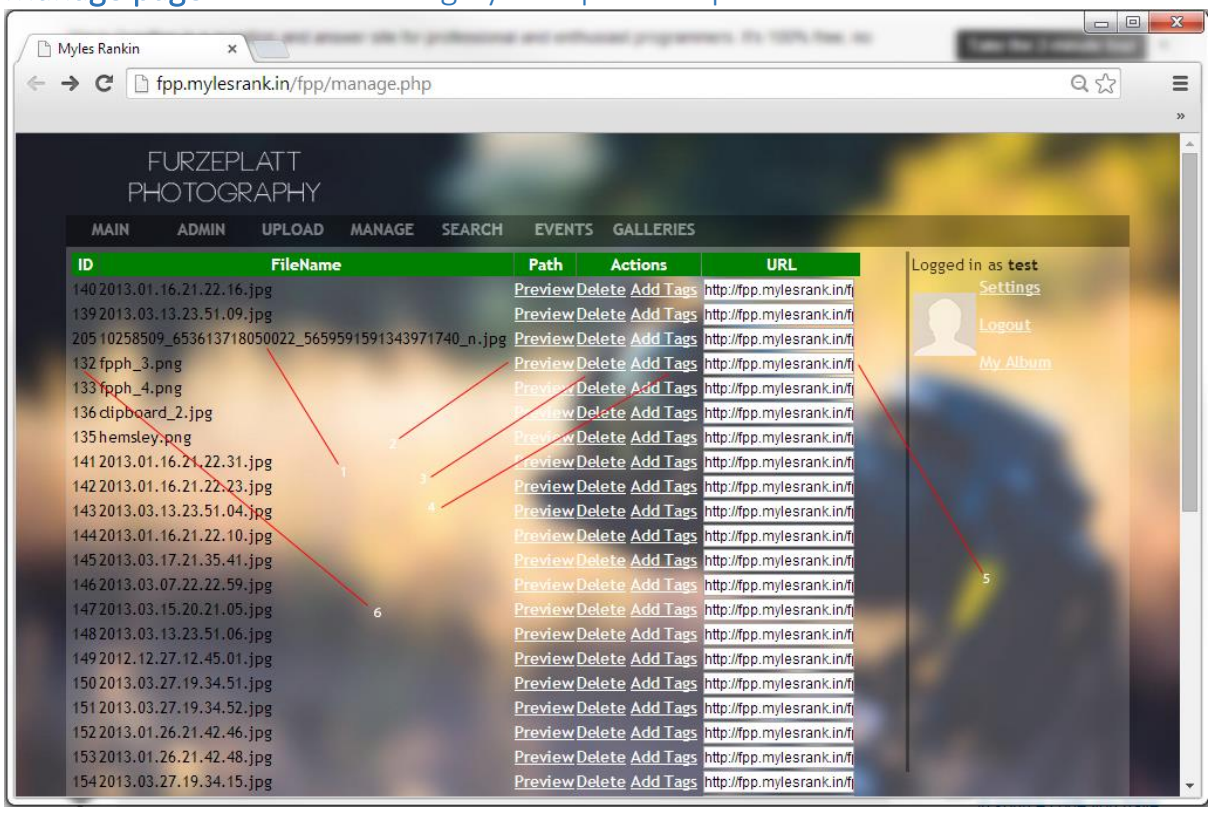


Fig. 6

1- **Filename –** This is the name of the file on that specific row in your table of uploaded files.
2- **Preview button –** This button allows you to preview the image on its row in a popup window.
3- **Delete button –** This button allows you to delete the image on its row.
4- **Add tags button –** This allows you to view and add tags to the image on its row.
5- **URL box –** This box has the url of the image on its row which when clicked, highlights it for copy & paste use.
6- **Unique File ID –** This is the unique ID of the file used for identifying each file.

This page is where you manage your photos, you can access it by pressing the manage button in the navigation bar (labelled 1 in Fig. 4). You are able to preview, delete, add tags, copy URL's and view filenames in a table for each image. The filename for each file can be seen under the filename column labelled 1, to manage each file look under the "Actions" column with each row having "Delete" and "Add Tags" buttons, labels 3 & 4. To delete a file, find the specific row of the image you wish to delete and press the delete button. You will be prompted with a warning message as shown in Fig. 6.1. To confirm deletion of file, press the OK button labelled 1 in Fig. 6.1. If you wish to cancel deleting the file hit either the, Cancel button or the X in top right of the window both labelled 2 in Fig. 6.1.

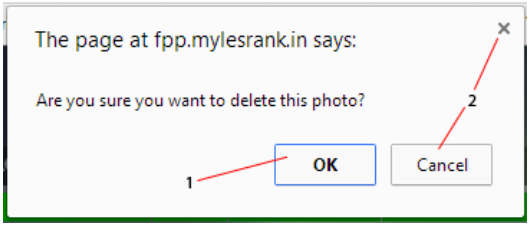


Fig. 6.1

1- **Confirmation button to delete file.**
2- **Cancel button(s) – "Cancel and X"**

Once deletion is confirmed you will see a confirmation message telling you which file was deleted in the format "*Filename* has been successfully deleted." as seen in Fig. 6.2 on the right. After seeing the deletion message you will be redirected back to the management page.

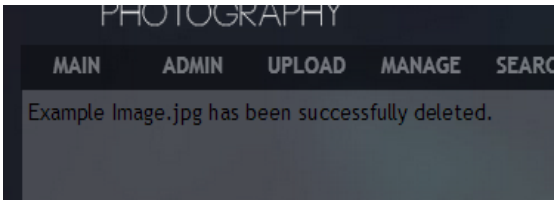


Fig. 6.2

Moving onto the second aspect of actions that you are able to manage your photos with is adding tags to them for use when searching. When on the manage page, find a photo you wish to add tags to by searching for it by row. Once found to add tags, simply press the "Add Tags" button labelled 4 in Fig. 6. This will forward you to a page to specifically edit and view current tags for that file as seen in Fig. 6.3 below.

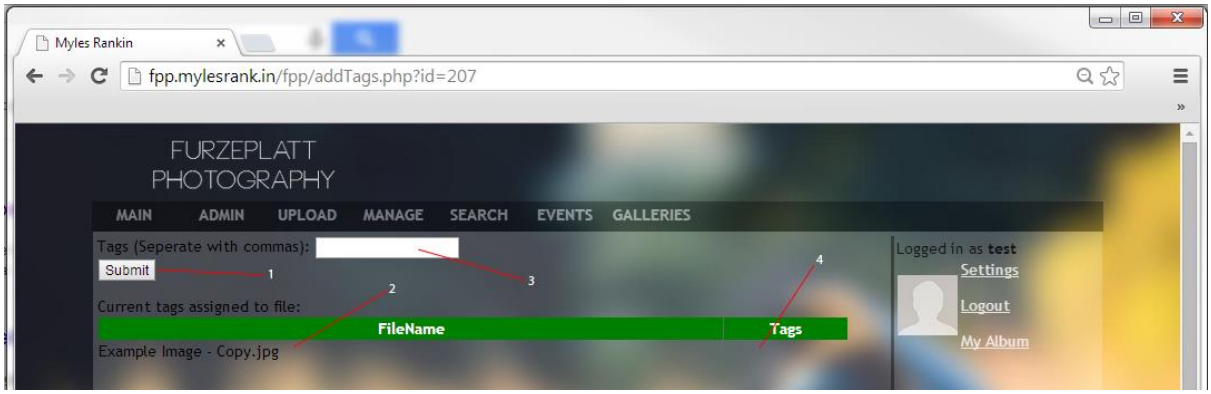


Fig. 6.3

1- **Submit button –** Used to submit tags entered in labelled 3 text box in Fig. 6.3.
2- **File Name of image to add tags to –** This is the filename of the image you are giving tags to.
3- **Tags text box –** Text box field for typing in tags to be added to image
4- **Current tags –** This is the position where current tags for the image are displayed.

To start adding tags, type in your desired tags with commas. This is shown on the right in Fig. 6.3.1 with examples of the tags: Example 1 and Example 2.

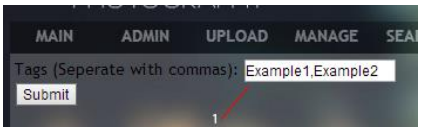


Fig. 6.3.1

1- **Tags being entered**

Once you have typed your desired tags, press the submit button labelled 1 in Fig. 6.3. The tags will then be assigned to the file and the assigned tags table will update showing the tags you entered as shown below in Fig. 6.3.2:

Fig. 6.3.2

1- Current assigned tags

In addition to being able to manage photos on the manage page, you are also able to preview and copy URLs for each photo.

To preview a photo, find the preview button on the row of the image you wish to preview and click it. It will bring up a window with the photo contained within and there will be a cross in the top left of the image which closes it. This is shown in Fig. 6.4 below:

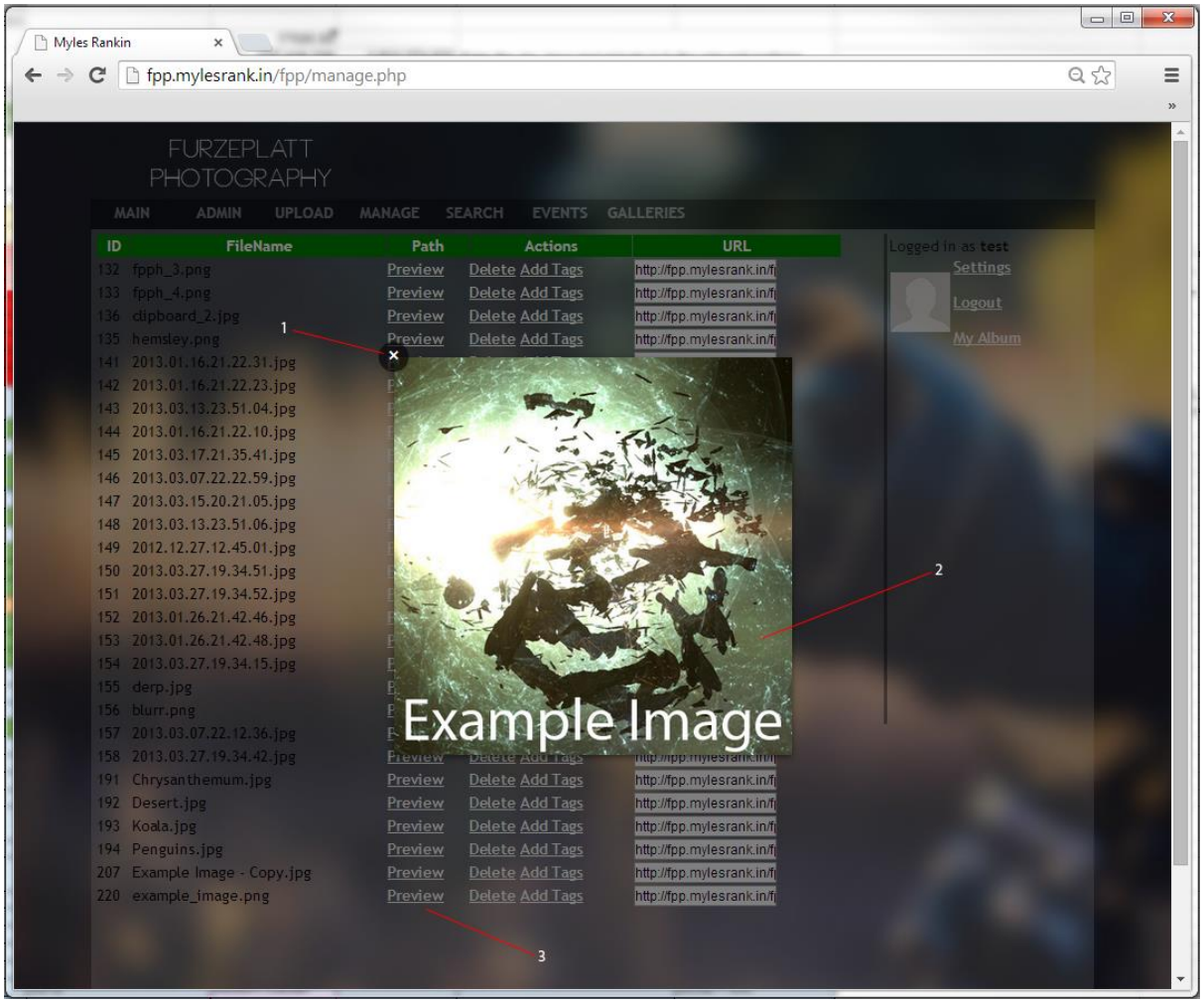


Fig. 6.4

1- **(X) button to close previewed image –** Click this to remove preview mode and return to manage page.
2- **Previewed image –** This is the image from the row selected to preview.
3- **Preview button –** Button pressed to open a previewed image of the image in the row selected.

To copy a URL as mentioned previously, find the image you wish to copy a URL for and navigate across its row till you are under the URL column as shown in Fig. 6 label 5. Then simply click the box and the URL be automatically highlighted, you then can either right click then press copy or hold down Ctrl + C so the link is copied to your clipboard. Once this is done you can then paste the link to where ever you desire by right clicking and pressing paste or by holding down Ctrl + P to also paste. This is shown in Fig. 6.5 on the right.

Pasting into an application (Example being Notepad) is shown below in Fig. 6.5.1

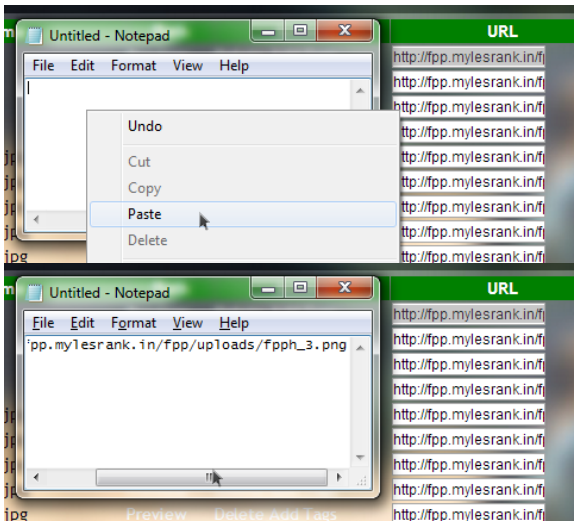


Fig. 6.5.1

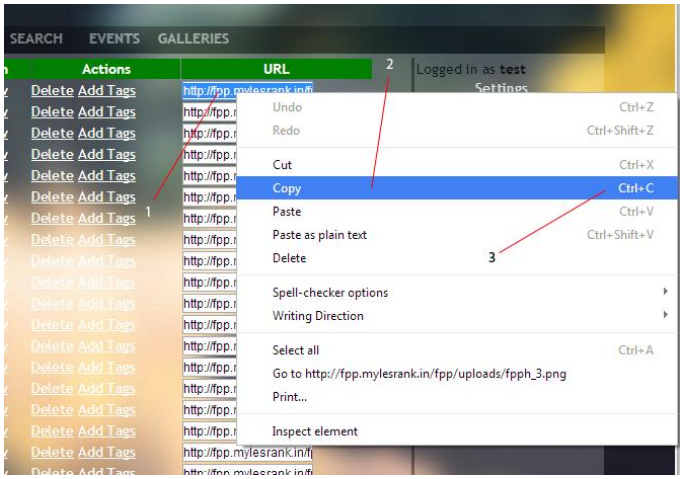


Fig. 6.5

1- **URL –** This is where you click then either right click + copy or Ctrl+C to copy to your clipboard.
2- **Copy in rightclick menu –** This is the copy selection when you right click the highlighted URL.
3- **Copy with keystrokes –** This is the keystroke combination to copy the highlighted text.

Search page – How to search for files

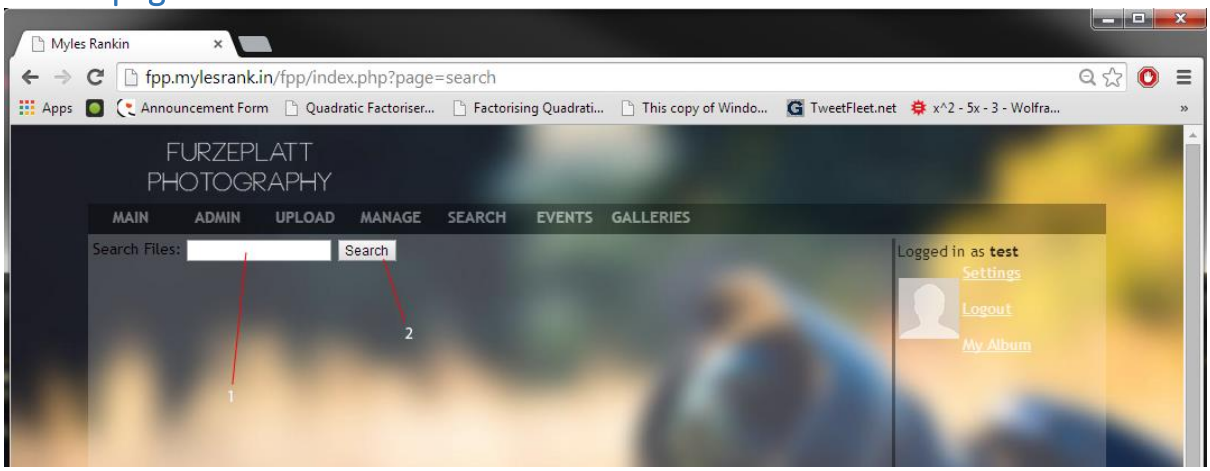


Fig. 7

1- **Search parameters text box –** This is where you enter the search parameter for the file you wish to search for.
2- **Search button –** This is the button you press once search parameters have been filled out

To search for a file, simply type a search parameter for a file you want to search in the textbox labelled 1 in Fig. 7. Once the search text boxed has been filled, press the Search button labelled 2 in Fig. 7. This will then show you a list of results shown in Fig. 7.1

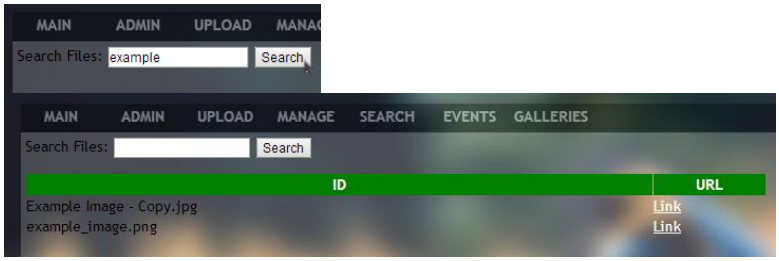


Fig. 7.1

When searching, anything put in the search box will search for filenames, file ID's and tags associated with each image to get a result.

This means you can tag a bunch of pictures with an event name to associate them to it, so when you want someone to find them, you can tell them to simply search for the event name on the search page.

Events page – How to view events

If you refer back to Fig. 4 label 1 or if you remember where the navigation bar is in the system, find the button that is called Events. By clicking this will take you to the events page which is shown below in Fig. 8.

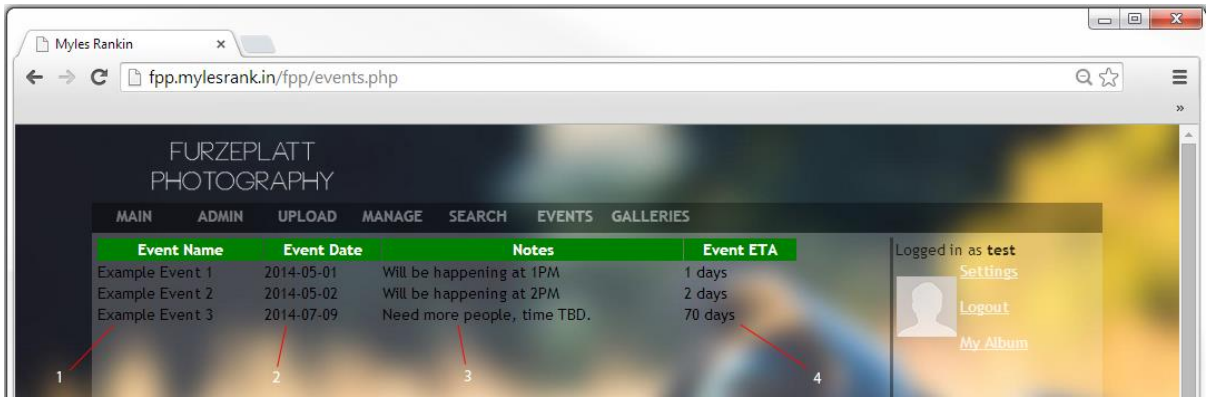


Fig. 8

1- **Event Name -** Here you can see the name of the event on each row.
2- **Event Date –** This is where you can see when the event is going to take place.
3- **Event Notes –** This is any notes an admin has attached to the event.
4- **Event ETA –** This number of days is how long until the event takes place.

Each event is displayed here, with an event name (label 1), date(label 2), notes(label 3) and ETA until the event(label 4). Any events that are dated before current date are automatically removed, so everything listed here is up to date. The ETA automatically updates each time you load the page, thus it is accurate.

Gallery page – How to view other users galleries of photos

Find the navigation bar (Fig. 4 label 1) on the page you are on and click the "Galleries" button. This will take you to a page shown below in Fig. 9

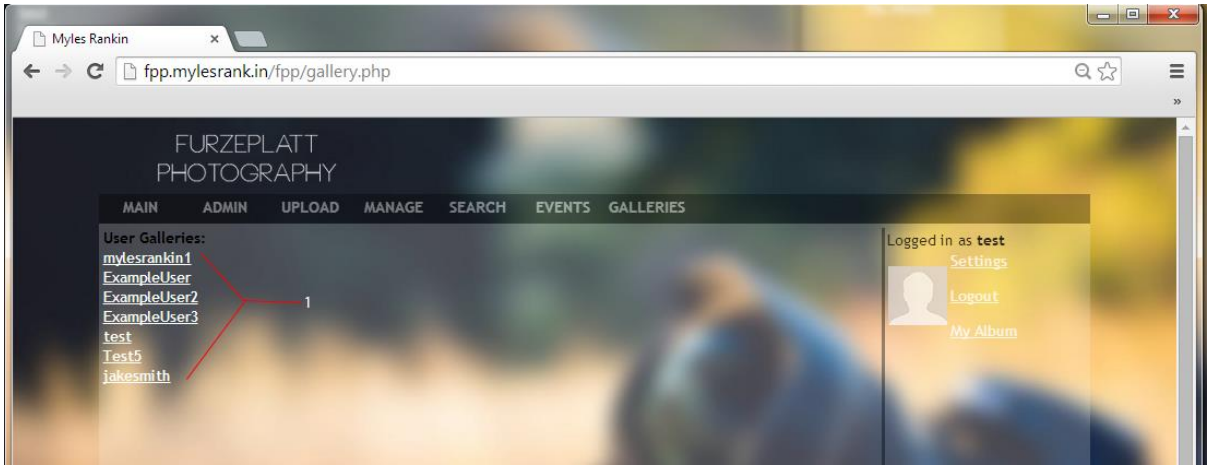


Fig. 9

1- **User gallery links –** Clicking on these links will take you to each users gallery page.

As seen in Fig. 9 the page displays all registered users on the site, all the names are links to their gallery page. To browse to a user's gallery, just simply find their username and click it. This is shown for user test in Fig. 9.1.

All images that belong to a user are displayed in a grid as shown on the right in Fig. 9.1. They are all cropped to fit into this grid with 5 being shown on each row, and the page will just extend to as many rows needed.

You are able to preview each image to it's full size in the grid like you can in the manage page. To do this click one of the cropped images (one of them labelled 3 in Fig. 9.1), this will bring up a pop up box with the full size image in. To close this box, find the (x) in the top left of the image, and press it. This will close the preview and you will see the gallery page again. These features are shown in Fig. 9.1.1 below

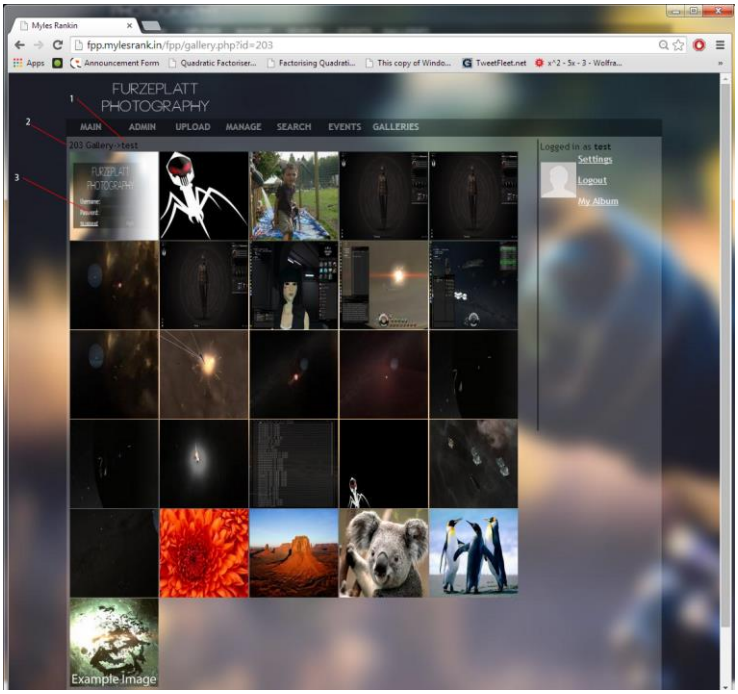


Fig. 9.1

1- **Current user gallery –** The username of the gallery you are browsing
2- **Current user ID –** The user ID of the gallery you are browsing
3- **Cropped images –** Images from a user's gallery are cropped and displayed in this grid.

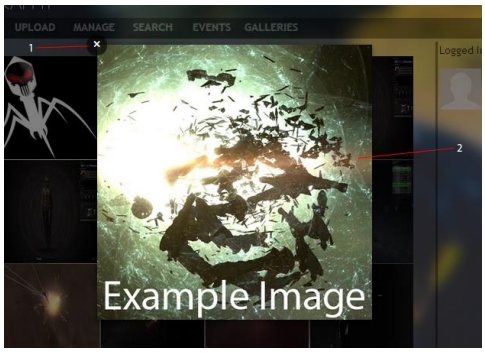


Fig. 9.1.1

1- Close preview button 2- Previewed image

Settings page & Logging out – How to

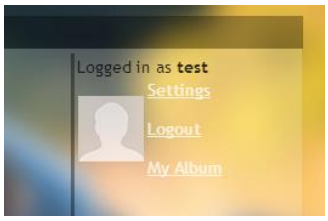


Fig. 10

Throughout all pages in the aside widget on the right, there are 3 links. Settings, Logout and My Album. These are shown in Fig. 10 to the right. The My Album button, just forwards to your own gallery as shown in how to view other users galleries of photos. There are another two buttons that do two key functions in the system.

The settings page allows you to change your password, and the Logout button ends your session in the system.

First of all, to change your password simply press the Settings button as seen in Fig. 10. This will take you to the settings page as shown below in Fig. 10.1.

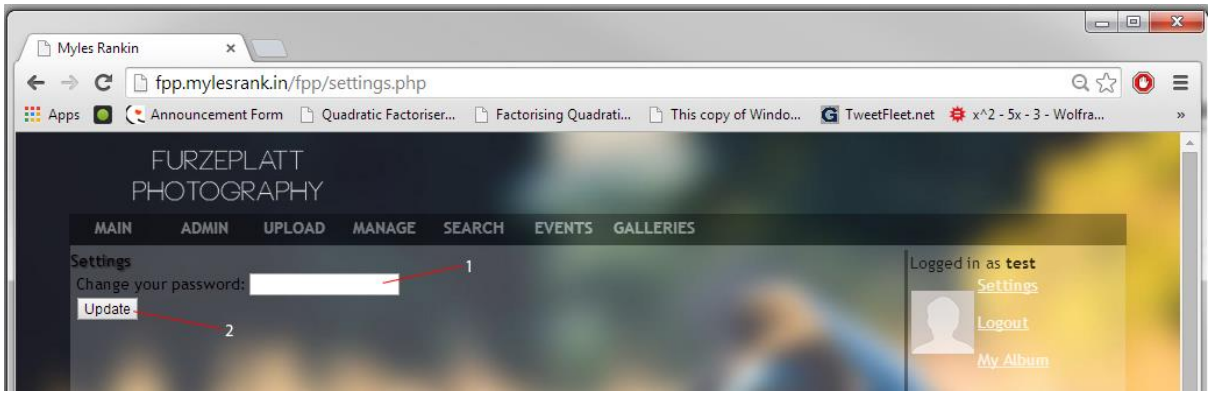


Fig. 10.1

1- **New password input field**
2- **Update button**

To update your password simply put a new password in the password input field labelled 1 in Fig. 10.1 and then press the Update button labelled 2 in Fig. 10.1. The page will then refresh and your password will be changed.

To Logout, find the logout button in Fig. 10 and press it. Once you have pressed it, you will be redirected to a logout success message/page saying "You have logged out." And have your session ended as shown below in Fig. 10.2 then back to the main login screen shown in Fig. 2.

Fig. 10.2

Admin page – How to manage the system

If you have the correct permission levels to be an administrator then you will have access to the admin page. To see If you have the required level, simply find the Admin button in the navigation bar and click on it.

If you do not have the required permissions level then you will see a message shown in Fig. 11.1 to the right.

Fig. 11.1

If you do have the required permissions you will be taken the admin page shown in Fig. 11 below.

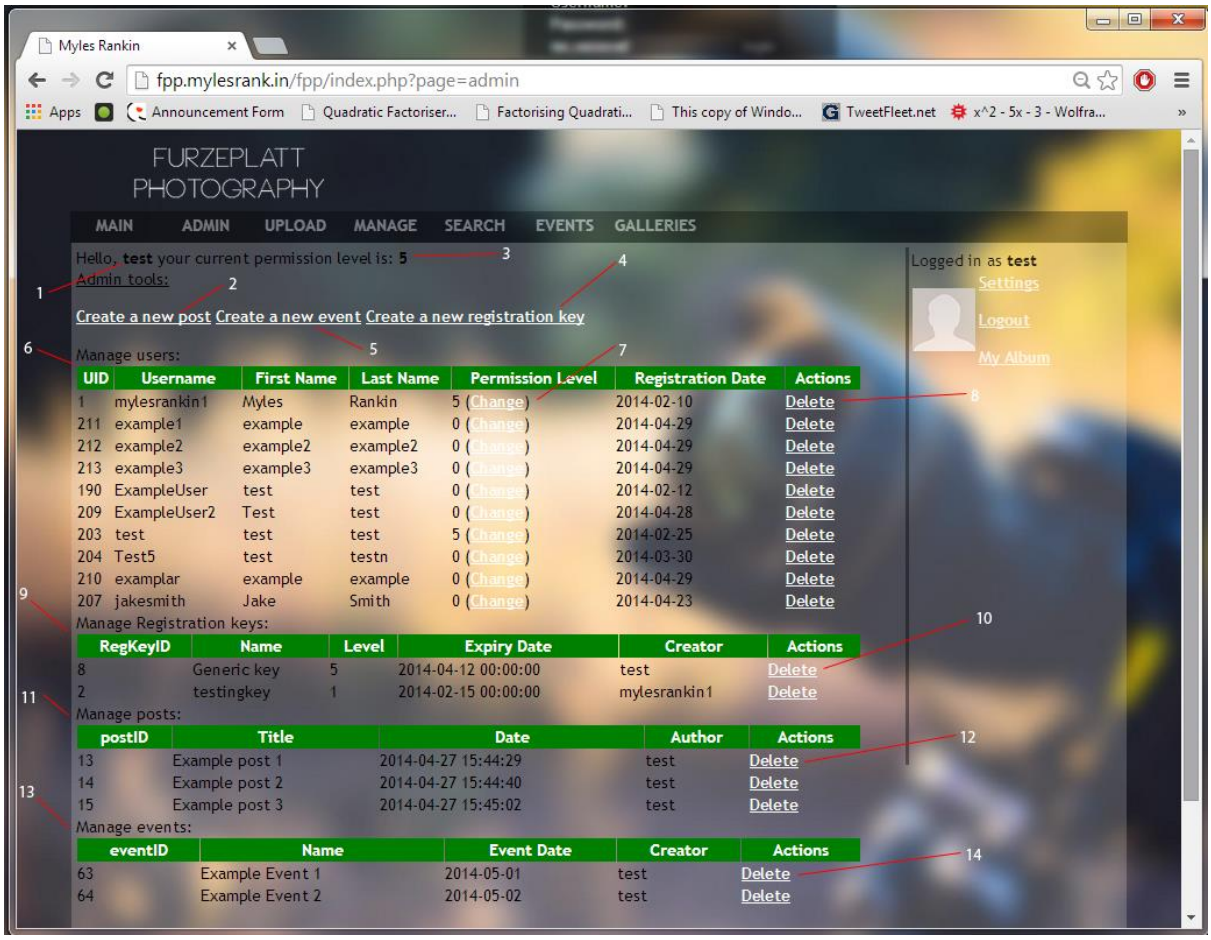


Fig. 11

1- **Who you are currently logged in as**
2- **Create a new post button**
3- **Your current permission level**
4- **Create a new Registration key button**
8- **Delete user button**
9- **Manage Registration keys table**
10- **Delete Registration key button**
11- **Manage posts table**

5- **Create a new Event button**
6- **Manage users table**
7- **Change user permission level button**
12- **Delete post button**
13- **Manage events table**
14- **Delete event button**

Admin page utilities

There are many utilities located on the admin page that are used to manage and maintain the system. You are able to create posts, events and registration keys whilst also being able to view and delete all existing ones. Additionally, you are able to manage users by being able to delete user accounts and change their permission level to restrict access to certain pages on the system.

Part 1 – User management

As mentioned you are able to manage users through the admin page. You are able to change user's permissions levels and remove their accounts from the system.

How to remove a user:

First of all, to remove a user from the system find their name in the manage users table (labelled 6 in Fig. 11) and navigate to the actions column. Then hit the delete button on the same row as their name, as labelled 8 in Fig. 11. Once you press the button you will see a warning pop up saying "Are you sure you want to do this?" to proceed in deleting the user, press OK, to stop the deletion press Cancel or the (x) in the top right of the pop up and it will close leaving you still on the Admin page.

Once you have pressed OK in the pop up you will be briefly directed to a page confirming that the user has been deleted. This page and message are shown in Fig. 11.1 on the right.

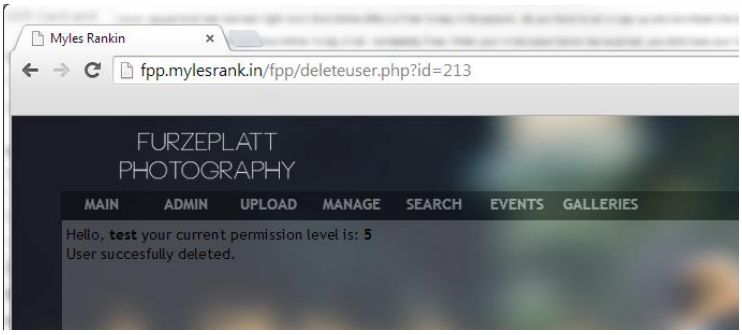


Fig. 11.1

Changing a user permissions level:

So the second action you able to do when managing users is change their permission level, to do this find the manage users table (labelled 8 in Fig 10). Then find the user you wish to change their permission level, once found you can move your mouse across so that you are under the Permission Level column but still on the users row you wish to edit. You will see a Change button in brackets (labelled 7 in Fig. 11), press this and you will see a warning pop up saying "Are you sure you want to do this?", to proceed in editing the user permission level, press OK, to stop the process press Cancel or the (x) in the top right of the pop up and it will close leaving you still on the Admin page.

Once you have pressed OK in the pop up, you will be redirected to a page to edit the specific user you have chosen to change. This page is shown below in Fig. 11.2.

To input a new permission level, type a number desired to set as the user's level in the number box labelled 2 in Fig. 11.2. Once a number has been inserted, find the submit button which is labelled 1 in Fig. 11.2 and press it. You will see the message shown in Fig. 11.2.1 confirming the change and will be sent back to the admin page. You will be able to see that the permission level has been changed by looking under the permission level column on the row you changed, as it will be set to what number you edited it to.

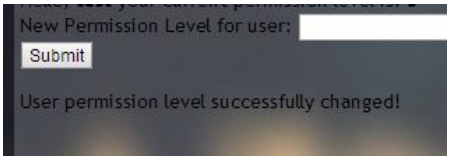


Fig. 11.2.1

Part 2 – Registration key management

In the Admin page, you are able to create and delete Registration keys which are given to users to register accounts with certain permission levels at registration.

How to create registration keys:

First to create a registration key, find the "Create a new registration key" text labelled 4 in Fig. 11. Once you have found it, click on it and you will be directed to a page shown in Fig. 11.3 below.

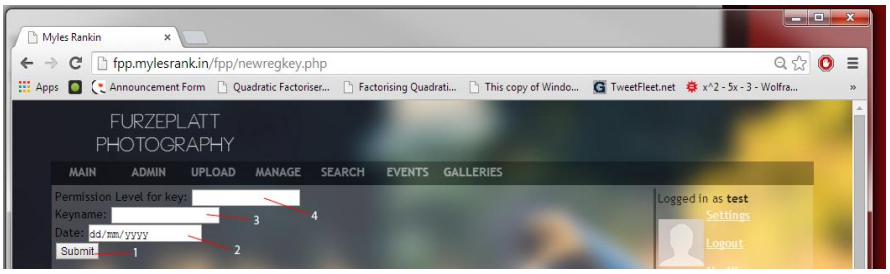


Fig. 11.3

1- New regkey submit button
2- Regkey expiry date input
3- Regkey name field input
4- Regkey permission level number input

You will need to fill out all 3 fields on the page before being able to create a key. The first field is the permission level you wish to give the key, this will update a user's permission level to that number when they register. This field is labelled 4 in Fig. 11.3 and can only be a number. Second field is a the key name for the registration key, this key name is what you will give to the user to enter in during registration, it is labelled 3 in Fig. 11.3. It can be any combination of letters and numbers, as it is a variable character input. The last field is the date of which you wish the registration key to expire, due to security reasons this is a required input as all keys must expire. The field is labelled as 2 in Fig. 11.3, as it is

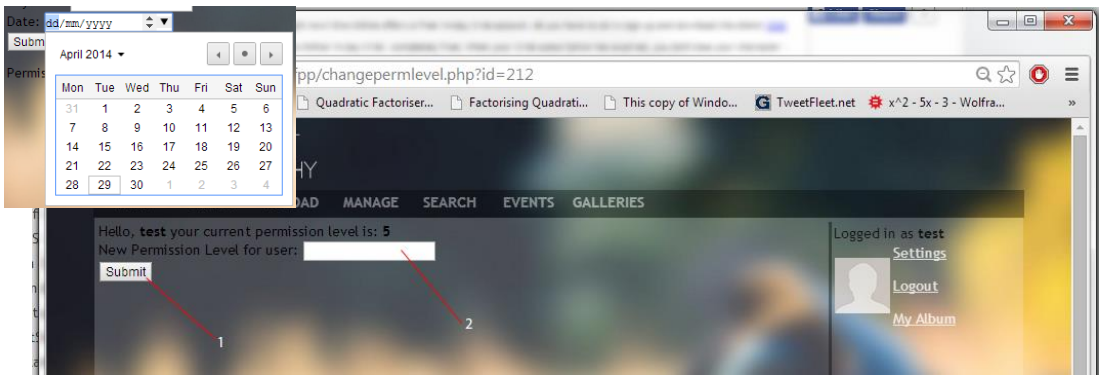


Fig. 11.2

1- Permission level change submit button
2- Permission level number input box

a date, If you click on it a calendar will appear to allow you to select a date more easily (This is shown in Fig. 11.3.1 on the left). As you can see the calendar has a simple interface, you simply use the two arrow keys on the top right to navigate the months up/down or click where it says the current month + year (April 2014 in the case of Fig. 11.3.1) to bring up a list of months to pick from. To choose a date, just select a number in the grid of days/dates.

Once all fields have been filled out find the submit button, labelled 1 in Fig. 11.3 and click it. Once you have clicked the submit button, the key will be created and you will see a "Success!" message and be redirected back to the Admin page.

How to delete registration keys:

To delete a registration key, find the manage registration keys table labelled 9 in Fig. 11. Now find the registration key you wish to delete, and move your mouse to the delete column of that row (This is shown as label 10 in Fig. 11). To delete press the "Delete" text, you will now be prompted with a warning saying "Are you sure you want to do this?" to proceed in deleting the registration key press OK, to stop the process press Cancel or the (x) in the top right of the pop up and it will close leaving you still on the Admin page.

Once you have pressed OK in the popup you see a message in a different page confirming the deletion of the registration key you choose in the manage registration key table (shown in Fig. 11.4), you will then be automatically redirected back to the Admin page.

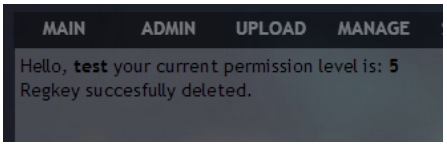


Fig. 11.4

Part 3 – Post management

In the Admin page, you are able to create and delete posts that are displayed on the front page of the systems UI.

How to create posts:

Find the text "Create a new post" labelled 2 in Fig. 11, once found click it. You will be redirected to a page to create posts, shown below in Fig. 11.5.

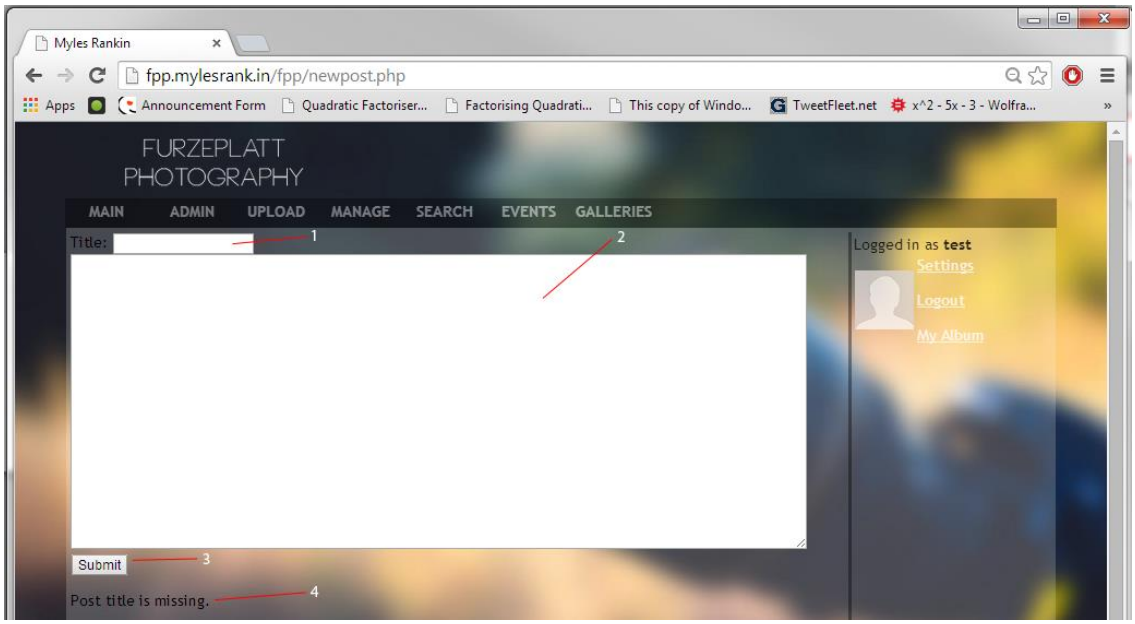


Fig. 11.5

1- Post title input box

3- Post submit button

2- Post content input box **4- Form errors**

All fields must be filled out, any errors of missing fields will be displayed under the submit button (label 3 Fig. 11.5) in the area labelled 4 in Fig. 11.5. First field is the post title field, this is the title of the post which will displayed with the post, and this field is shown in label 1 Fig. 11.5. Second field required is the post content field, which the content of the post will be entered to be displayed underneath the title on the main UI page.

Once all fields have been filled out, find the submit button labelled 3 in Fig. 11.5 and press it. The post will now be submitted and you will be redirected back to the Admin page after seeing a confirmation message shown in Fig. 11.5.1.

Fig. 11.5.1

Once posted you will be able to see the posts as shown in Fig. 4 on the main system UI, the index page.

How to delete posts:

To delete a post, find the manage posts table labelled 11 in Fig. 11. Now find the post by looking for the title you wish to delete, and move your mouse to the delete column of that row (This is shown as label 12 in Fig. 11). To delete press the "Delete" text, you will now be prompted with a warning saying "Are you sure you want to do this?" to proceed in deleting the post press OK, to stop the process press Cancel or the (x) in the top right of the pop up and it will close leaving you still on the Admin page.

Once you have pressed OK in the popup you see a message in a different page confirming the deletion of the post you choose in the manage posts table (shown in Fig. 11.6), you will then be automatically redirected back to the Admin page.

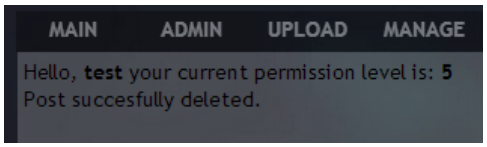


Fig. 11.6

Part 4 – Events management

The last aspect of the system you are able to manage is events. You can create, and delete them through the admin page.

How to create events:

To create an event, find the "Create a new event" text shown labelled as 5 in Fig. 11. Once found, click on it and you will be directed to a page shown in Fig. 11.7 which is the event creation page.

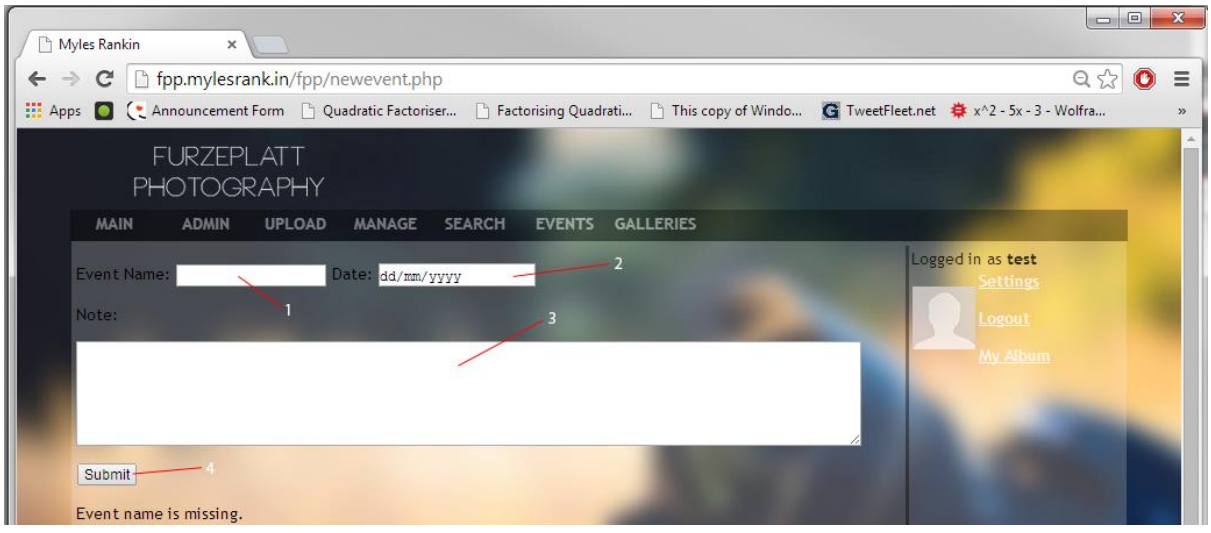


Fig. 11.7

1- Event Name input field **3- Event note input field**
2- Event Date input field **4- Event submit button**

Once you have landed on this page, you will see 3 fields. The event field is labelled in Fig. 11.7 as 1 which is required to be filled out, as does the event date field (labelled 2 in Fig. 11.7). The event note field

labelled 3 in Fig. 11.7 does not require to be filled out to create an event, but is useful for letting other users know extra details shown on the events page. Once the required fields have been filled out, find the submit button labelled 4 in Fig. 11.7 and press it, any errors of missing fields will be shown below it.

When the submit button has been pressed, the event will be created and you will be redirected back to the Admin page. You will be able to see the event you have created in the manage events table (shown in Fig. 11 label 13) and the events page as shown in Fig. 8 previously on how to view events.

Error messages

Here is a table of possible error messages and locations you will find them at when using the page and what to do when you experience them.

#	Error message	Location	Reason	Fix
1	"Incorrect Username"	Main login page (Fig. 2)	Username doesn't match any on record	Put in a correct registered Username
2	"Incorrect Password"	Main login page (Fig. 2)	Password doesn't match with Username	Put in the correct password for Username entered
3	"Username missing"	Main login page (Fig. 2)	Username field is empty	Fill out username field
4	"Password missing"	Main login page (Fig. 2)	Password field is empty	Fill out password field
5	"Username field is empty"	Registration page (Fig. 3)	Username field is empty	Fill out username field
6	"Password field is empty"	Registration page (Fig. 3)	Password field is empty	Fill out password field
7	"First Name field is empty"	Registration page (Fig. 3)	First name field is empty	Fill out First name field
8	"Last field name is empty"	Registration page (Fig. 3)	Last name field is empty	Fill out Last name field
9	"No valid RegKey was found during registration, you have been registered without permissions."	Registration page (Fig. 3)	No valid RegKey was entered	Fill out with a valid RegKey
10	"One or more files had un-allowed extensions during upload, please retry."	Upload page (Fig. 5)	A file uploaded had an un-allowed extension	Only upload files that are allowed (JPG, JPEG, GIF, PNG)
11	"Something went wrong when uploading, please retry!"	Upload page (Fig. 5)	Critical error – Something went wrong during upload from client to server	Retry uploading, If still broken contact system admin for support.
12	"That is not your file to delete!"	Manage page –	Tried to delete a	Don't try to delete

		During deletion of file (Fig. 6/Fig 6.2)	photo which ID isn't tied to you (In other words, you don't own it)	files that don't belong to you.
13	"Permission field is empty" / "Permission Level is missing."	Changing permission level page (Fig. 11.1) / Creating new RegKey page (Fig. 11.3)	Permission level field is empty	Fill out permission level
14	"Keyname is missing."	Creating new Regkey page (Fig. 11.3)	Keyname field is empty	Fill out Keyname field
15	"Expiry date is missing."	Creating new Regkey page (Fig. 11.3)	Expiry date field is empty	Fill out expiry date field
16	"That key name is already taken, please enter a new one!"	Creating new Regkey page (Fig. 11.3)	Keyname entered already exists	Enter a different Keyname that does not exist.
17	"Post title is missing."	Creating a new Post page (Fig. 11.5)	Post title field is empty	Fill out post title field
18	"Post content is missing."	Creating a new Post page (Fig. 11.5)	Post content field is empty	Fill out post title field
19	"Event name is missing."	Creating a new Event page (Fig. 11.7)	Event name field is empty	Fill out event name field
20	"Event date is missing."	Creating a new Event page (Fig. 11.7)	Event date field is empty	Fill out date name field
21	"You are not logged in, you will now be redirected to the login page."	Any page that requires you to be logged in	Not logged in to the system	Log into the system
22	"You do not have a high enough permission level to access this page. This page requires a minimum permission level of 5"	Any page that is restricted to admins only	Do not have a high enough Permission Level	Become an admin If appropriate, or do not access those pages.

Appraisal

Comparison of Project Performance Against Objectives

The following table compares the performance of the completely system to the objectives created in the Analysis section of this document (Objectives for the Proposed System). This is used to assess to what extent the user's needs have been met.

From the list of objectives, I have only included the main 12 objectives in the table below but I have considered each sub point of each objective (as seen in Analysis) to further help my assess performance.

#	Original Objective	Completed System	Objective status
1	The system must be able to store photos uploaded by photographers	This system does allow to for photographers to upload photos, most of all the sub objectives have been met. The system stores the photos securely, it pulls data from them when uploading then stores this in a database, and it also allows the user to upload an almost limitless amount of files at once due to the algorithm used when parsing each photo into the system and lastly, the system does rename any duplicate files being uploaded. The only objective only partially met is the security permissions for certain groups. The permissions system only stops unregistered people from viewing private gallery photos, other than that any registered user can view the gallery page.	Objective met
2	The system will need a login system where each user is given login details to access the system	Users can login to the system with details once they have registered. All sub objectives have been met, as logins are restricted by a permissions level system which only allows users to access certain pages when they're permissions level is higher than normal. The logins are also encrypted, all passwords are stored hashed, so no plaintext passwords can be stolen from the database by intruders.	Objective met
3	A registration system will be needed to go with the login system.	The system does have a registration system allowing users to make accounts to the system automatically. They are secure as mentioned in the security of the login system, as all passwords are hashed and stored in a way that they cannot be reversed and stolen. Registrations aren't restricted as wanted in the objectives, but instead	Objective met

		you are registered with a base permission level of 0 which restricts a user's access to protected pages such as administration. To get a higher permission level the user needs to either register with a registration key with a certain permission level, or have an admin manually increase it.	
4	Users will need to be able to manage their account.	The system allows for users to partially manage their accounts, they can only change their passwords. During development I concluded that an avatar system wouldn't be worthwhile feature, which my client agreed with.	Objective partially met
5	A searching function will be required to allow users to search for photos based on information pulled during upload (see 1.a).	This objective has been partly met, you are able to search for a file with parameters being: filename, fileid and tags. You are able to add tags to an image to be searched for but EXIF data was a feature that was too complicated to implement and impractical as EXIF is only supported by certain image types, so not all data would be pulled from each photo being uploaded.	Objectively mostly met
6	As mentioned in 1.c and 2.b a permissions system will be required to underpin the system.	A permissions system does exist, but there is no implementation to protect *certain* files. Permission levels are tied to each account, they do restrict which pages users can access, like certain galleries. Additionally, all user management or admin panel access is restricted to any user who does not have a permission level of 5 and above. Default permissions level is 0.	Objectively mostly met
7	Event calendar to allow photographers to post events for other users to see where they are needed.	The system has an events calendar which admins can put events up with date/time, event name and event notes. The only sub objective not met for this was a system where users can tick If they are able to attend the event.	Objective met
8	Image processing should be implemented into photo management	This objective was not met and implanted, due to the time restraints and the complexity of image processing. The processing would either require long functions that I was unable to create or use of many libraries. This being said, there was a sub objective met, when uploading the images are resized for thumbnails but this isn't sufficient enough to class as a partially met objective.	Objective not met
9	User should be able to manage their photos they have uploaded.	Users are able to upload and then delete/preview/fetch URL's for their images in a management page in the system. They are unable to make their photos private, but they are protected by a permissions level from any unregistered users. Additionally, the last sub objective was partially met, as each users images automatically go into their own user album/gallery.	Objective partially met
10	The solution will need a help/FAQ section, popup(s) or page to help the user use the system.	I decided that a help/FAQ section was not needed due to the creation of a very detailed user manual with written in help/tips. There are some pop ups throughout the system to help the user If they are doing anything wrong, such as entering wrong field data.	Objective not met (Further development deemed it unnecessary)
11	The system should	I have included a statistics page which allows system	Objective

	have features to ease maintenance.	administrators view the % each users is taking up in terms of file space. I decided to not implement a system for backing up files, as the method I wanted to use would only work on a certain server operating system therefore another system would have to be made for a different server OS (so basically compatibility issues).	partially met

Potential for Future Developments/Extensions

For future developments, currently the permissions system is very basic and needs some more features added to it. It needs to have permissions level applied to the images, with only being able to see the permission level of images that are below yours. So, you can have public users that can only view images at their permission level instead of photographer's images which would have a higher permission level. Additionally, having some administrative features given to other permission group's not just admins would be another extension of the permissions system. This means you could have photographers with access to the add posts page and add events page, so they could go create them themselves instead of having to ask and admin to do it (As If they are given the permission level to access these features, on the current system they would be able to manage other users and access admin only features).

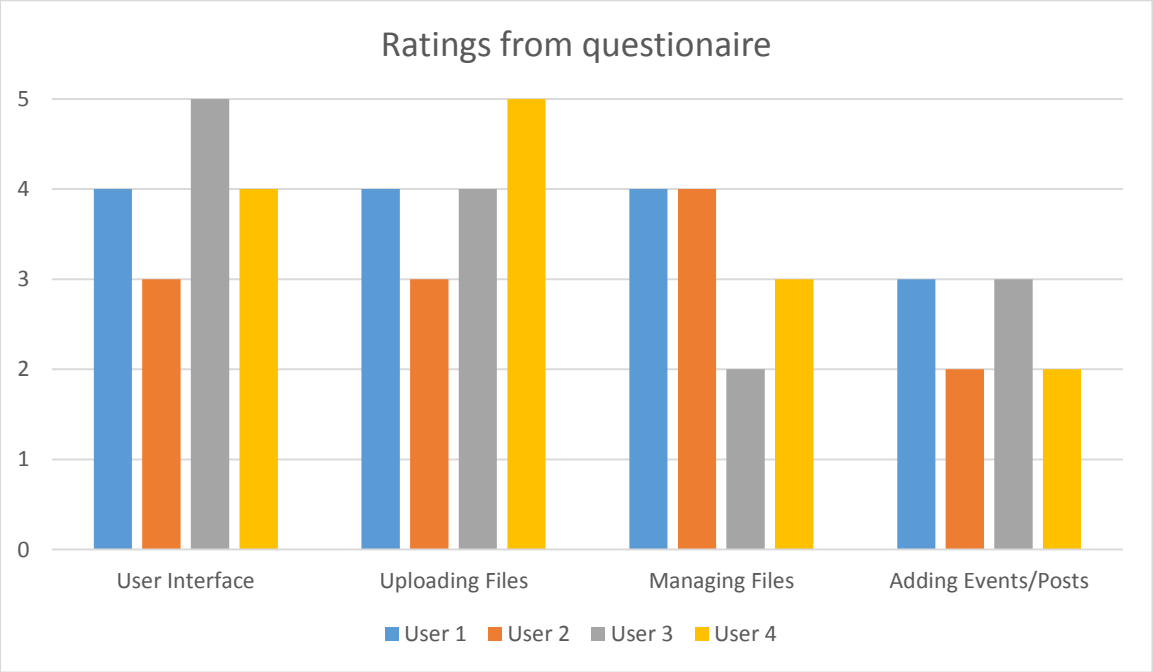
Another development to implement would be a functional albums system where users can specifically create an album and assign images to it. Currently, you can only view albums that are generated for each user, so they only have all the images from each users contained inside them not specifically chosen to have certain content.

Branching off from the albums, I would like to have had a system where you can assign an album to an event, so when the events finish you could view the photos taken at that given event. I would need to first make the albums system more complex as mentioned in the last paragraph, but also I would need to make events stay in the database even after they have expired then display them in a separate table of expired events.

Looking back on my final codes listing, I found that I had a lot of repeated code. I found this very inefficient, if I had time to redevelop the project again I would approach the programming differently. By this I would probably code it using more functions, as a lot of algorithms I used repeatedly on each page I could've just coded into functions and just included a file of classes that could be used in each page.

Analysis of User Feedback

I gave the user manual, feedback questionnaires and access to the system to 4 users from the photography group via an online form. To enable me to get an assessment of how users rated the system after their initial use, I included a rating section for users to rate the user interface, uploading files, managing files and adding events/posts, out of 5. I also asked them for any general feedback/improvements to any aspect of the system which has been summarised below the results of the rating questionnaire.

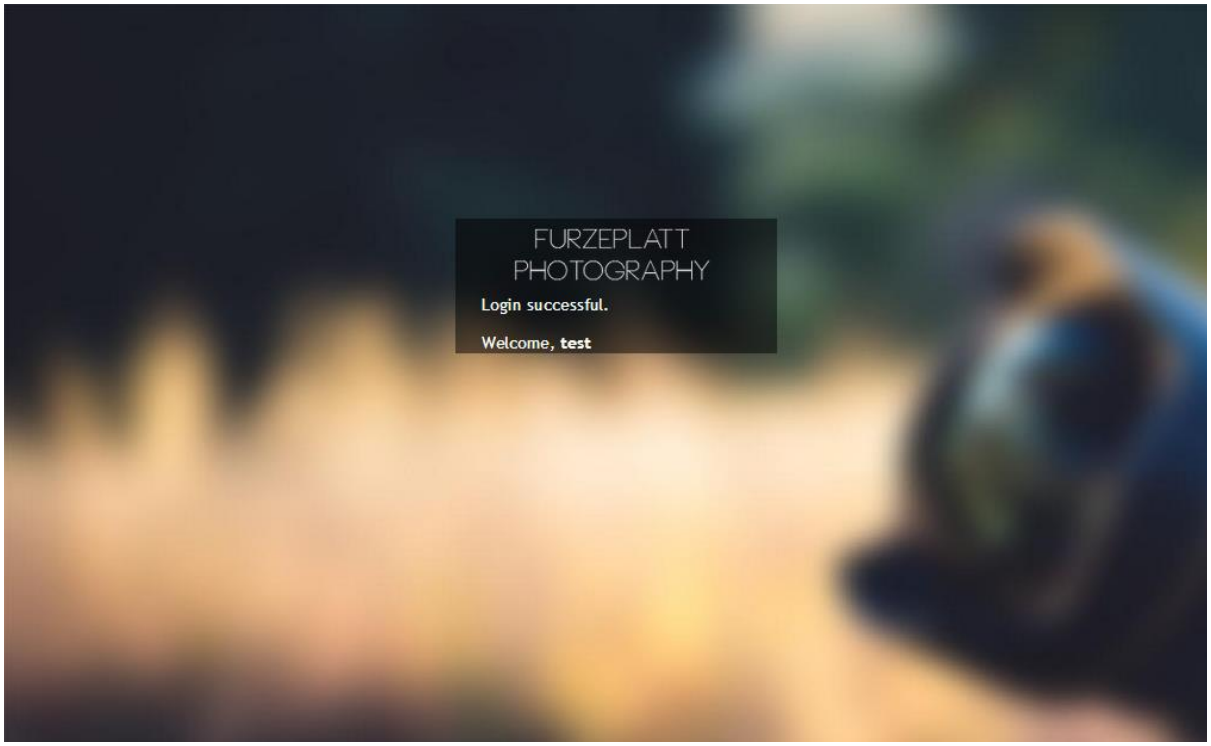


User 1	User 2	User 3	User 4

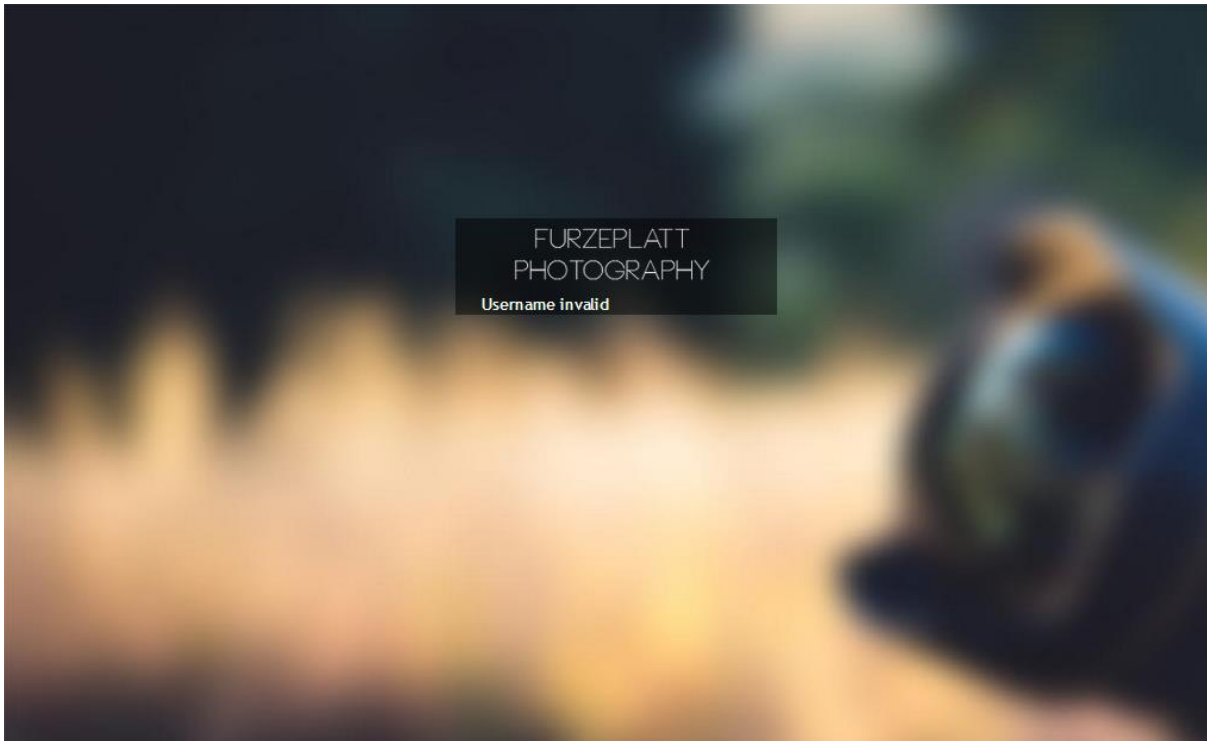
• UI is good and easy to navigate. • Adding posts and events is a bit clunky/not a smooth process. • Upload system could do with some improvements, such as a progress bar for uploads.	• Uploading files didn't show you any progress to your current uploads. • User interface was average, seemed a bit dark. • Adding events/posts needs improvements, very basic.	• User interface was brilliant, looked very professional and transition between pages was a breeze. • Managing files was lacking, would be good to have a feature that allowed you to delete multiple files at a time. • Events/posts was hard to understand current existing posts in the manage events table.	• Overall, system was good, easy to navigate • When managing files, the tagging system feels like it needs more polishing as it was a slow process to add tags to each individual file – a feature should be added to add tags to multiple files, for example tagging a bunch of images from an event. Maybe even add the ability to tag files on upload.

Testing Outcome Screenshots

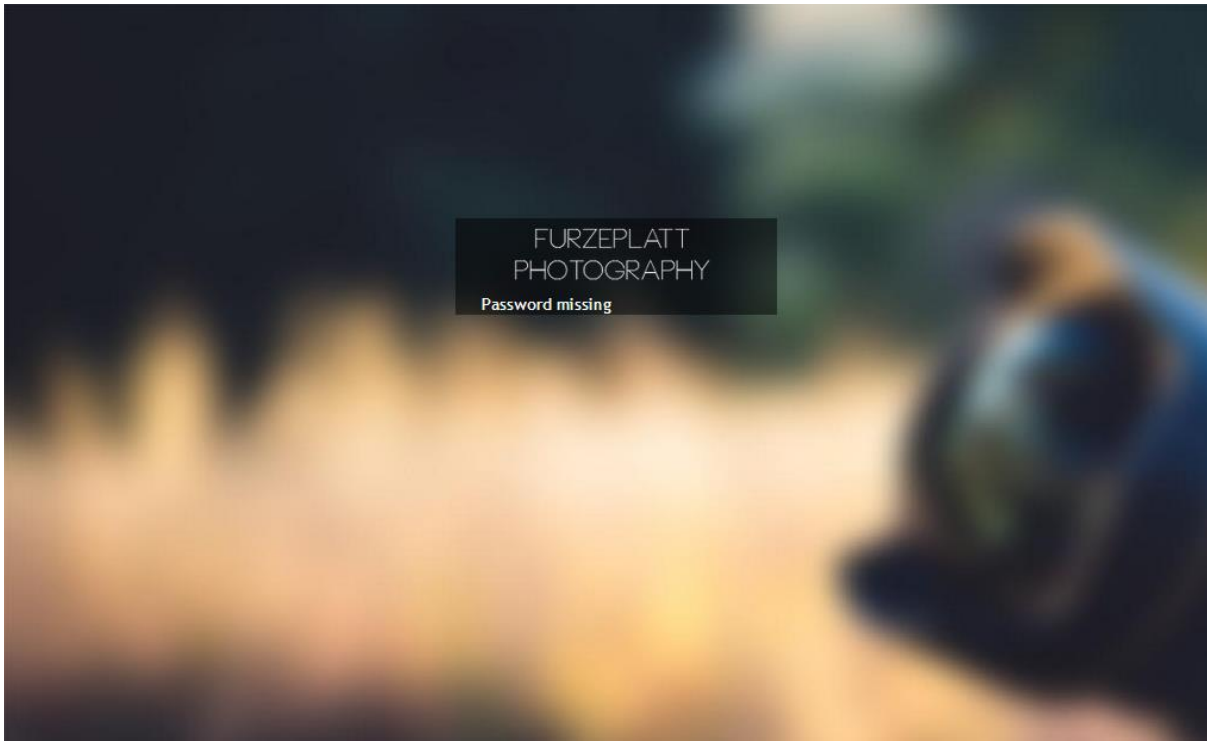
Test – 1



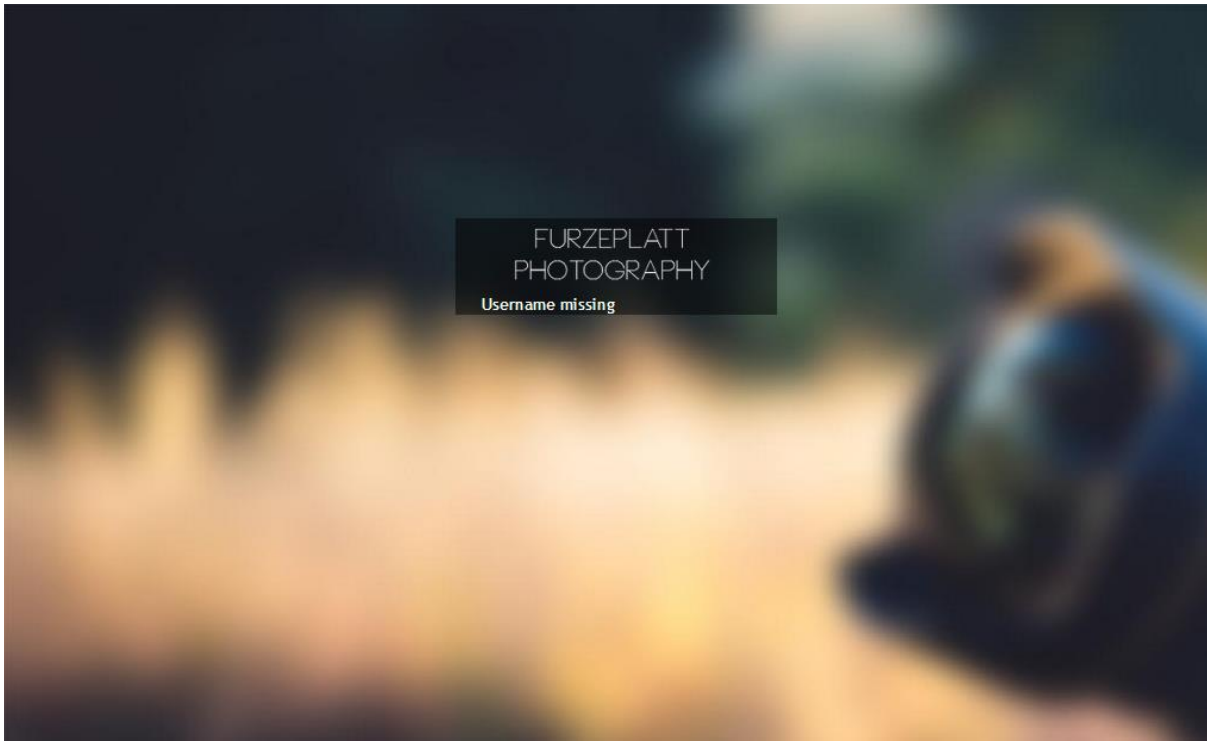
Test – 2



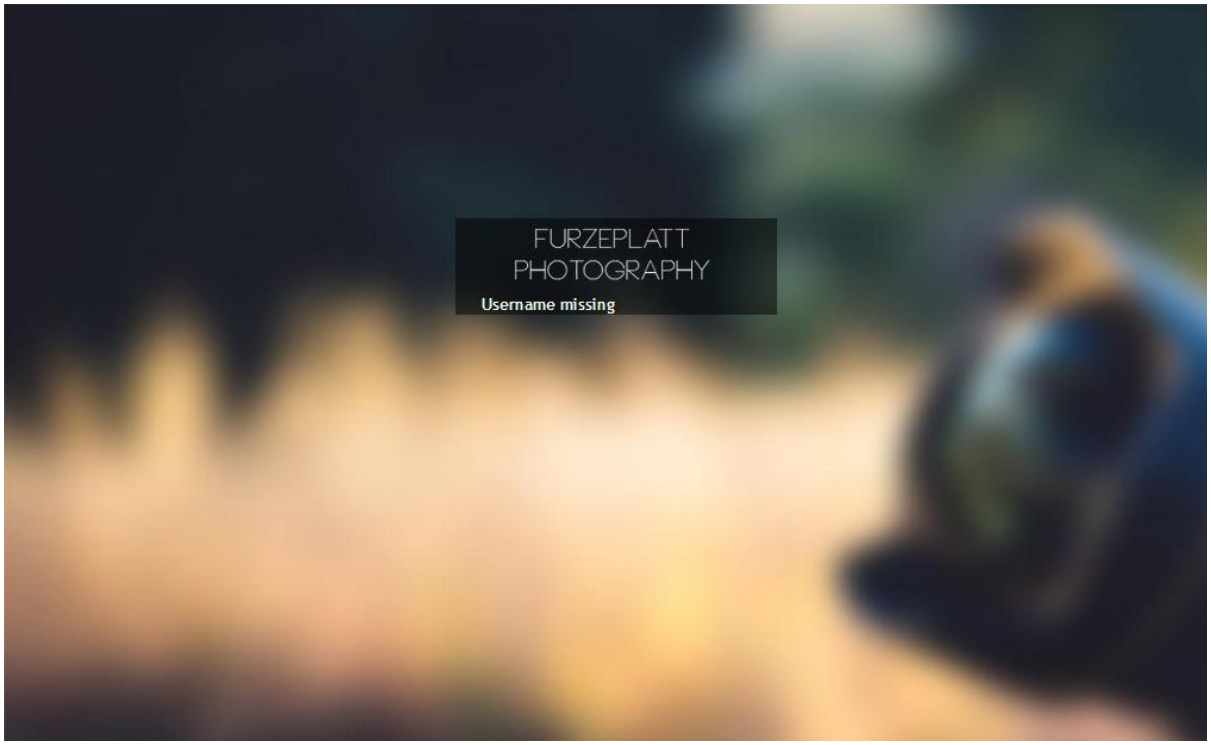
Test – 3a



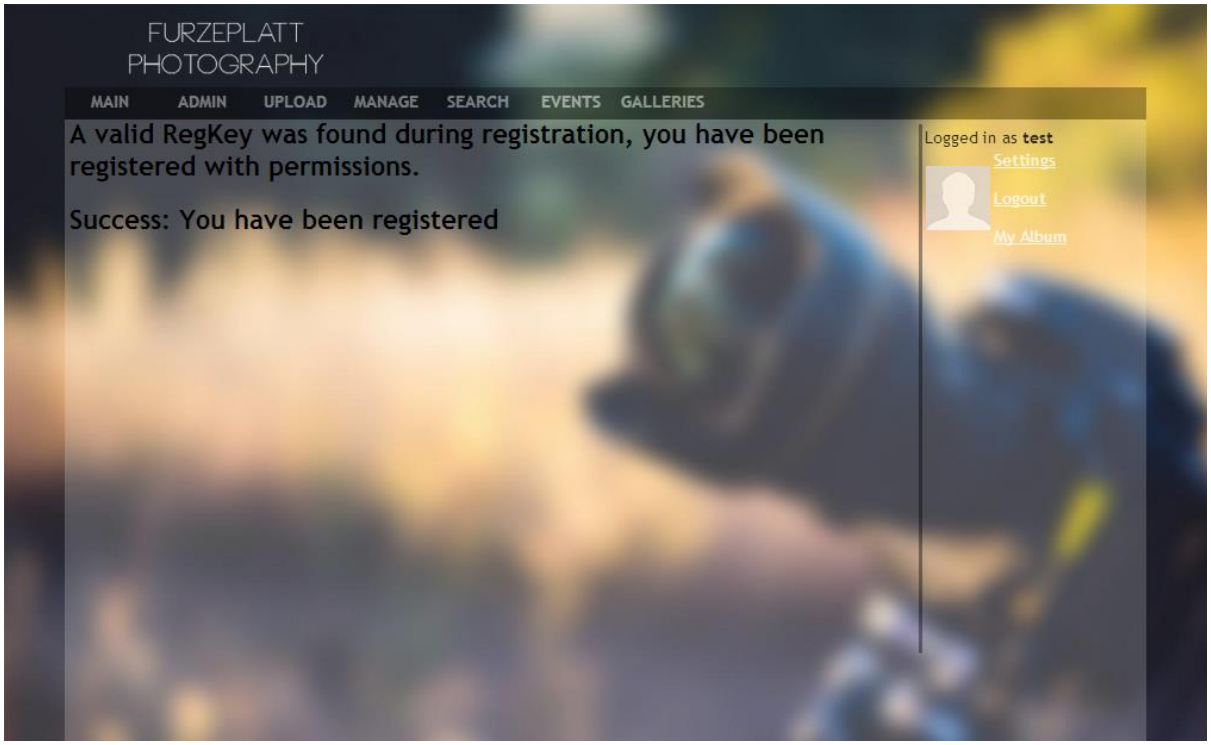
Test – 3b


FURZEPLATT
PHOTOGRAPHY
Username missing

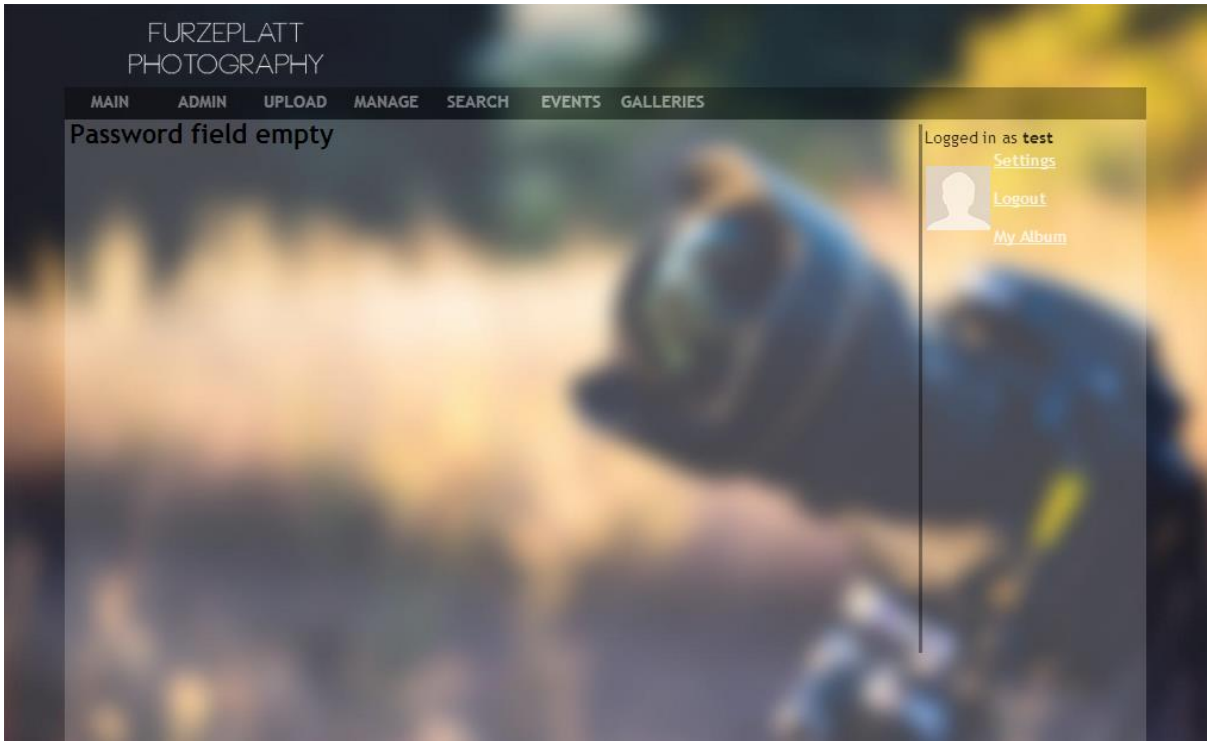
Test – 4


FURZEPLATT
PHOTOGRAPHY
Username missing

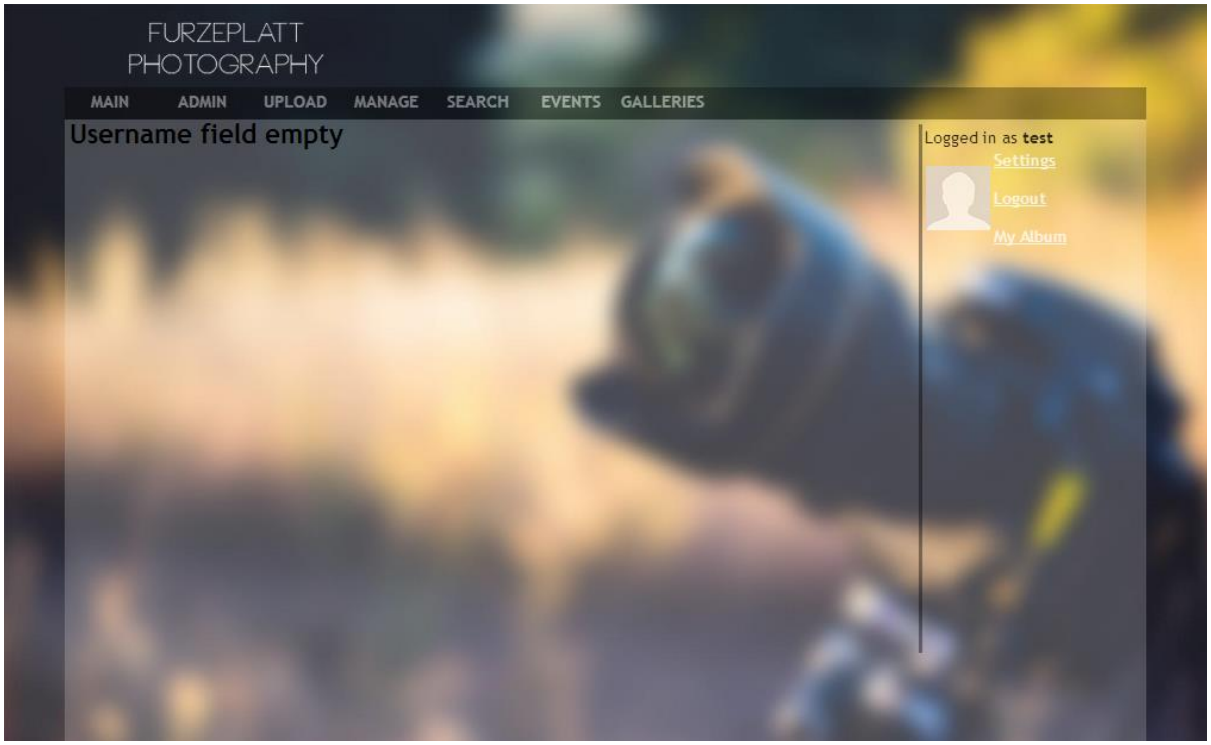
Test – 5



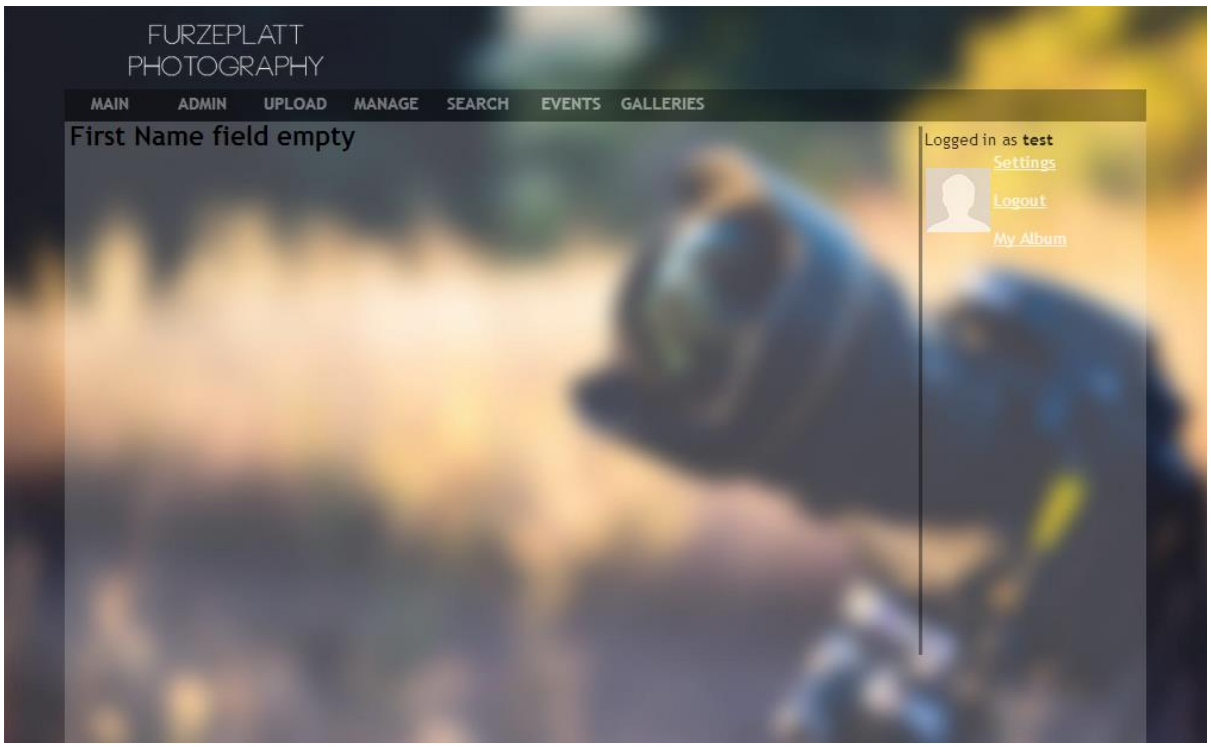
Test – 6a



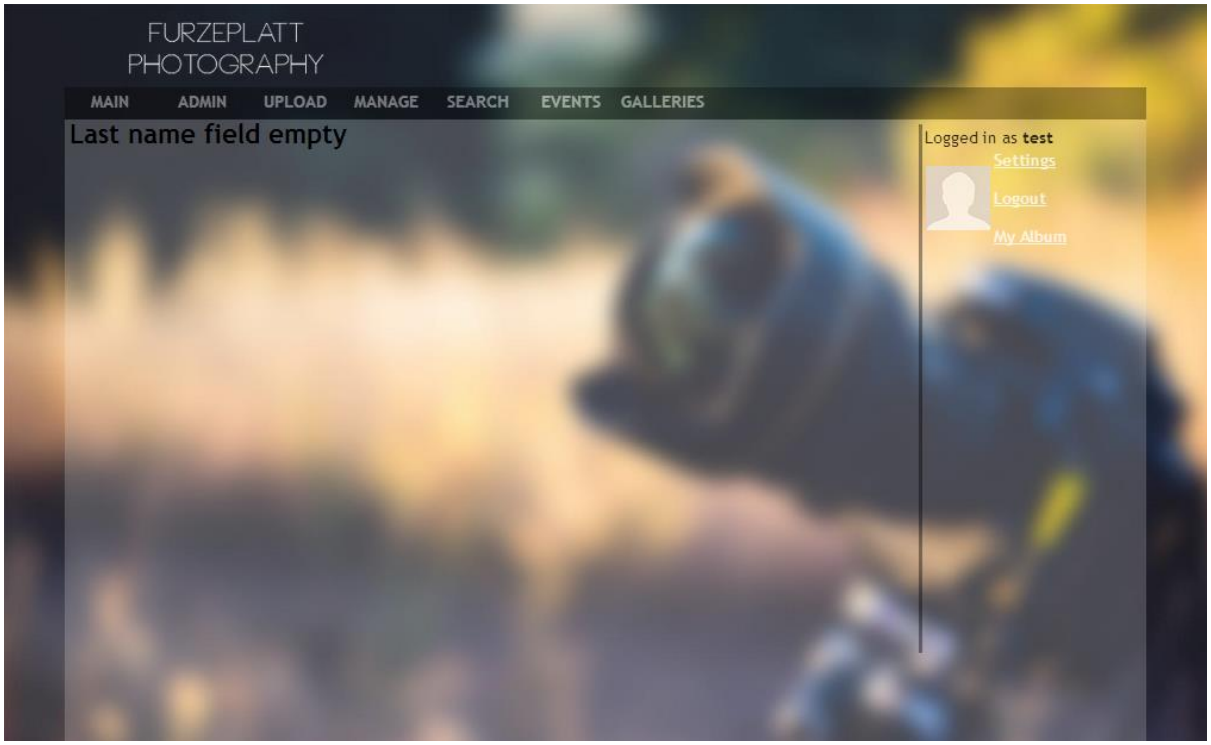
Test – 6b



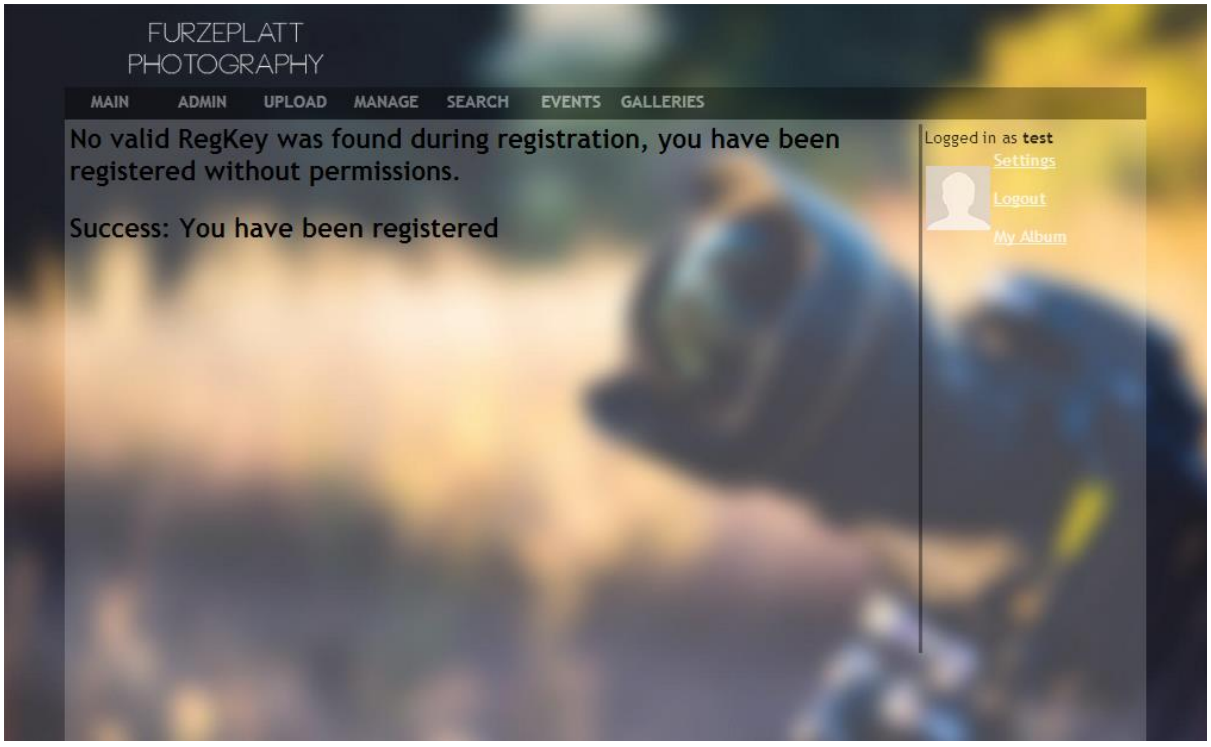
Test – 6c



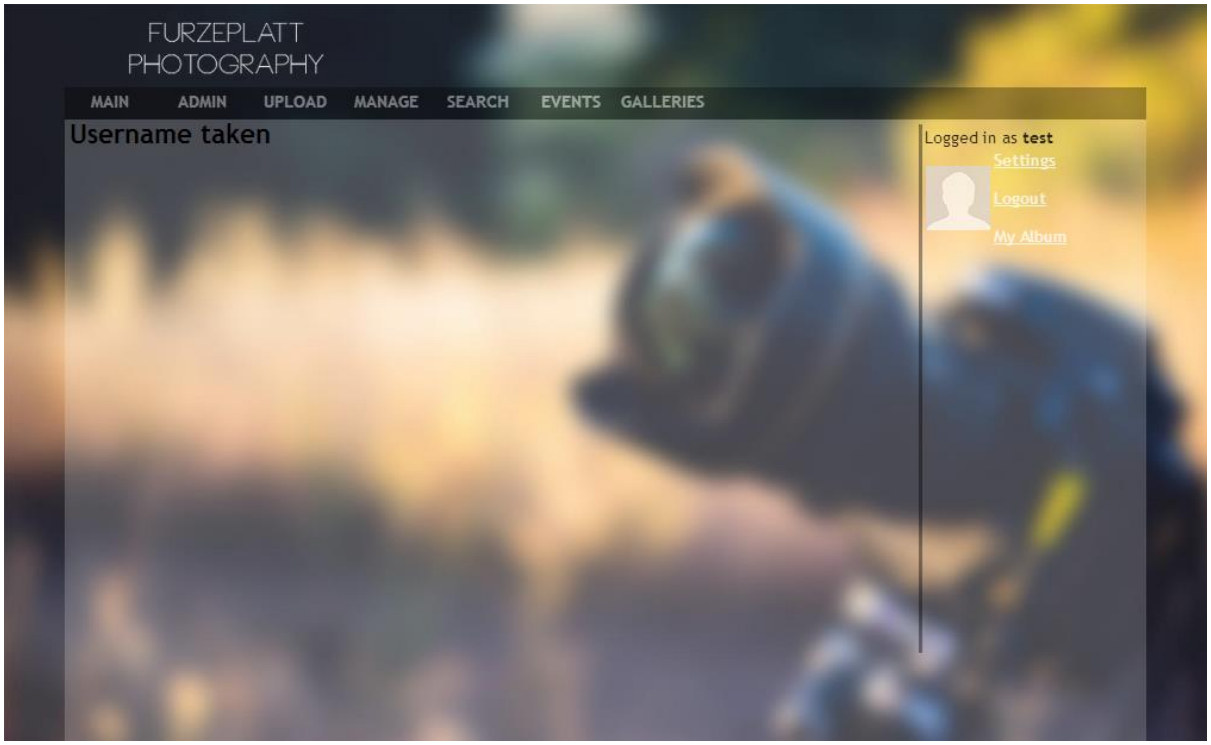
Test – 6d


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
Last name field empty
Logged in as test
Settings
Logout
My Album

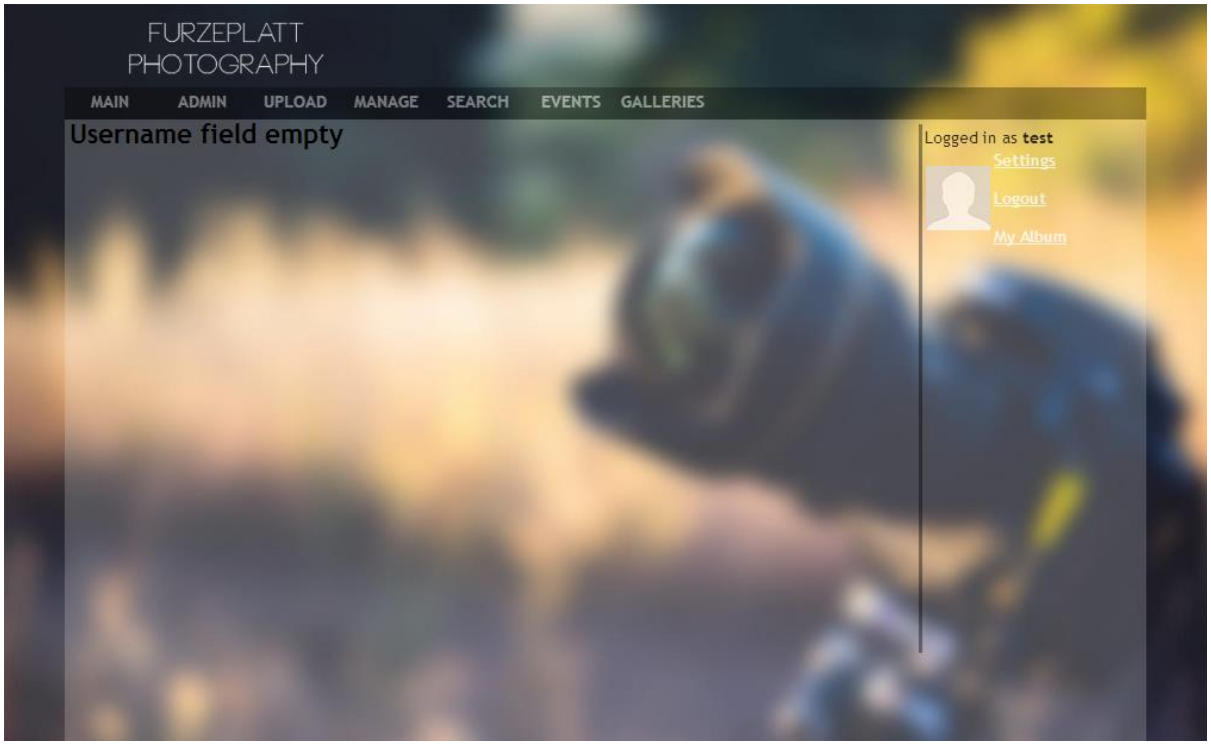
Test – 7


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
No valid RegKey was found during registration, you have been registered without permissions.
Success: You have been registered
Logged in as test
Settings
Logout
My Album

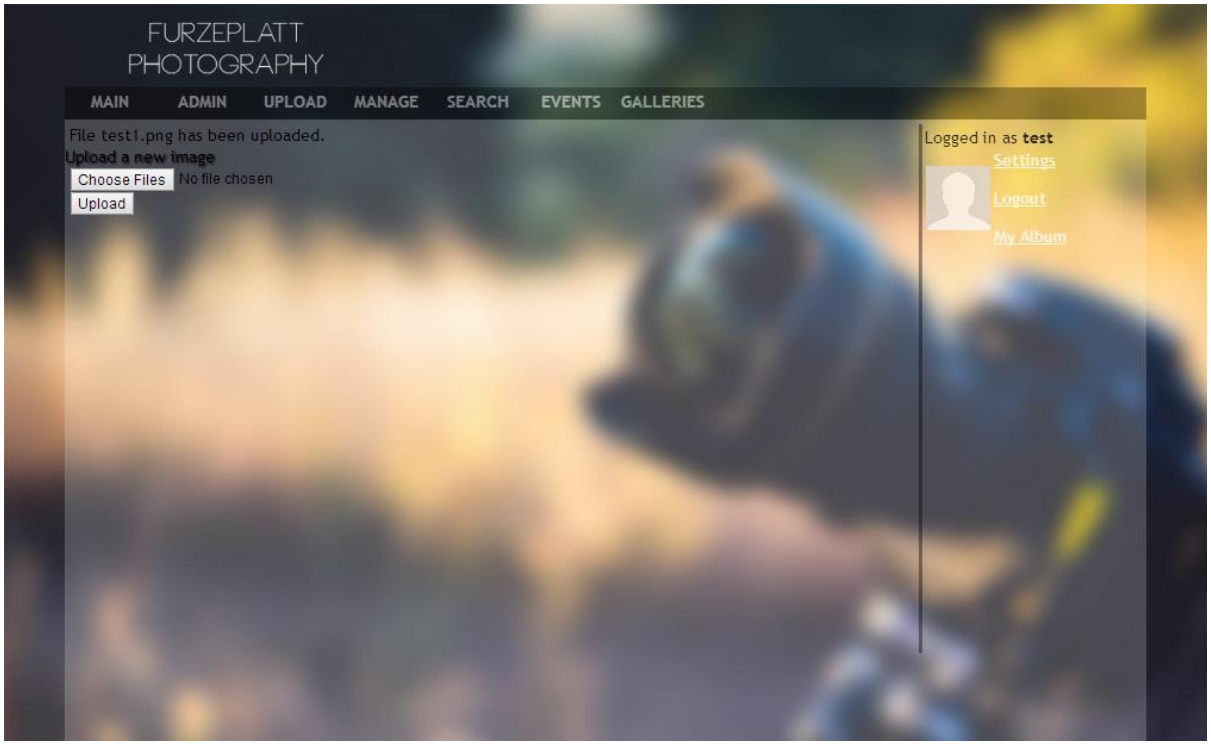
Test – 8



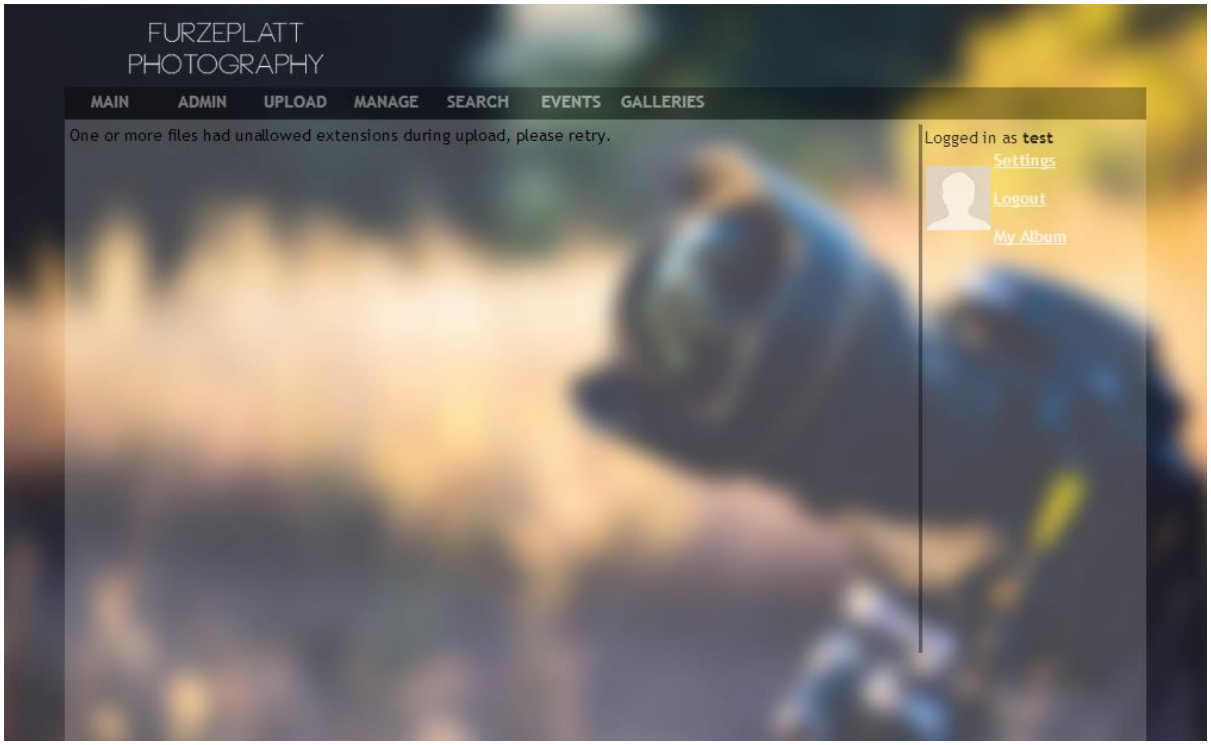
Test – 9



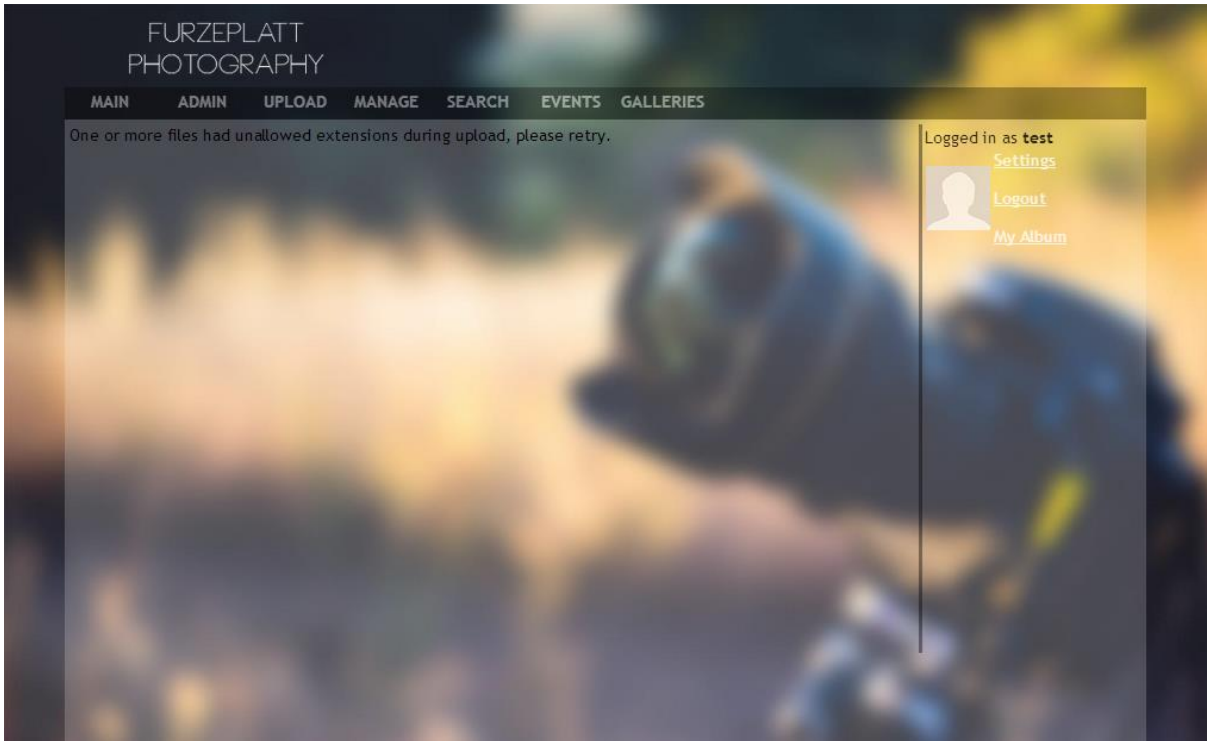
Test – 10



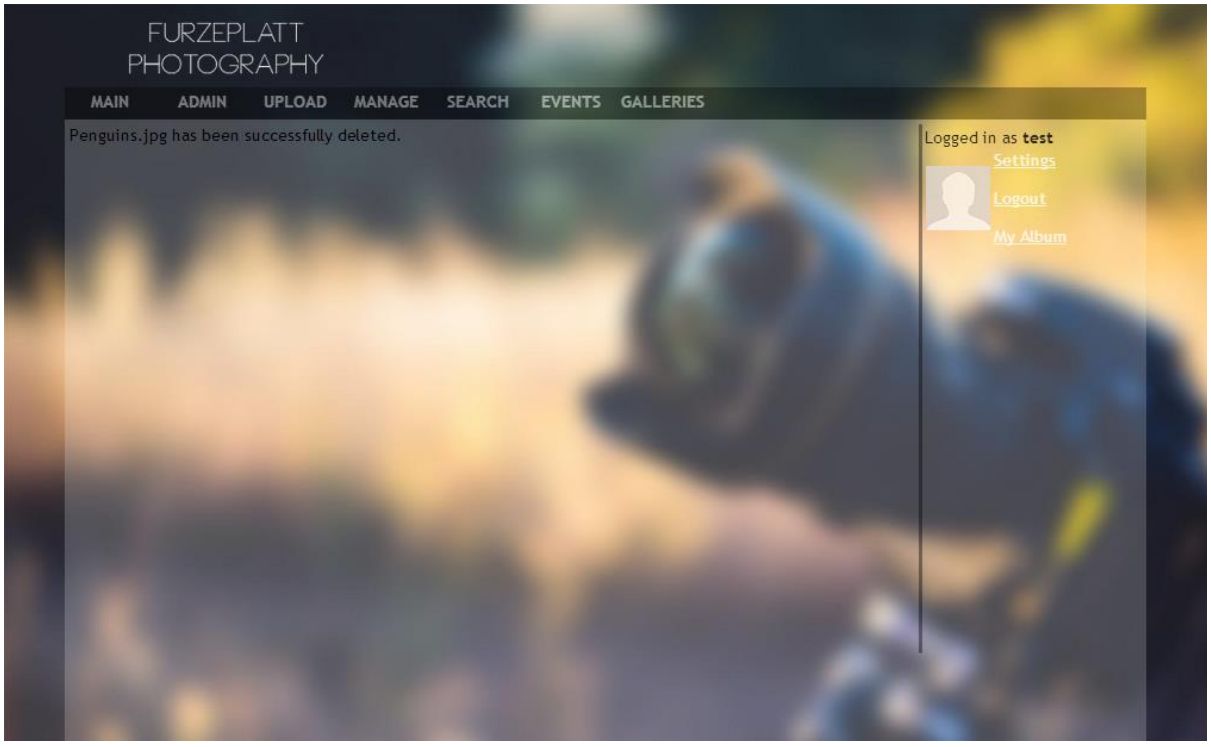
Test – 11



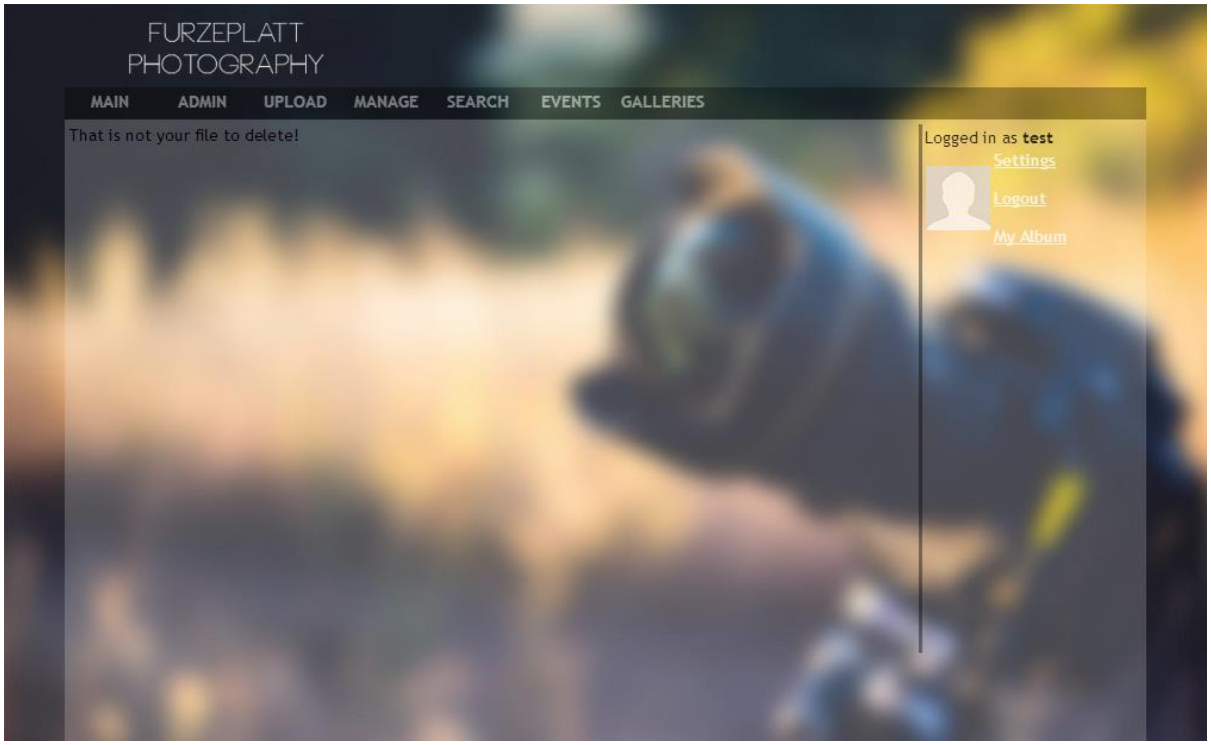
Test – 12



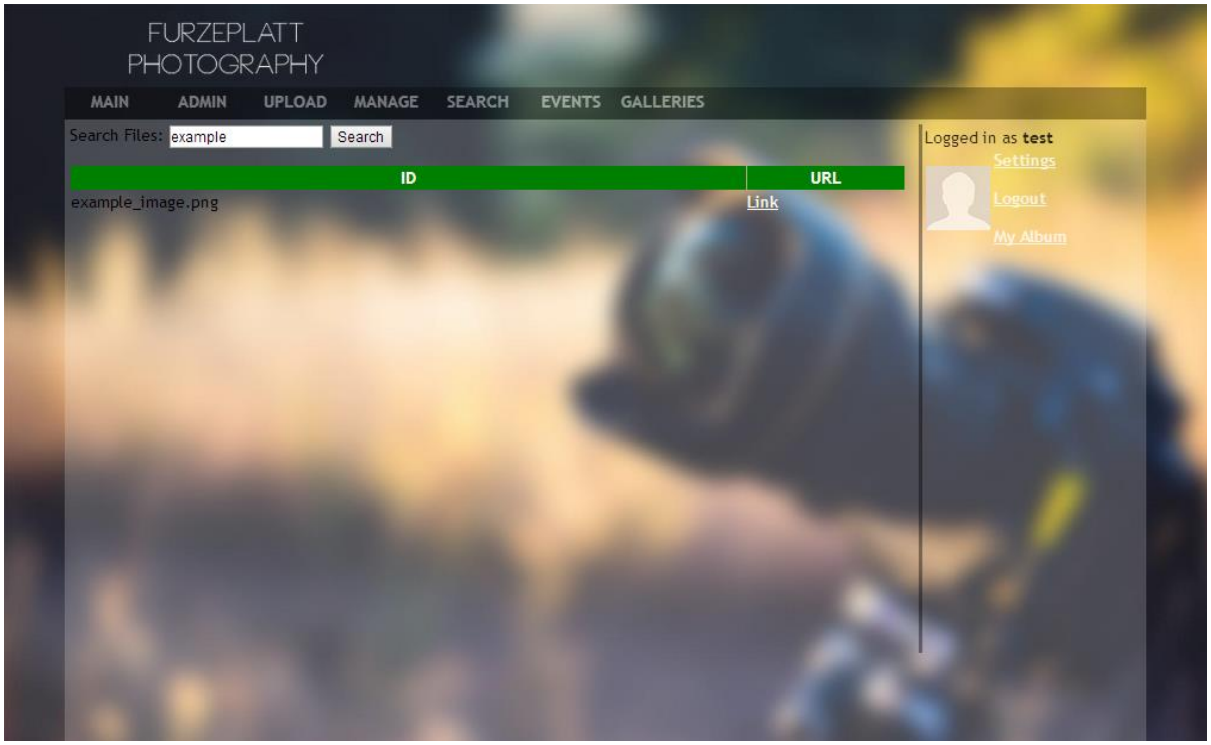
Test – 13



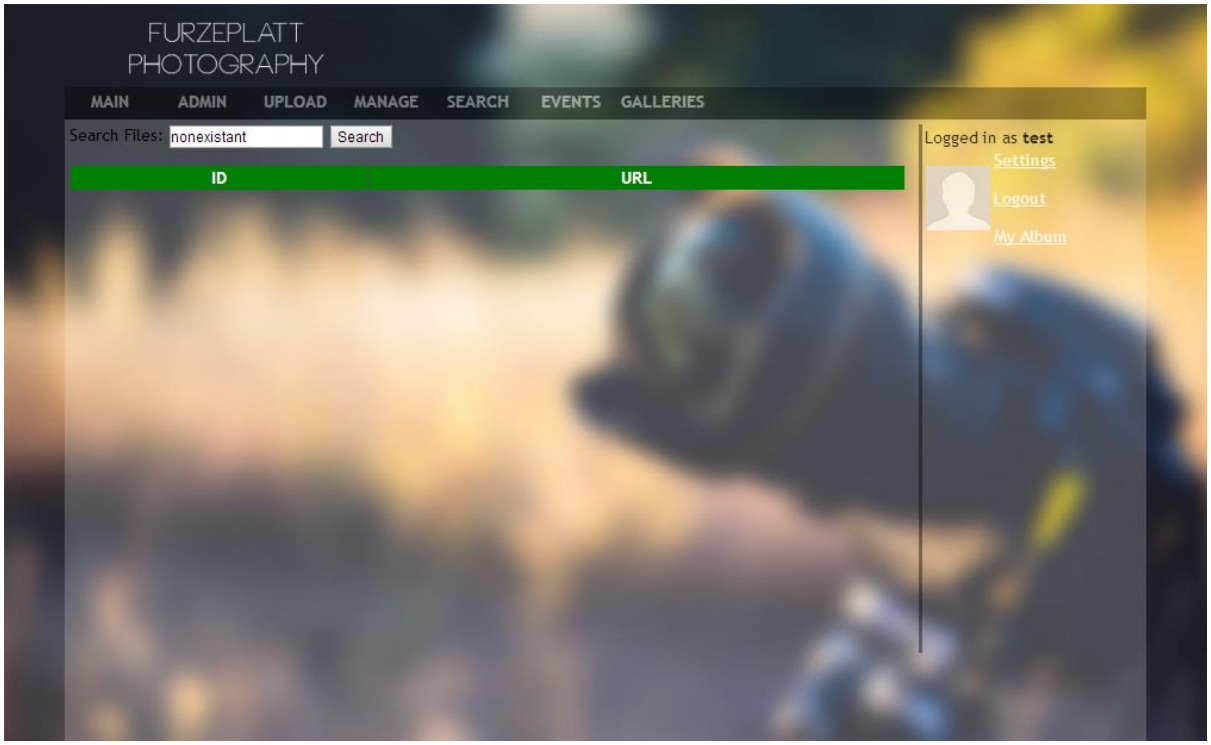
Test – 14



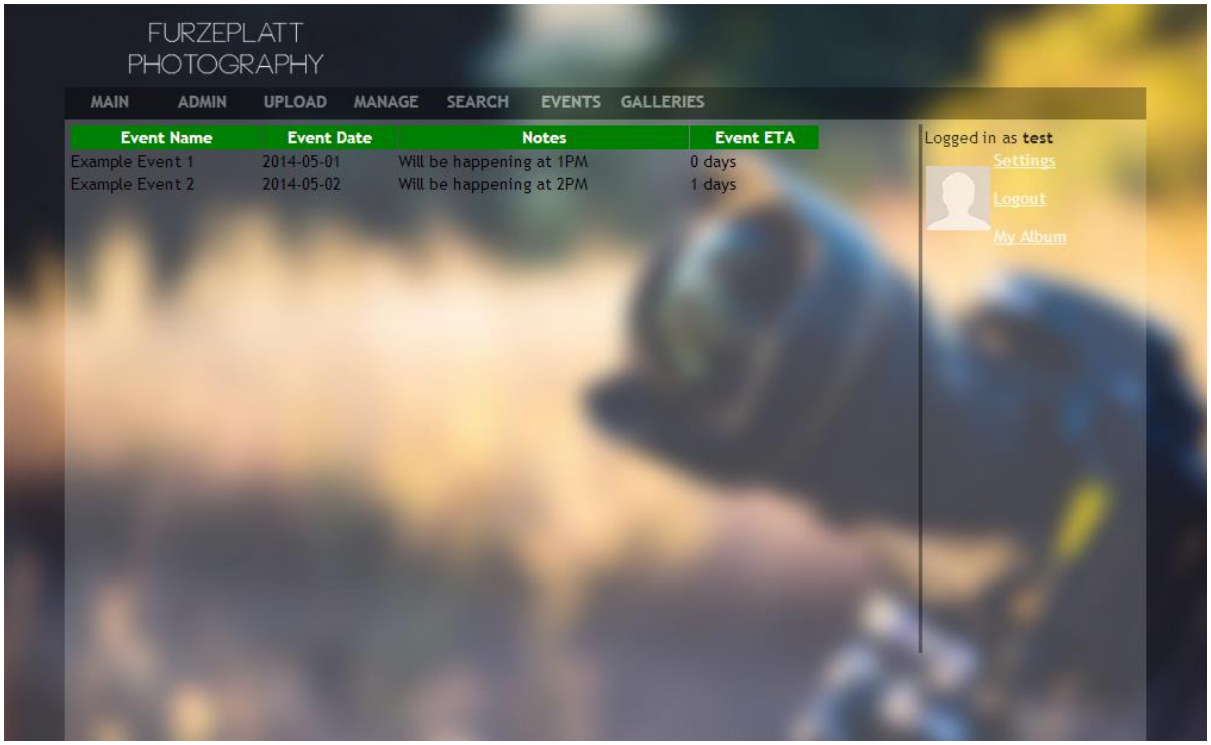
Test – 15



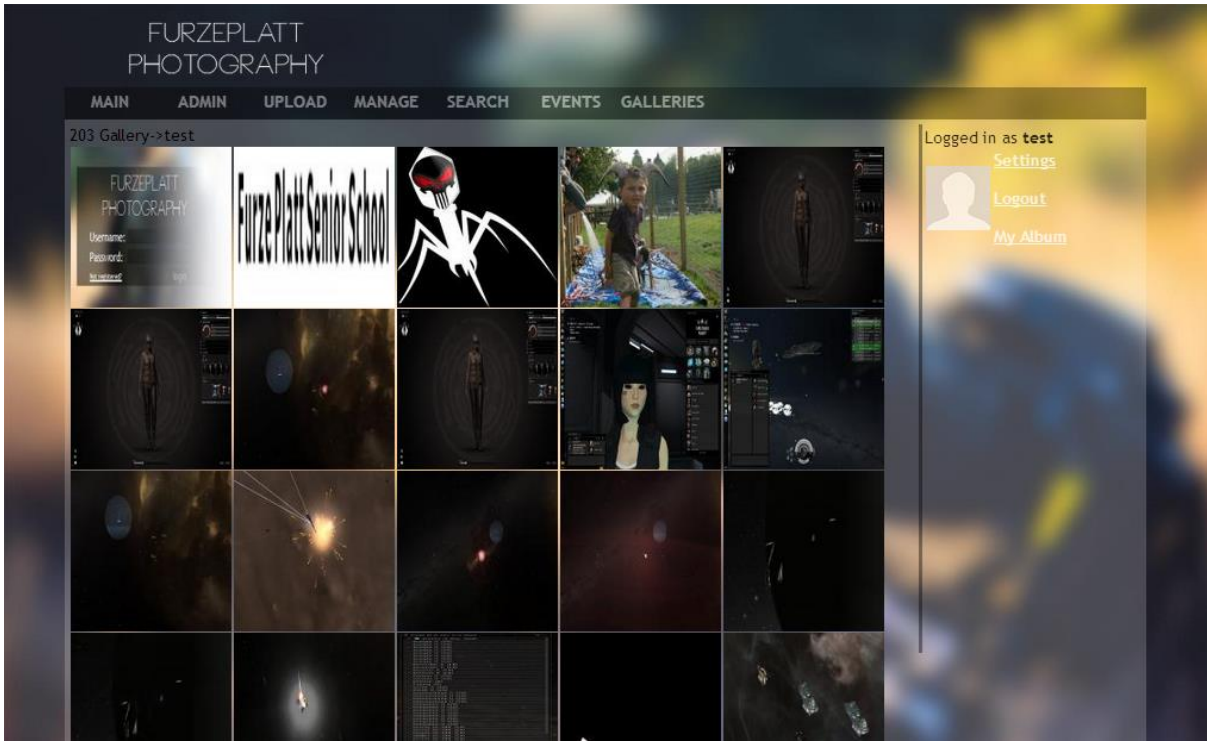
Test – 16



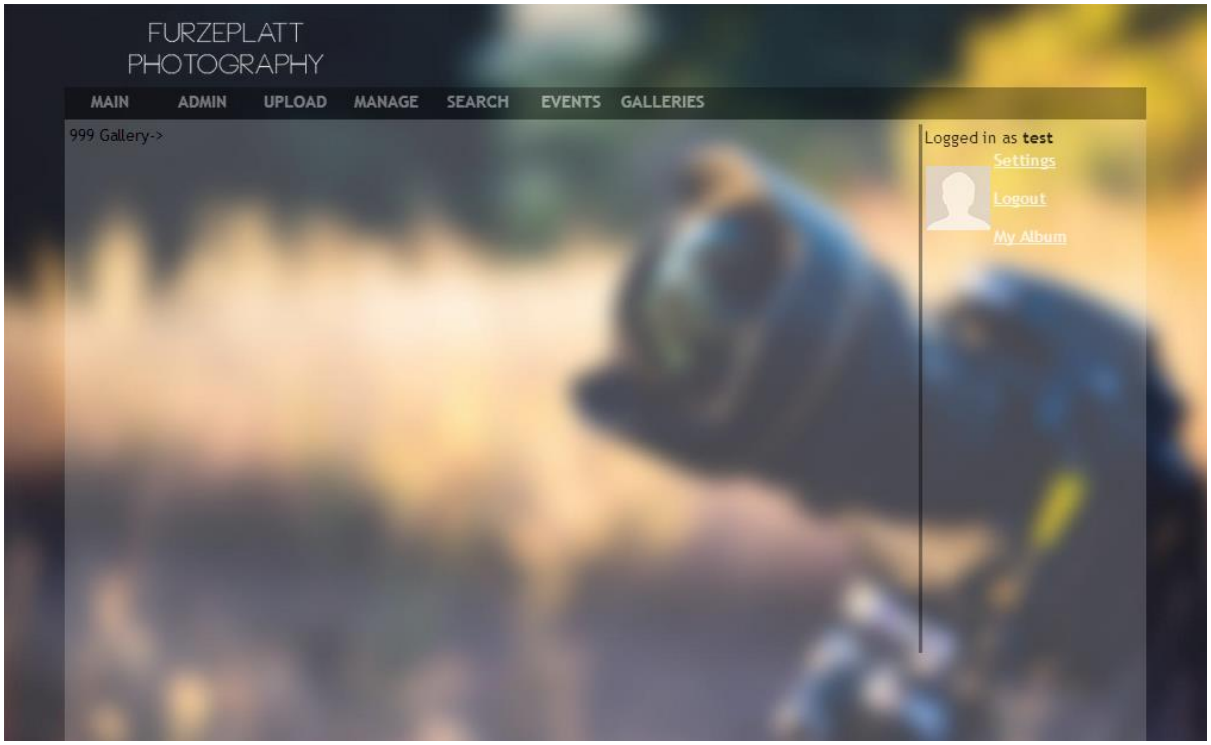
Test -17



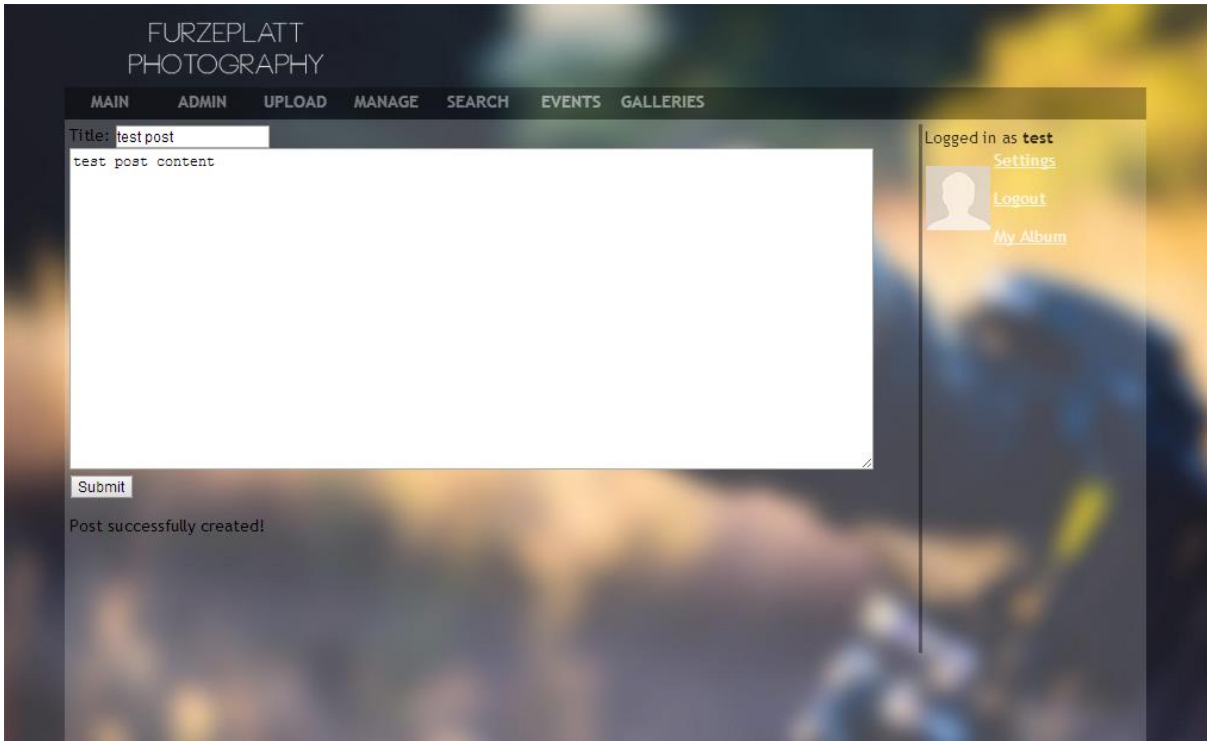
Test – 18



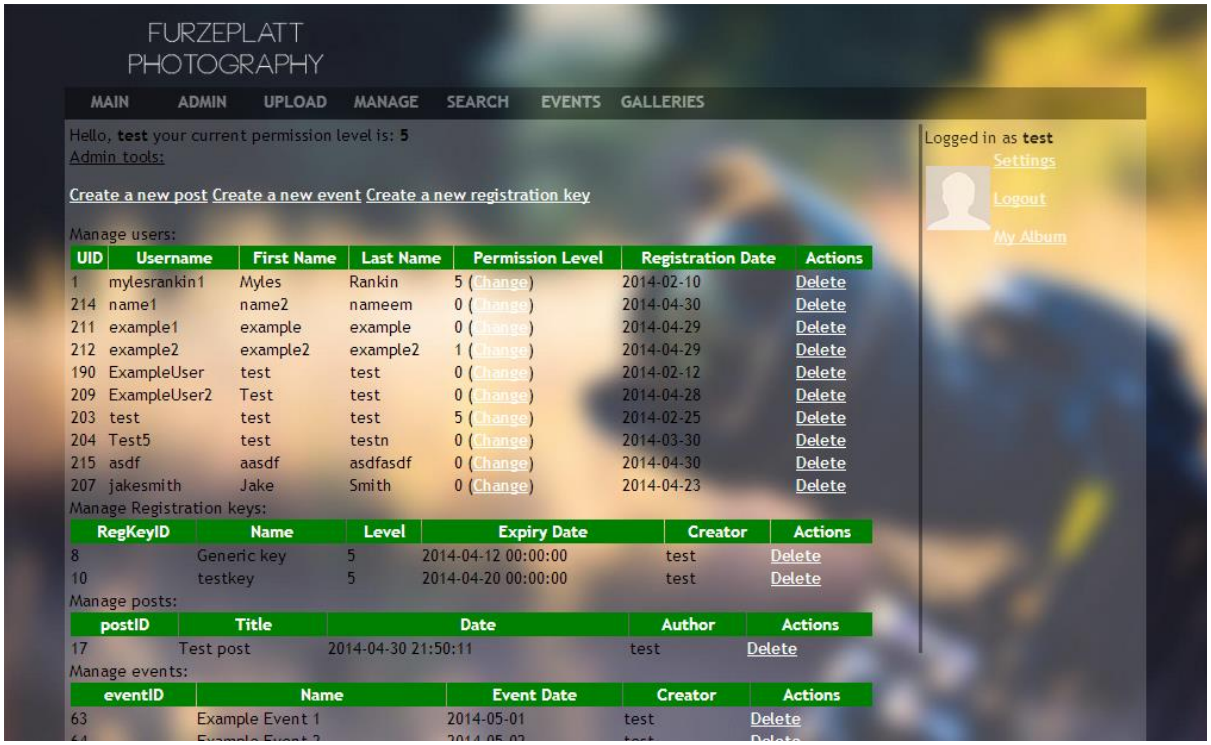
Test – 19



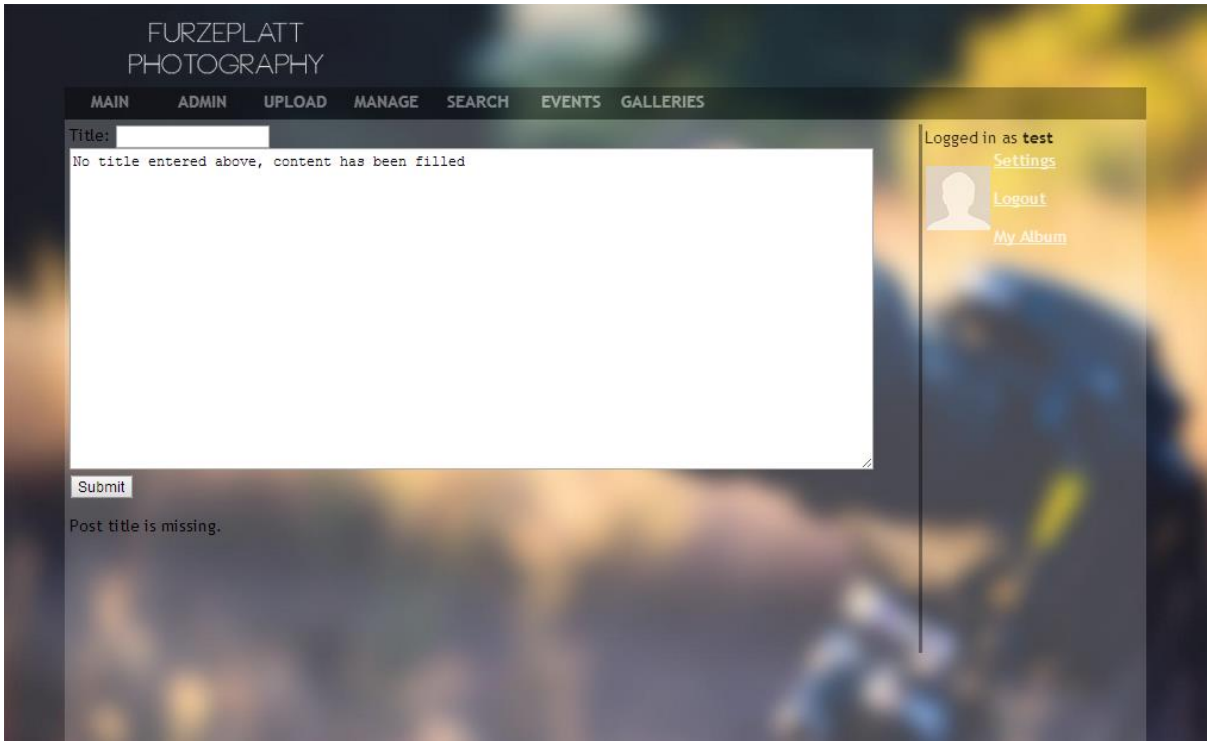
Test – 20a



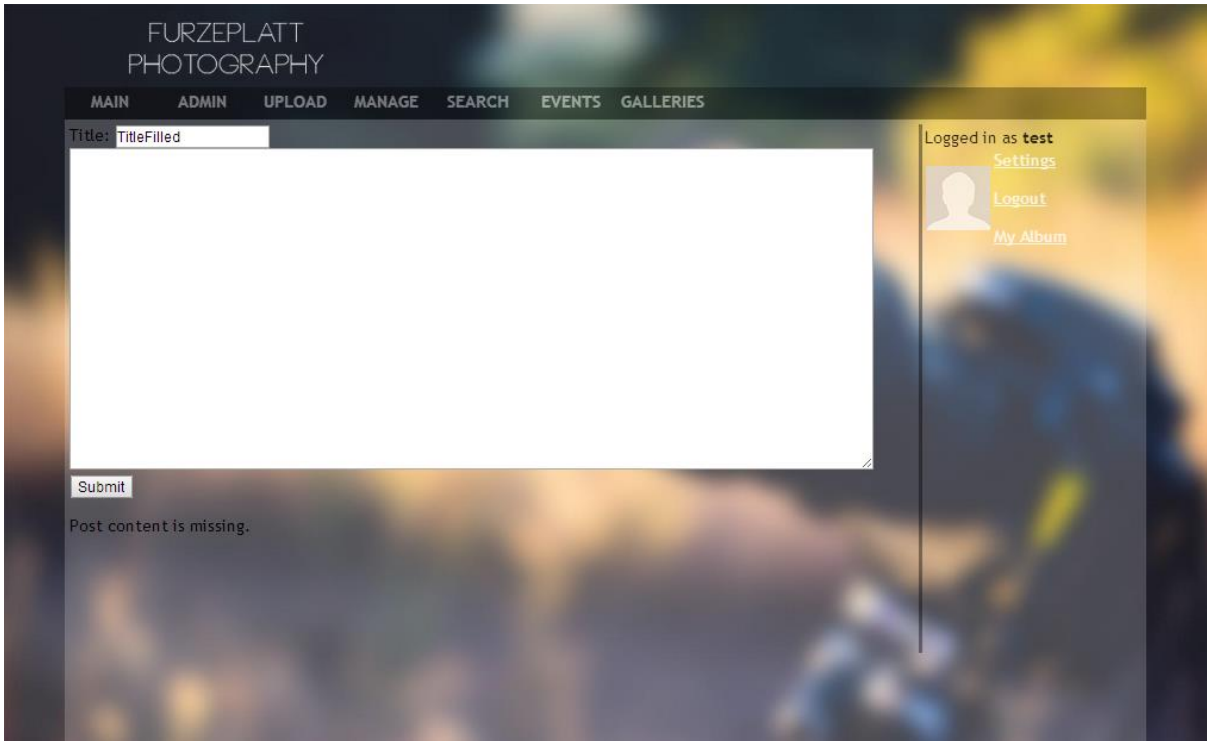
Test – 20b



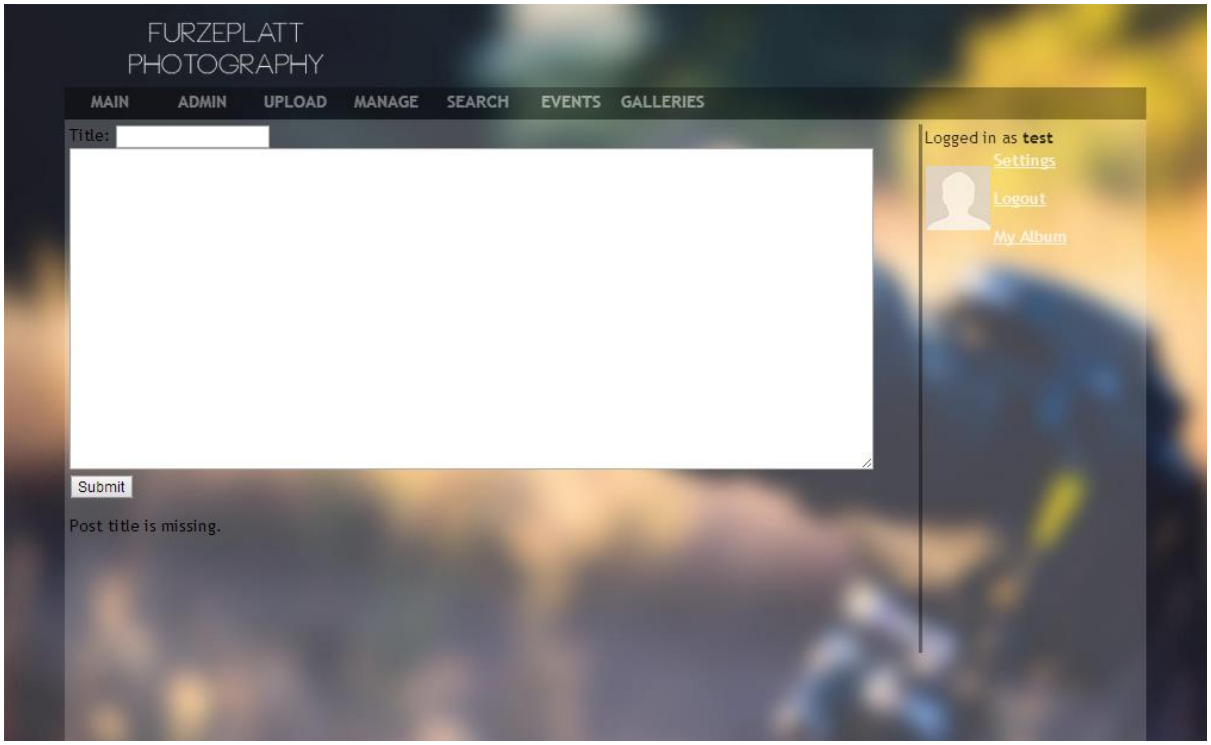
Test – 21a



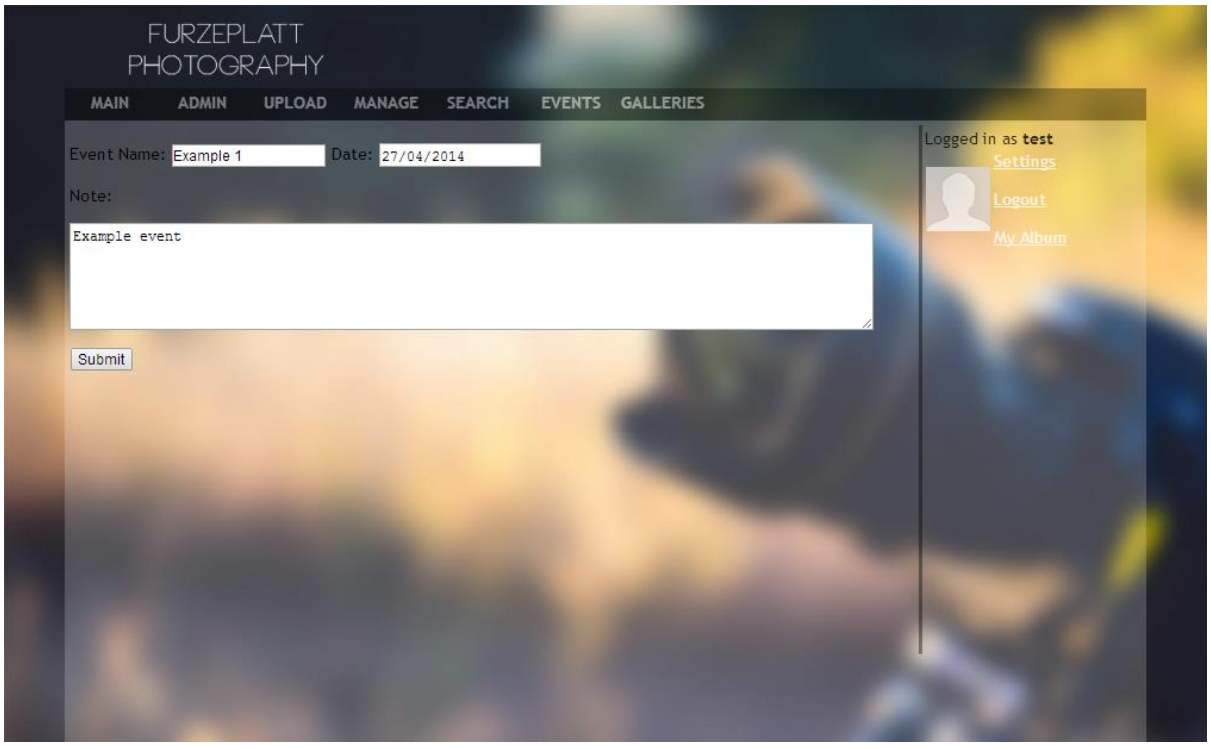
Test – 21b



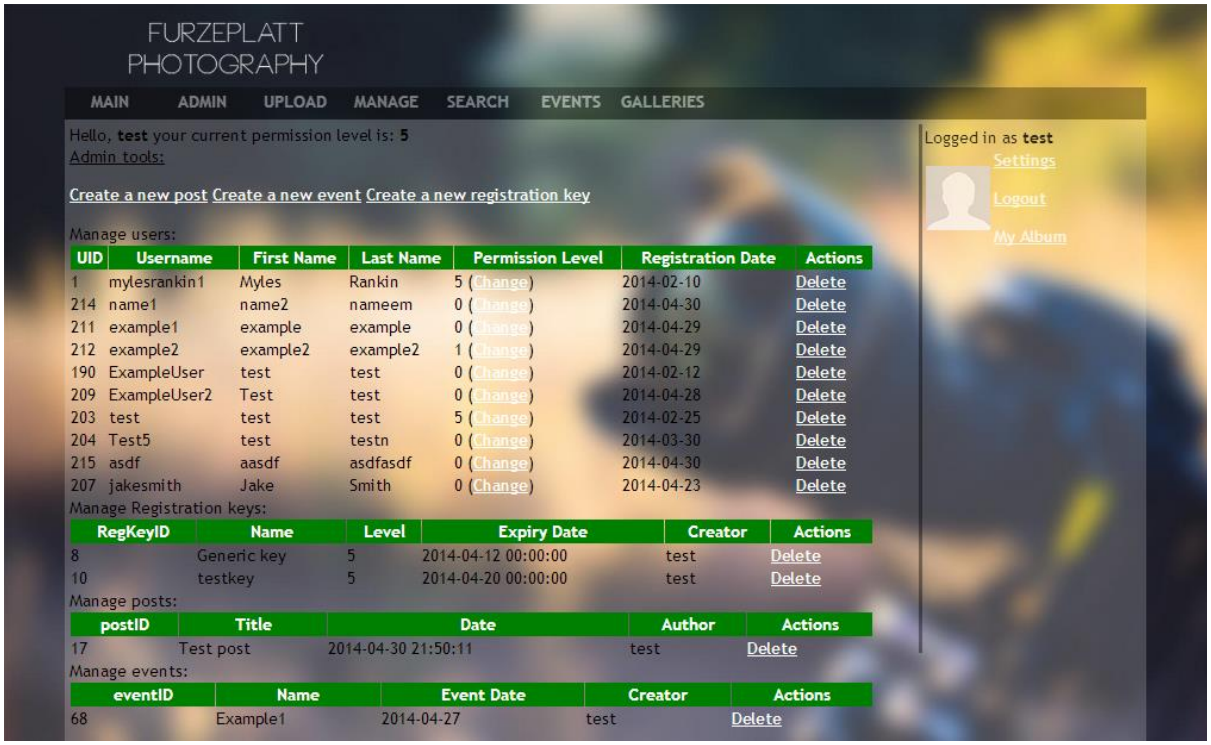
Test – 22



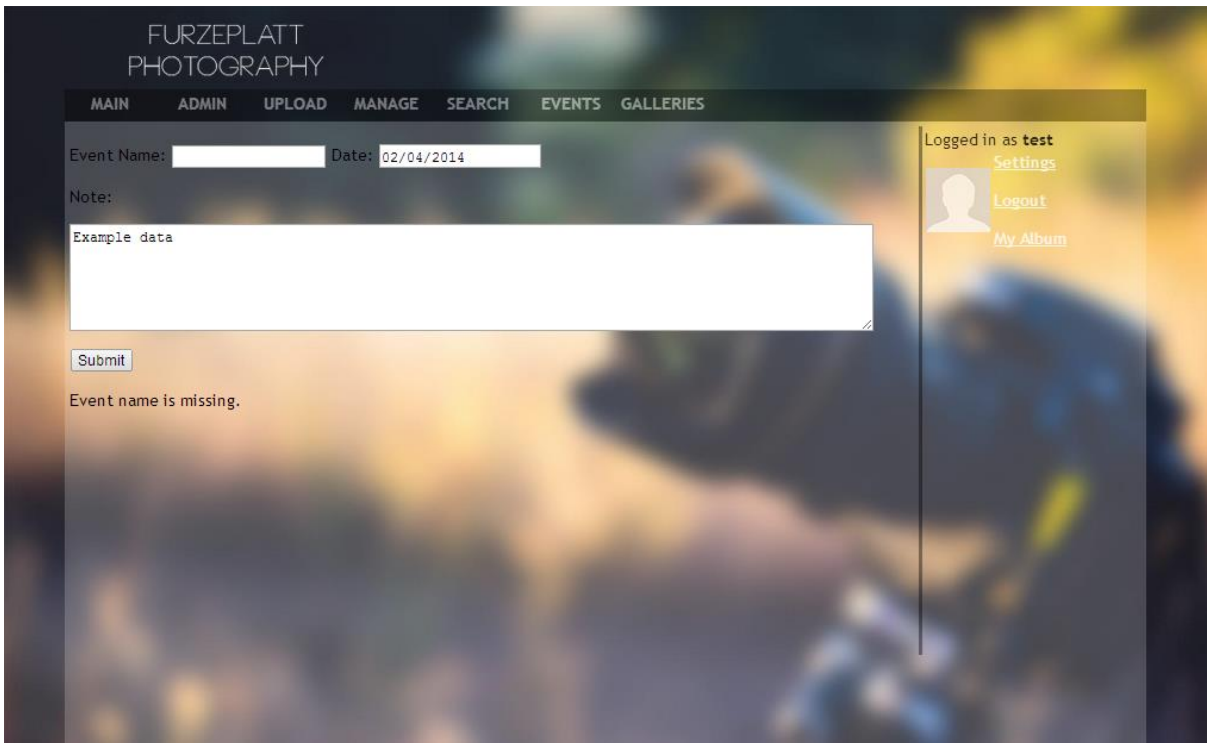
Test – 23a



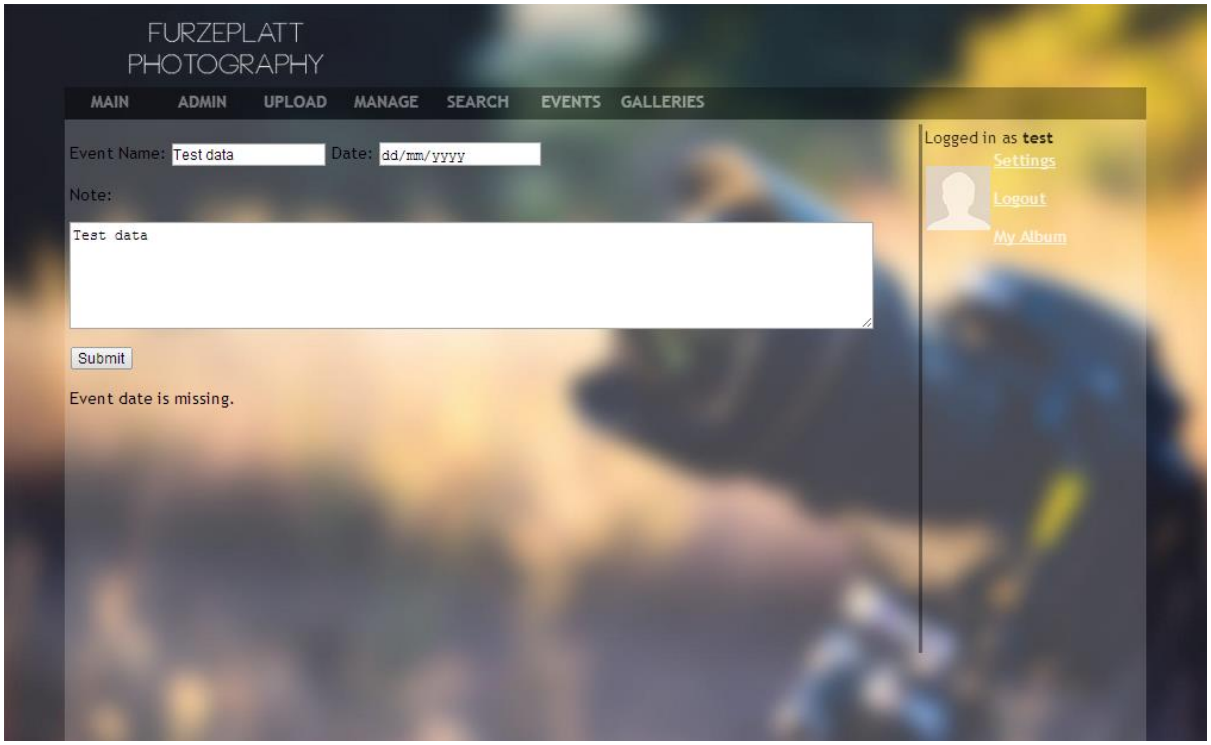
Test – 23b



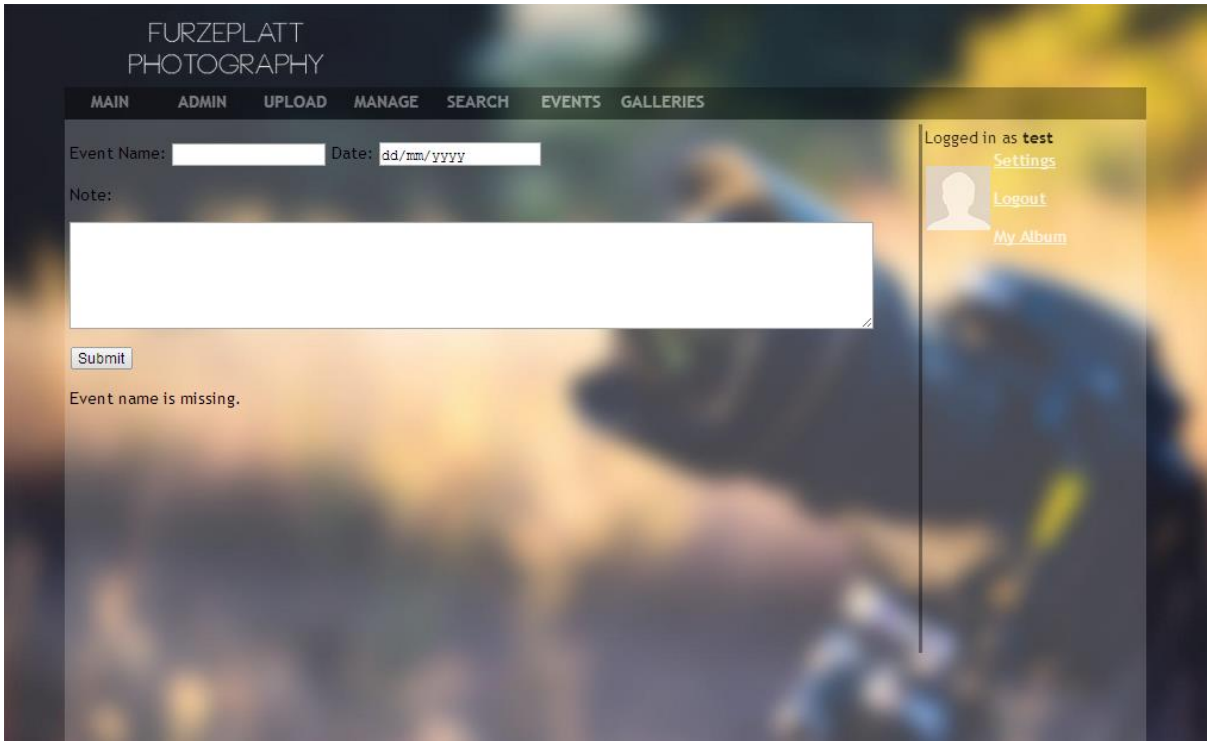
Test – 24a



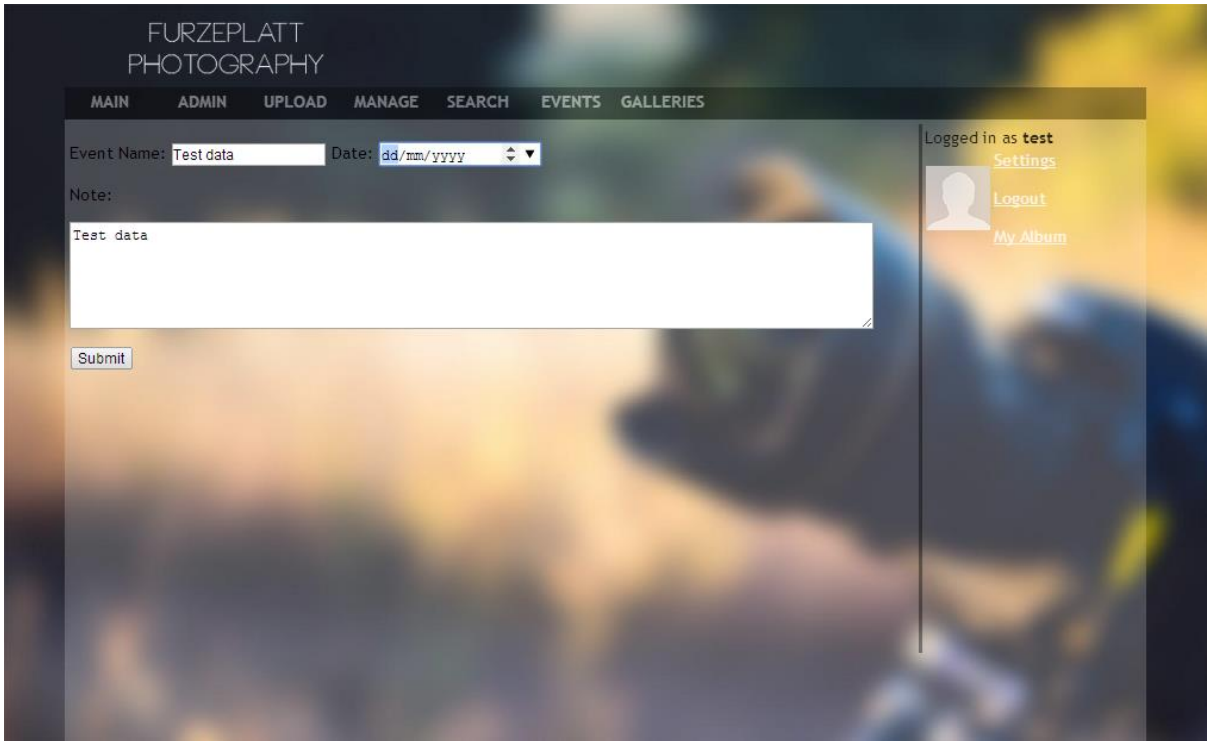
Test – 24b



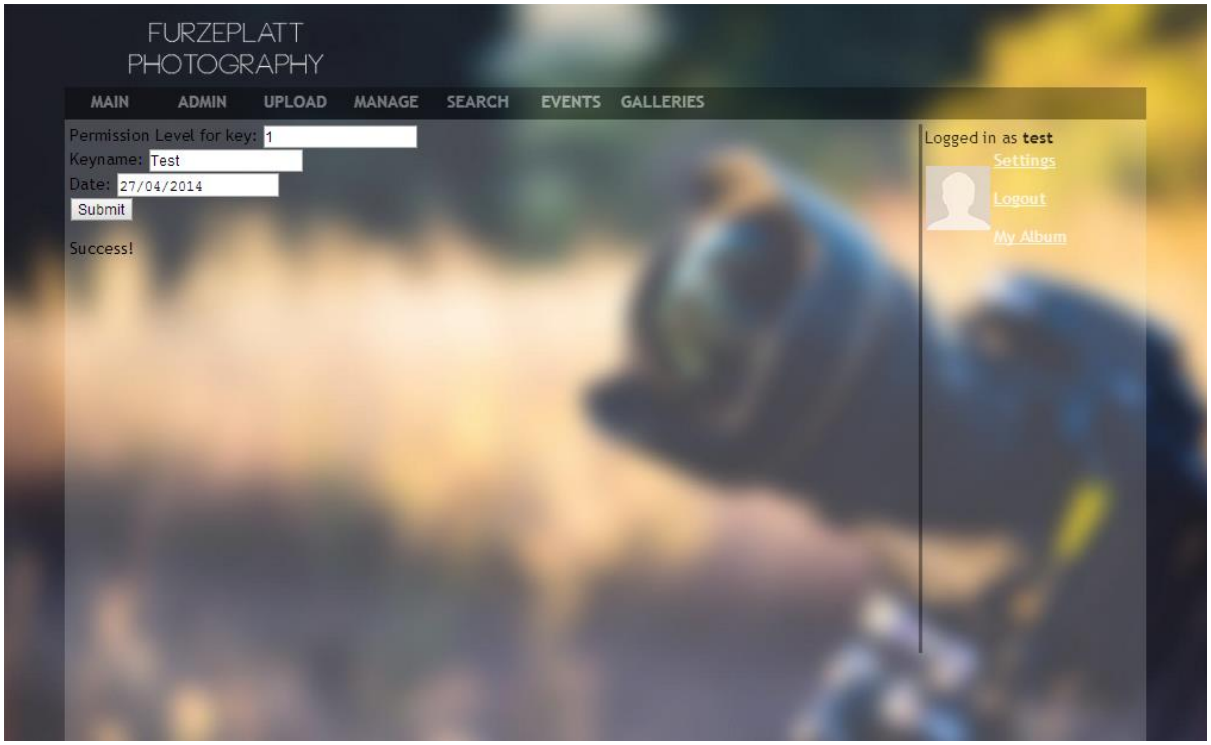
Test – 25



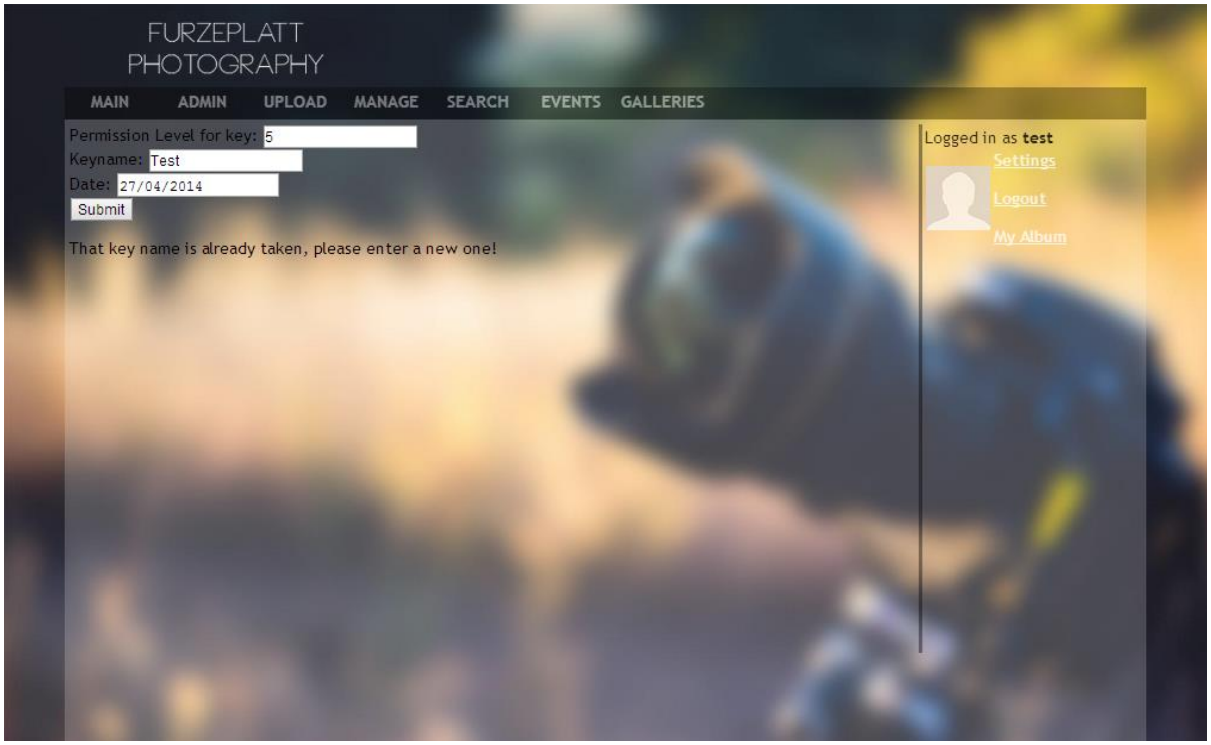
Test – 26



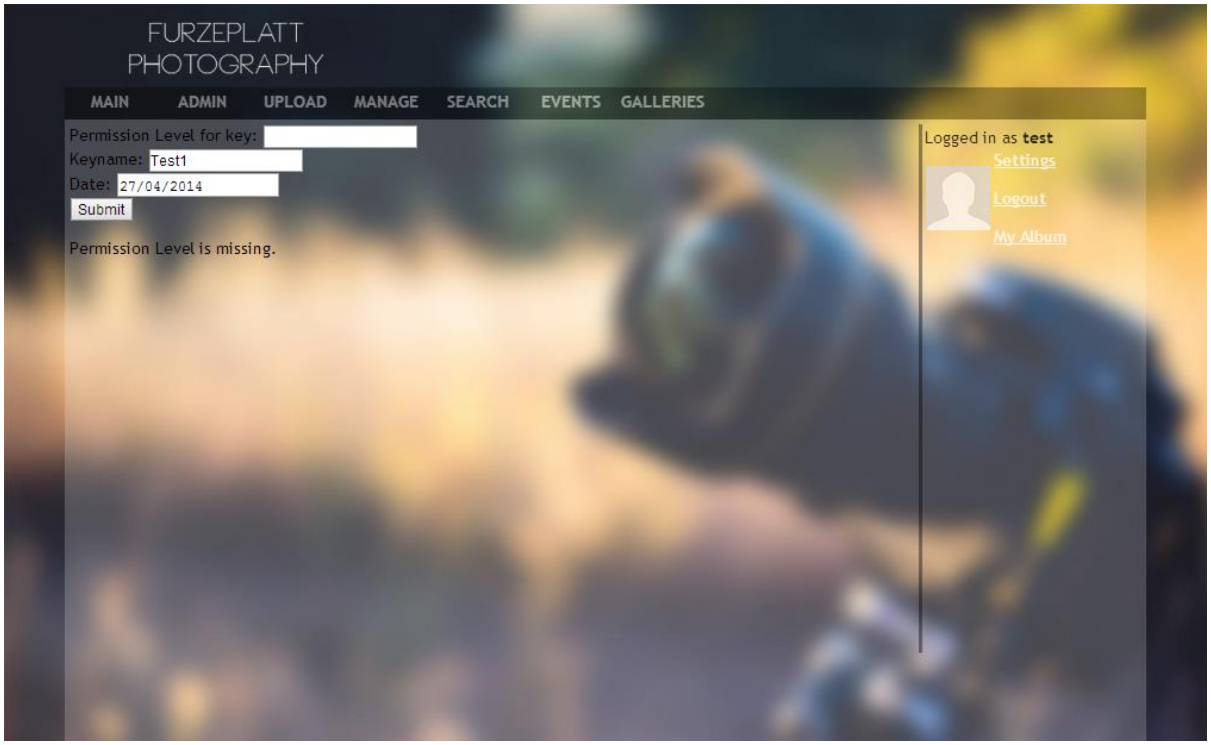
Test – 27



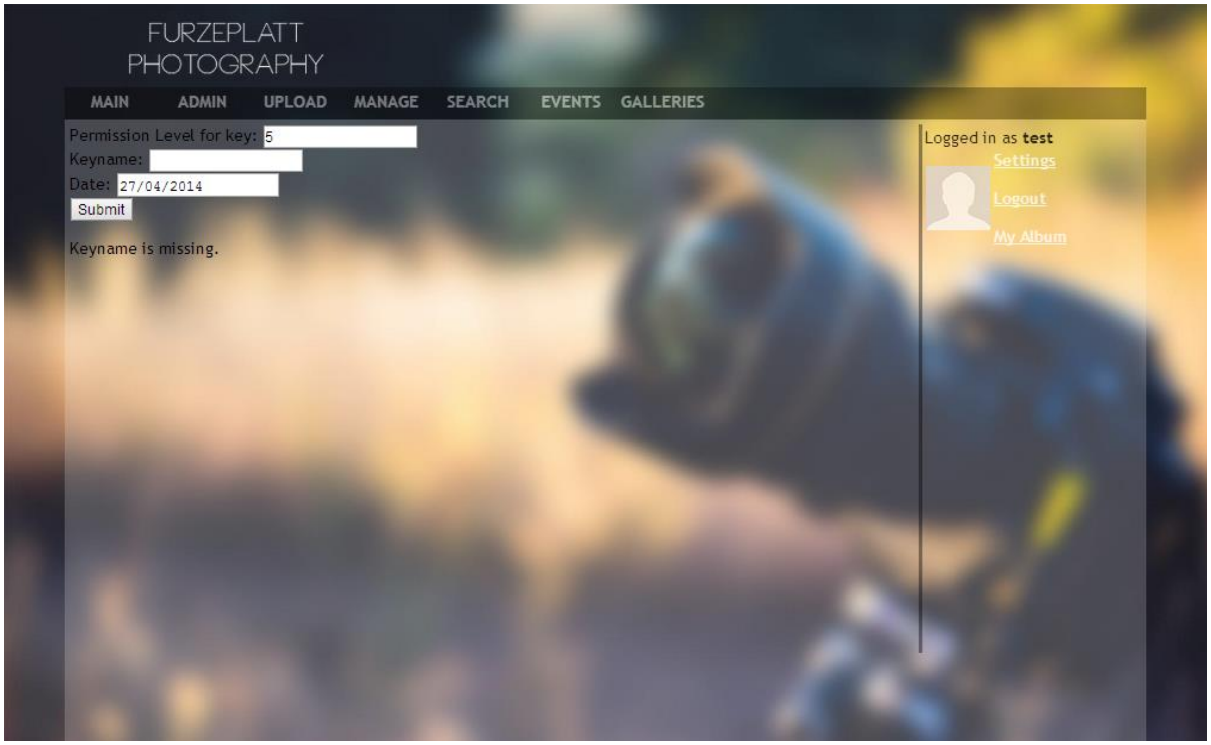
Test – 28



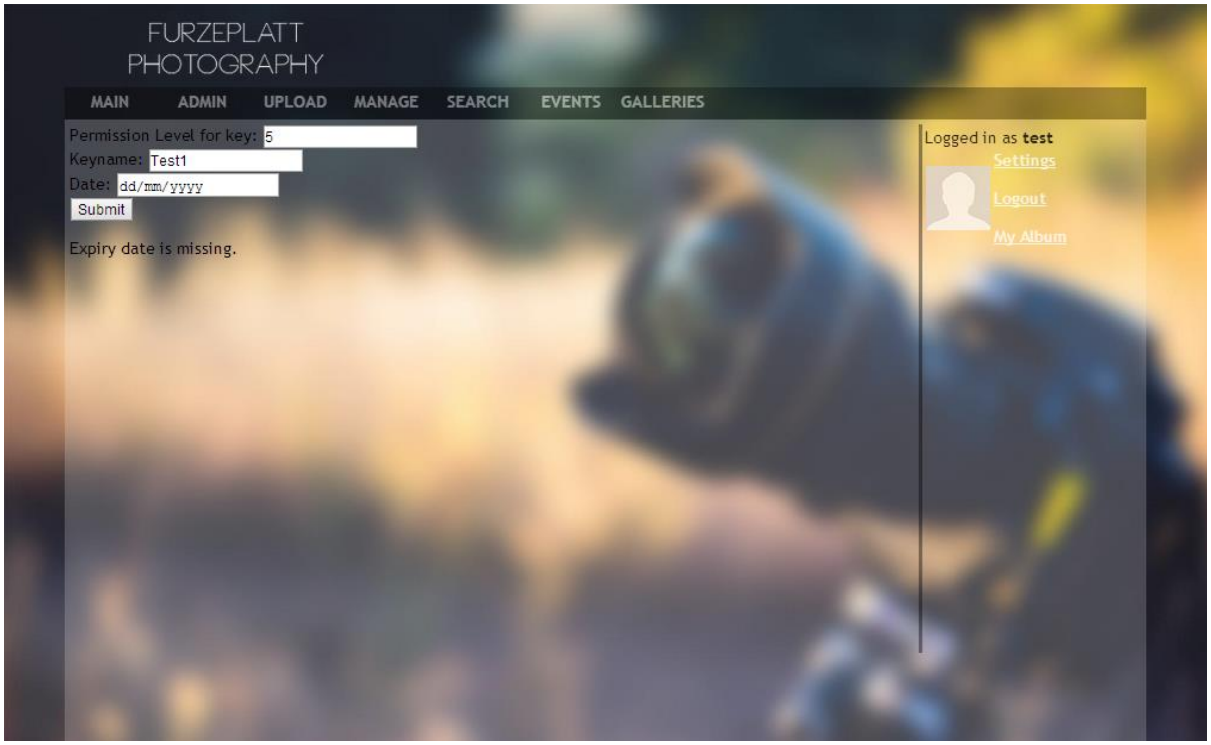
Test – 29a



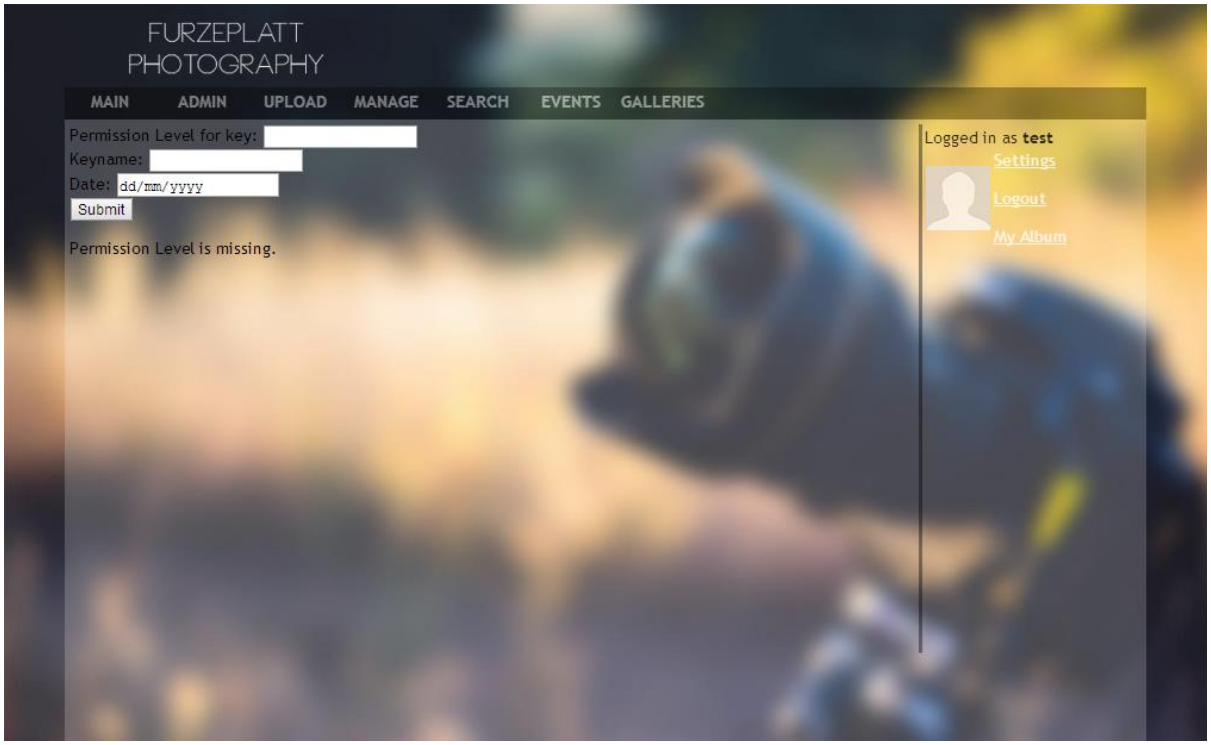
Test – 29b



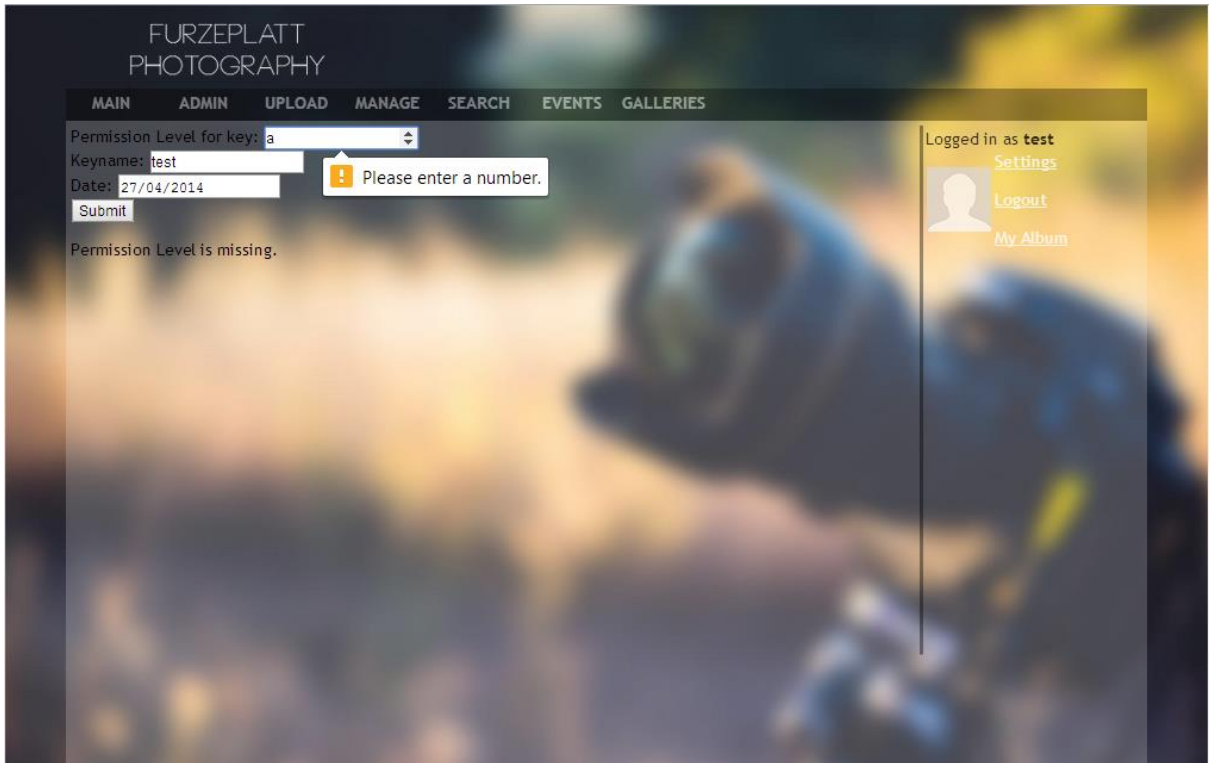
Test – 29c



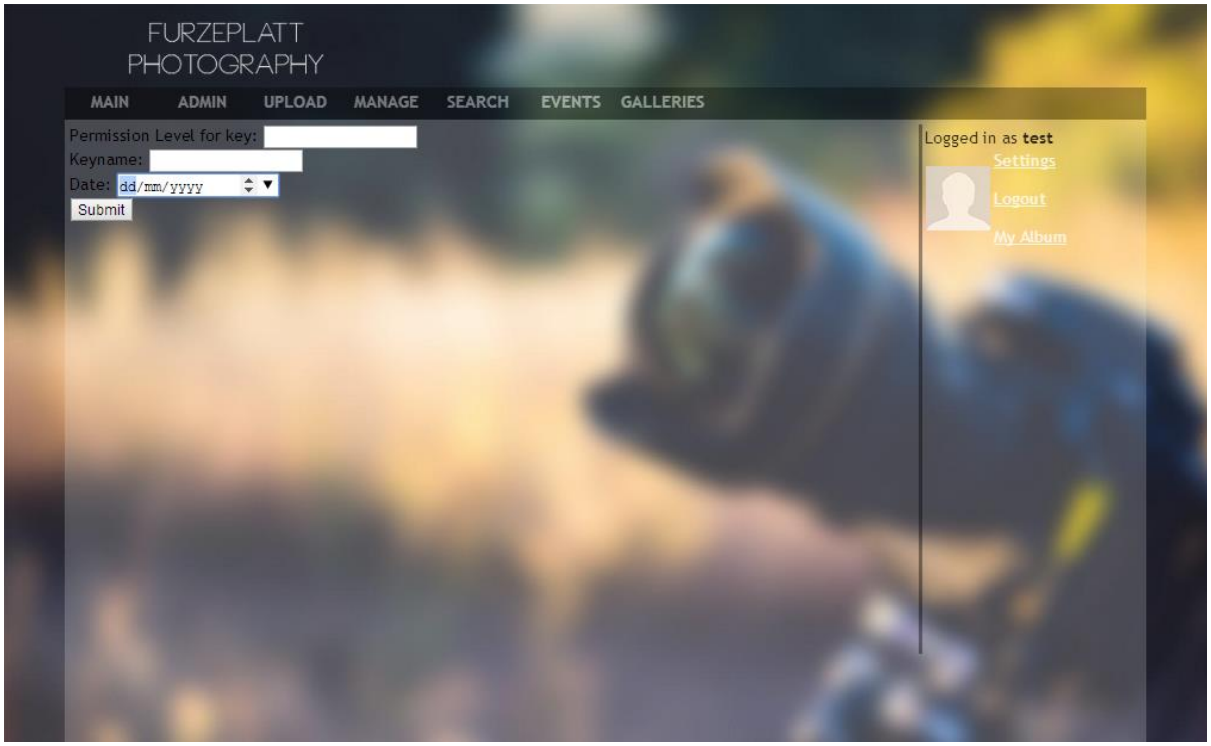
Test – 30


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
Permission Level for key:
Keyname:
Date: dd/mm/yyyy
Submit
Permission Level is missing.
Logged in as test
Settings
Logout
My Album

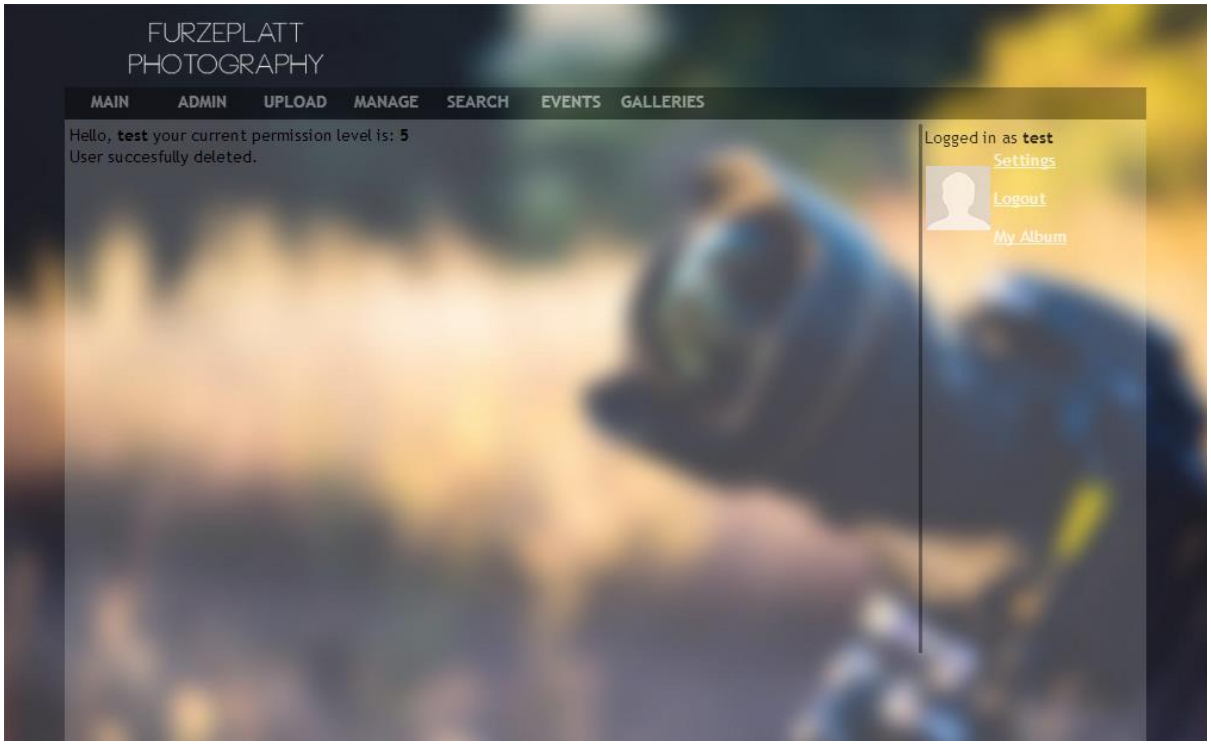
Test – 31


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
Permission Level for key: a
Keyname: test
Date: 27/04/2014
Please enter a number.
Submit
Permission Level is missing.
Logged in as test
Settings
Logout
My Album

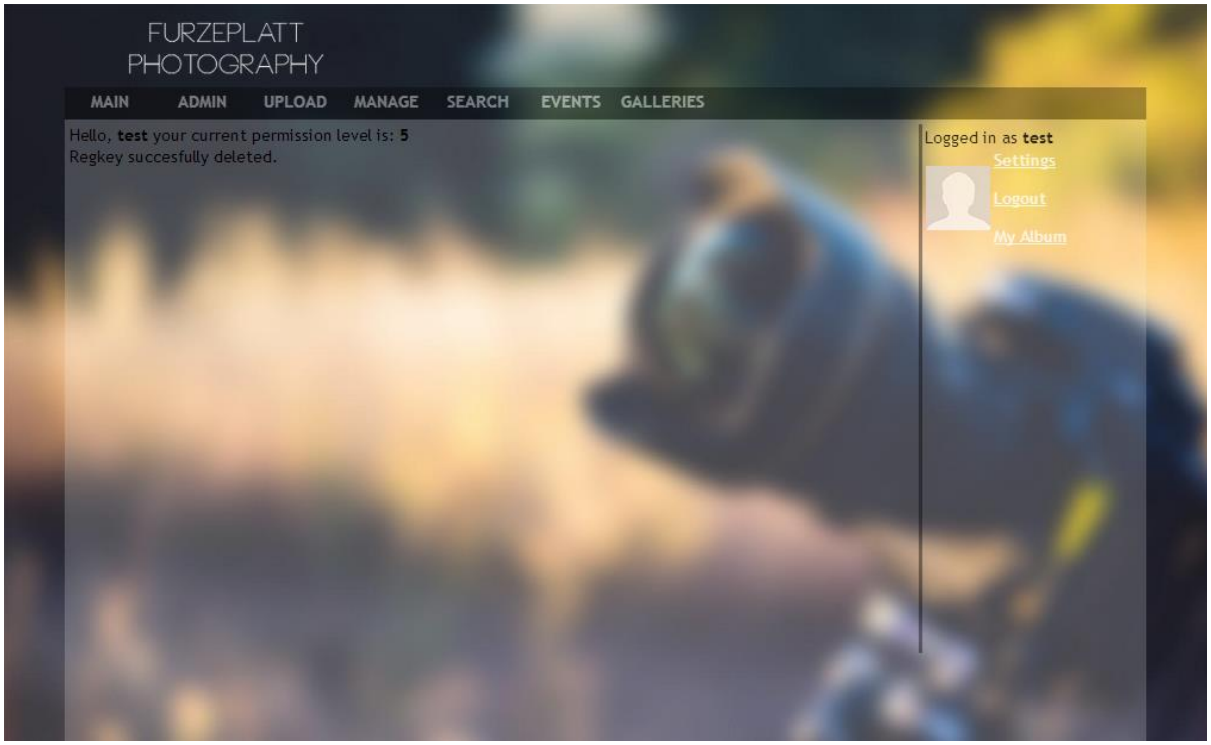
Test - 32



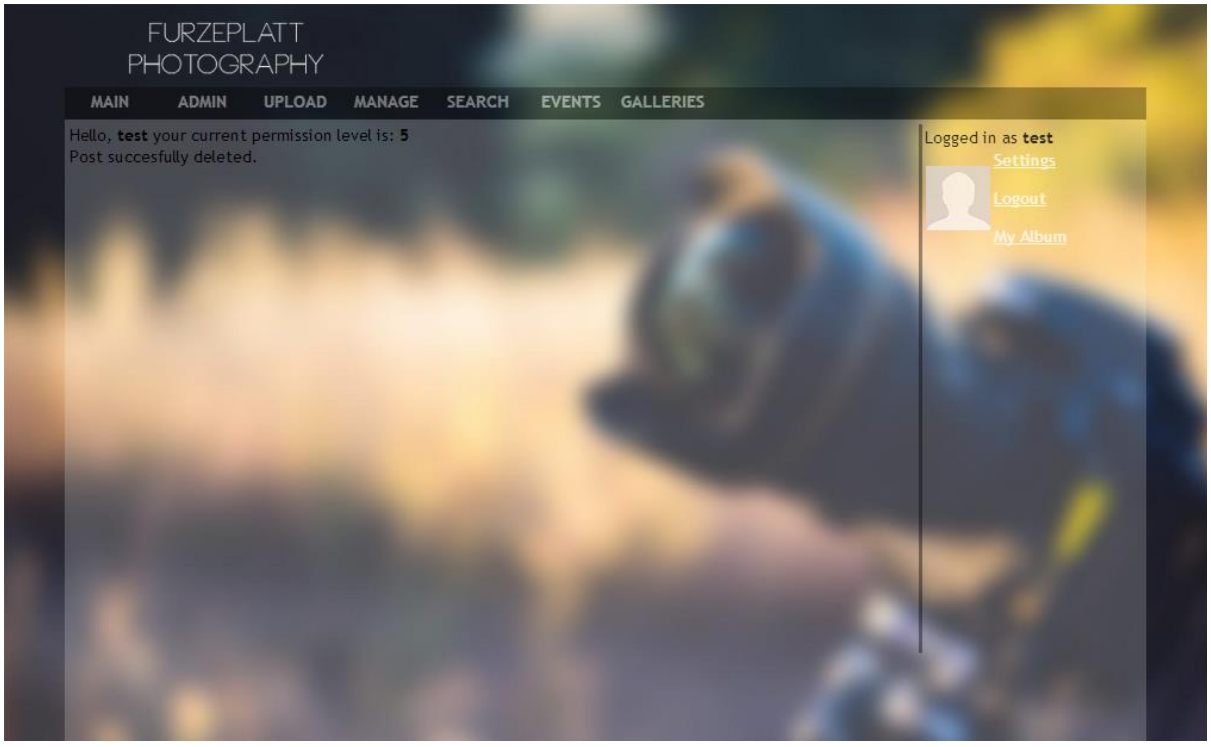
Test – 33



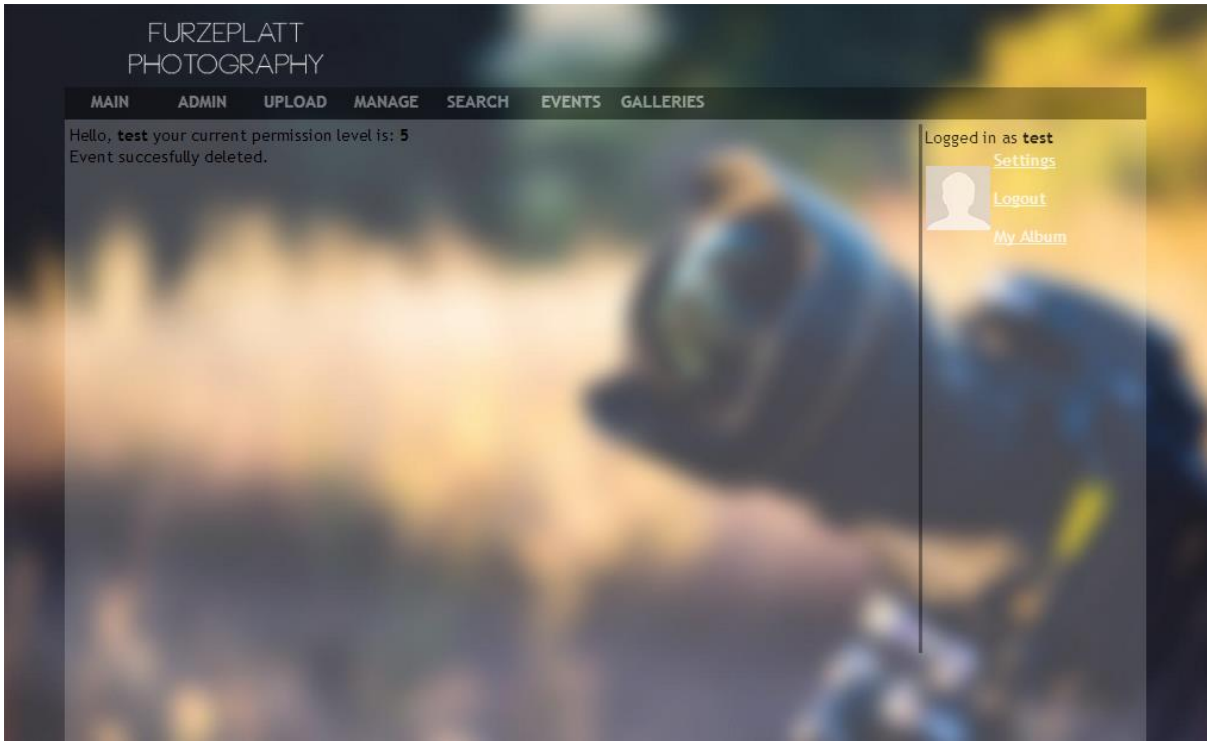
Test – 34



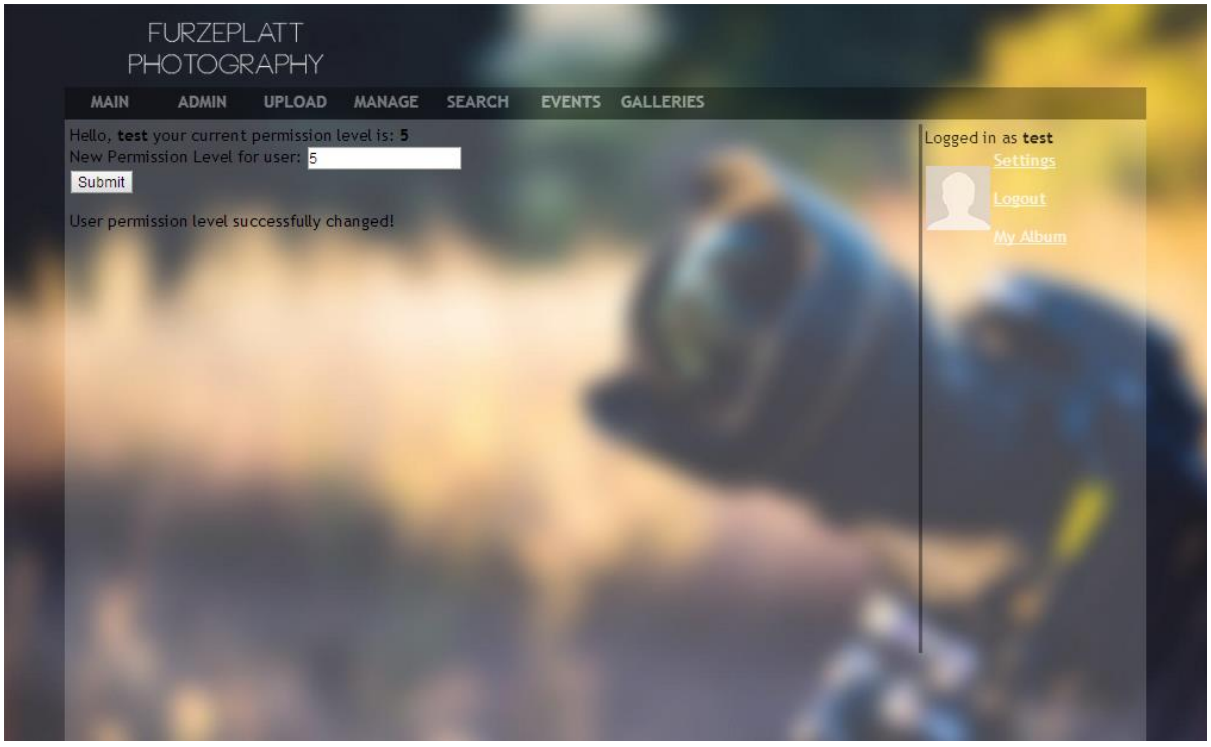
Test – 35



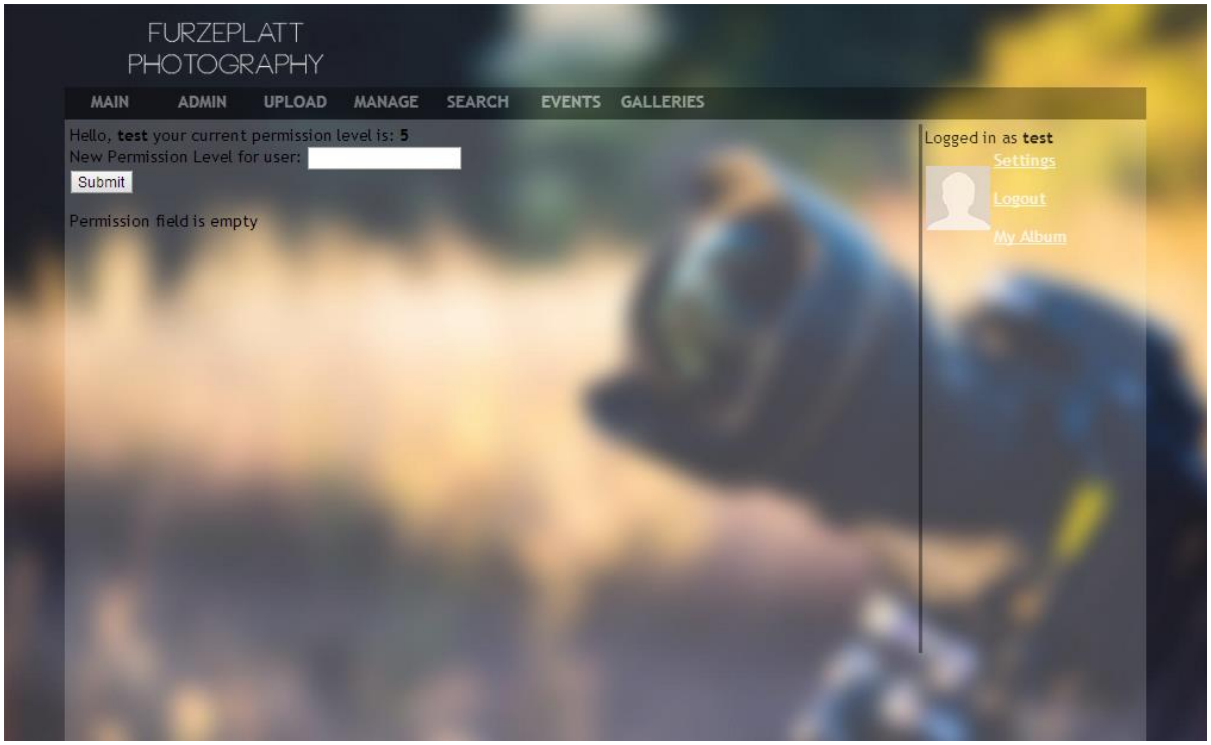
Test – 36



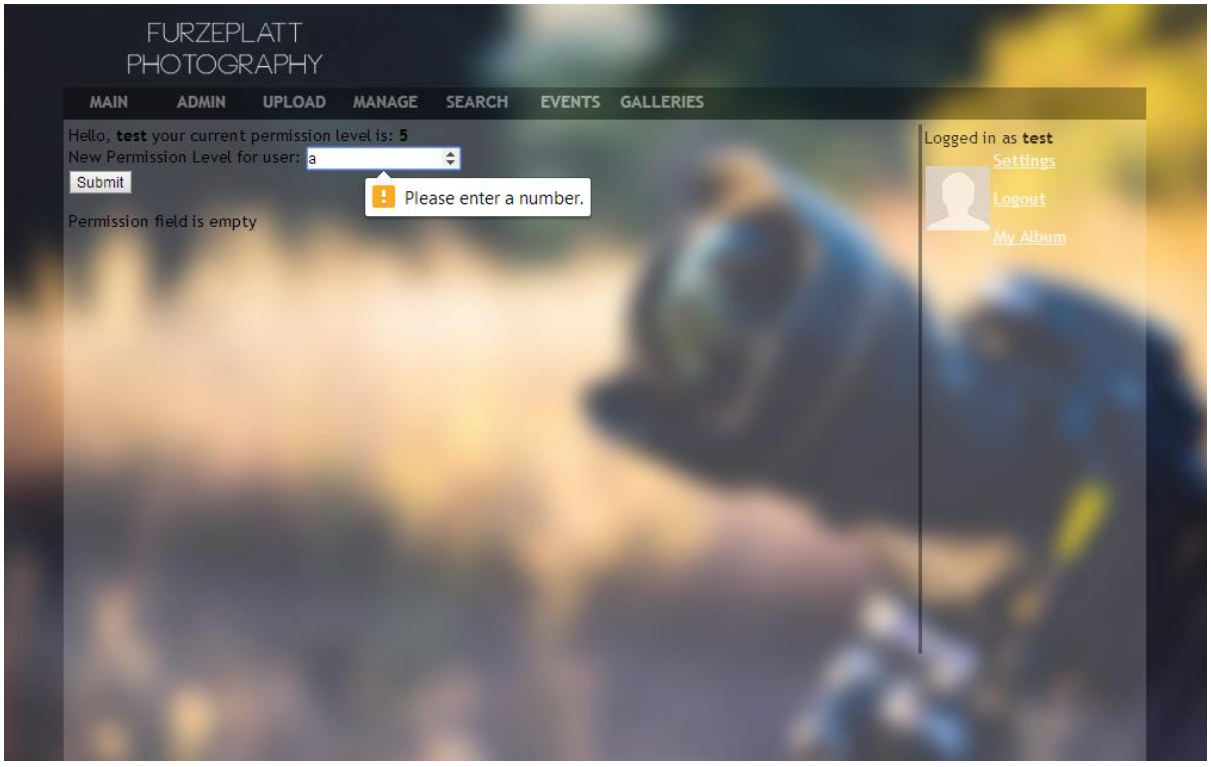
Test – 37



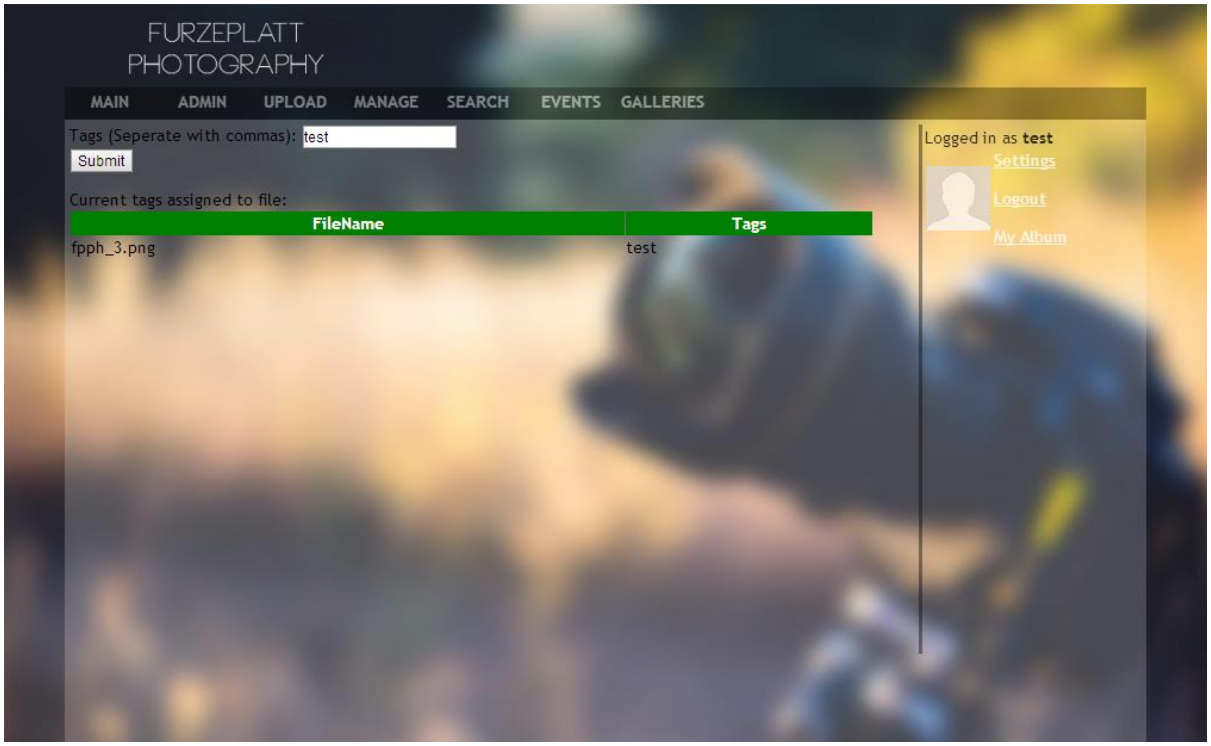
Test – 38


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
Hello, test your current permission level is: 5
New Permission Level for user:
Submit
Permission field is empty
Logged in as test
Settings
Logout
My Album

Test – 39


FURZEPLATT
PHOTOGRAPHY
MAIN ADMIN UPLOAD MANAGE SEARCH EVENTS GALLERIES
Hello, test your current permission level is: 5
New Permission Level for user: a
Submit
Please enter a number.
Permission field is empty
Logged in as test
Settings
Logout
My Album

Test – 40



Test – 41