Vinter 2005 / Forår 2006.

Bachelor opgave

IMM-B.Eng-2006-5

Afleveringsfrist: mandag d. 6/3-2006 kl. 12.00

Holddeltagere:

Studie nr.,	Efternavn, Fornavne	Underskrift
s022350,	*Bech, Peter*	_____

Denne rapport er modtaget af: _____ den ___/___kl.: ___:___

(Udfyldes af instituttet)

Indhold

Indledning

I min bachelor opgave har jeg valgt at udvikle et billeddelings system. Opgaven skal udarbejdes over 10 uger og skal munde ud i en rapport og et produkt. Ideen bag er at brugere skal kunne dele billeder med hinanden online. Dette kunne være venner og familie imellem, som ikke har interesse i at alle skal kunne se deres private billeder.

Problemformulering

Problemstilling

Jeg ønsker som produkt at udvikle en prototype af et billeddelings system, som indeholder alle de centrale funktioner som gør brugere i stand til at dele billeder hinanden imellem. Jeg vælger derfor at fokusere på disse dele, frem for det grafiske. Selve brugergrænsefladen vil jeg prøve at opbygge så simpelt som muligt, men så det stadig kan illustrere funktionaliteten.
En anden grund til at jeg vælger primært at fokusere på billeddelings funktionerne er, at de stiller nogle interessante spørgsmål hvad angår sikkerhed. Hvem der skal have adgang og hvem der skal kunne redigere billederne.

Afgrænsning Af Opgaven

Systemet skal kunne:

Selve billeddelings systemet skal kunne foretage de centrale funktioner i forbindelse med deling af billeder. Derfor skal brugere som et minimum kunne:

- Være i stand til at uploade billeder til deling
- Dele billeder med en anden bruger
- Slette delinger af billeder når de ikke længere skal ønskes delt

For en detaljeret beskrivelse - se kravspecifikationen.

Begrænsninger/restriktioner:

For at kunne opfylde disse ting har jeg valgt at udvikle systemet i PHP/HTML/CSS. CSS/HTML vil blive brugt til at lave brugergrænsefladen og til at styre en brugers input. Til at behandle brugers valg og anden data vil jeg anvende PHP. Dette sætter os også i stand til at kommunikere med en database. Da der skal være flere brugere i systemet har jeg valgt at anvende en MySQL database til at holde styr på brugere og deres data.
For at teste og debugge systemet, er det nødvendigt at have en webserver. Til dette har jeg valgt en apache 2.0 webserver, som er sat op på et MS Windows styresystem.
I denne opgave vil jeg dog kun fokusere på sikkerheden i forbindelse med PHP koden og databasen. Jeg vil derfor forudsætte at web-serveren er konfigureret korrekt.
Da jeg har valgt at anvende PHP som script sprog giver dette os også nogle begrænsninger. PHP er et ”server side” sprog og vi vil derfor ikke være i stand til at eksekvere kode på bruger siden (client side).

Planlægningsfasen

Kravspecifikation

1. *Introduktion*
   1.1. *Formål*. At fastlægge kravene for prototypen af et billeddelings systemet. De efterfølgende krav er tænkt som en prototype af programmet..
   1.2. *Overblik*. Jeg har valgt at dele kravspecifikationen i 3 dele. 1. afsnit skal give en kort introduktion. 2. afsnit giver et indblik i systemets generelle funktioner. I 3. afsnit vil jeg gennemgå systemets tiltænkte funktioner mere specifikt.
2. *Overblik*
   2.1. *Produktets formål*. Programmet skal gøre brugere i stand til at dele digitale billeder med hinanden. Det skal gøres på en nem og sikker måde. Brugergrænsefladen vil være rettet mod IE (Internet Explorer), og vil blive placeret på en web-server så brugere vil kunne få adgang til det ved hjælp af en aktiv Internetforbindelse og en maskine med IE.
   2.2. *Produktets funktioner*. Systemet skal tilbyde følgende funktioner:
      - Oprette en bruger
      - Brugerinddeling Admininistrator/Bruger (om ikke andet til senere udvidelse)
      - Upload af billede
      - Billede management til redigering/deling/sletning af billeder
      - En måde at vise billeder
      - En måde at overskueliggøre delinger af billeder for brugeren.

      Funktionerne er beskrevet mere detaljeret i 3.2.

   2.3. *Brugere*. Programmets brugere defineres som personer der har oprettet sig som bruger på registrerings siden. Brugerne vil kunne blive instrueret v.h.a. en vejledning som kunne tilføjes på hjemmesiden.
3. *Specifikke krav*
   3.1. *Grænseflader*
      3.1.1. *Brugergrænseflade*. Vil være designet minimalt men funktionel, med henblik på at være simpel og overskuelig. Dette skulle gerne hjælpe bruger til at handle intuitivt selv om der er tale om en prototype.
   3.2. *Funktionelle krav*.
      3.2.1. Forside:
         3.2.1.1. Login boks som giver mulighed for at logge ind i systemet
         3.2.1.2. Link til registrerings side
         3.2.1.3. Evt. mulighed for udvidelser som kontakt og præsentation af relevant system info.
      3.2.2. Registrering/opret kunde
         3.2.2.1. En form der kan holde alt relevant information om en kunde.
         3.2.2.2. Alt information skal kontrolleres inden det skrives til databasen
      3.2.3. Login
         3.2.3.1. Her kontrolleres Brugernavn og Password
         3.2.3.2. Data skal skrives til en cookie (krypteret) til brug gennem hele systemet
      3.2.4. Main
         3.2.4.1. Main skal indeholde en oversigt med menu over resten af hele systemet. Denne menu skal være med til at overskueliggøre for brugeren, hvor han befinder sig. Den skal indholde links til de vigtigste hovedfunktioner i systemet. Dvs. Visning af billeder, Styring af billeder og Upload af billeder.
         3.2.4.2. Det skal herfra være muligt at logge ud af systemet
         3.2.4.3. Denne side skal fungere som portal til de andre sider som linker ud fra denne. Derved styrker det også brugerens overblik.
      3.2.5. Album overview
         3.2.5.1. Der skal være en overskuelig oversigt over en brugers mapper
         3.2.5.2. Bruger skal kunne se hvilke mapper andre brugere deler med ham

3.2.5.3. Hvis en bruger trykker på en mappe, får han vist indholdet af den pågældende mappe. Dette skal ske ved hjælp af paging så en bruger selv kan vælge hvor mange billeder han ønsker at få vist pr side. Hvis der kun vises et billede skal der også vises tilhørende kommentarer.

3.2.5.4. Ved hver mappe skal der vises beskrivelse af mappens indhold.

3.2.6. Management

3.2.6.1. Det skal være muligt at slette mapper, der skal dog være et tjek først der sikrer at en bruger ikke kommer til at slette en mappe ved et uheld.

3.2.6.2. Det skal være muligt for en bruger at dele en mappe med en anden bruger. Når han har valgt en mappe og trykker på share, vil han blive sendt videre til en anden side, hvor han kan vælge hvilken bruger, han ønsker at dele mappen med.

3.2.6.3. En given bruger skal kunne se hvilke mapper andre deler med ham.

3.2.6.4. Brugeren skal kunne fjerne en mappe som en anden bruger har delt med ham.

3.2.6.5. Brugeren skal kunne se hvilke mapper han deler med andre.

3.2.6.6. Brugeren skal kunne fjerne en sådan deling, hvis han ikke længere ønsker at have denne mappe delt med den pågældende person.

3.2.6.7. En bruger skal kunne trykke på en mappe han ejer og derved få vist indholdet.

3.2.6.7.1. Her skal han så kunne se navn billede

3.2.6.7.2. Der skal være en mulighed for at ændre informationerne for et givent billede, dvs. navn og beskrivelse.

3.2.7. Upload

3.2.7.1. Her skal være en oversigt over hvilke mapper man ejer

3.2.7.2. Man skal have mulighed for at oprette en ny mappe med tilhørende beskrivelse

3.2.7.3. Man skal vælge en mappe som man vil uploade til.

3.2.7.3.1. Man skal have vist indholdet af mappen

3.2.7.3.2. Det skal være muligt at browse et billede

3.2.7.3.3. Bet skal være muligt at ændre navn på billede

3.2.7.3.4. Brugeren skal kunne tilføje en beskrivelse til det billede han ønsker at uploade.

Risikostyring

Ved et projekter som dette, er det en god ide at tage stilling til ting som sygdom og sikkerhed af projekt data. Så man f.eks. ikke mister hele projektet hvis ens harddisk skulle gå i stykker. Nedenfor har jeg kort beskrevet hvad jeg har gjort for at imødekomme sådanne uforudsete hændelser.

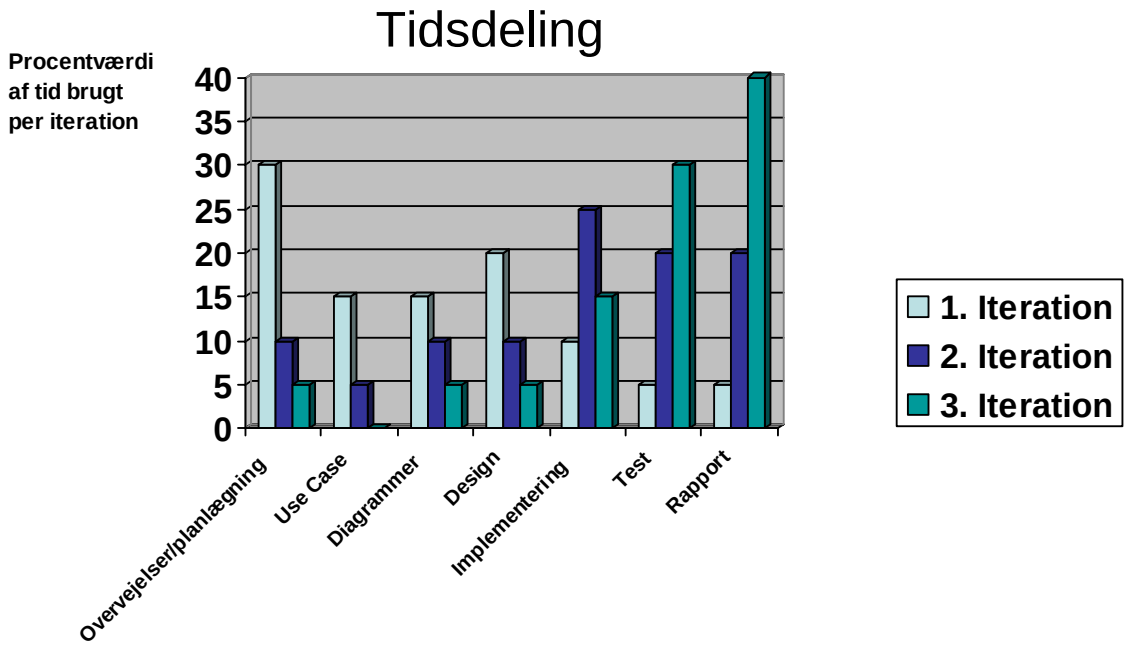
Sygdom

Da projektet bliver udarbejdet over en 10 ugers periode, er det usandsynligt at jeg skulle ligge mig syg. Skulle det alligevel ske, vil en enkelt sygedag eller 2 være til at arbejde ind igen. Hvis det ikke forekommer i slutningen af projektet! Skulle jeg i værste fald ligge syg længere end 4-5 dage, kan det gå hen at blive nødvendigt, at få udskudt aflevering af projektet et tilsvarende antal dage. I så fald er det nødvendigt at skaffe en læge erklæring.

Filer

For at mindske risikoen for tab af data, har jeg valgt at arbejde med 3 sikkerheds kopier. Grunden til dette er at jo længere hen i projektet jeg kommer, desto mere fatalt vil det være at miste data. Jeg har derfor valgt fra starten at arbejde med 2 backup kopier. Da jeg har 2 harddiske i min maskine har jeg valgt at have en kopi på mit andet drev. Derved vil jeg have en kopi hvis den ene harddisk skulle gå i stykker. Den anden backup er forhåbentligt overflødig, men jeg har valgt at have den for ekstra sikkerheds skyld, for at imødekomme f.eks. indbrud. Den ligger på en usb key som jeg holder adskilt fra computeren, når den ikke er tilsluttet for at opdatere den pågældende backup.

Iterationsplan



Ovenstående diagram skal ses mere som en cirka angivelse. Jeg mener dog at den viser meget godt hvordan jeg vil vælge at fordele min tid. Jeg har fra projektets start vidst, at jeg ville benytte mig af 3 iterationer. Jeg har erfaringer med at dette hjælper til at øge overskueligheden af et projekt og derved mindske risikoen for fejl.

I første iteration vil jeg fokusere meget på analysen af min problemstilling. Jeg mener dette er vigtigt, da det er denne analyse der skal danne grundlaget for hele mit projekt. I analysen vil jeg lave Use Case og sekvensdiagrammer for de centrale dele af systemet. Jeg vil ligeledes i første iteration begynde at designe mit program, dels for at få en ide om hvordan det kommer til at se ud og fungere, og dels for at gøre det så brugervenligt og overskueligt som muligt. Jeg vil derfor også begynde at kode en skabelon af GUIén til mit produkt. Denne vil indeholde CSS design og menu til mit produkt. Der vil ikke i første iteration ikke blive kodet på databasen. Den vil dog blive designet her men vil først blive implementeret i anden iteration.

I anden iteration vil jeg implementere databasen og de centrale dele af systemet. Dette skal gøres tæt op af de Use case og sekvens diagrammer, som blev udviklet i første iteration, så jeg ikke kommer til at afvige unødigt fra disse. Der kan dog være tilfælde, hvor man vælger at ændre lidt i disse, da man nogle gange kan støde på en mere hensigtsmæssige løsning. I sådan tilfælde er det

vigtigt at vende tilbage til sine Use case og sekvens diagrammer og opdatere dem, så man hele tiden sikrer overensstemmelse mellem disse og ens program kode. Efterhånden som jeg får afsluttet implementeringen af en funktion vil jeg teste den, så jeg er sikker på at funktionaliteten er i orden. Derved håber jeg også på at minimere risikoen for sikkerhedshuller i systemet. I denne iteration vil jeg ligeledes påbegynde skrivningen af rapport.
I tredje og sidste iteration skal alle de centrale dele af systemet gerne være implementeret. Her skal der så ryddes op og finpudses i koden, så den er klar til de endelige tests. Fokus vil i denne iteration primært være på rapport der skal udarbejdes. Tests af det samlede system vil fylde en del i denne iteration for at finde evt.fejl i den endelige prototype. Hvis disse ikke umiddelbart kan rettes vil de blive dokumenteret i afsnittet ”Fejl og mangler” i rapport.

Analyse

Begrundelse For Valg Af Opgave

Jeg har valgt denne opgave, fordi jeg mener den indeholder nogle relevante og interessante problemstillinger af både sikkerhedsmæssig og programmeringsmæssig karakter. I disse år bliver det mere og mere almindeligt for personer at have et digitalt kamera. Samtidigt med at folk rejser mere og længere, kunne man forestille sig et stigende behov for at man ønsker at kunne dele oplevelser med venner og bekendte hjemme.
Dette skal de kunne gøre på en forsvarlig måde, hvorpå ingen andre end dem, de ønsker, skal kunne få adgang til billederne, da det ellers vil kunne krænke deres privatliv..
Jeg vil derfor i denne opgave prøve at lave et projekt, der illustrerer en måde at dele digitale billeder på. Det skal være så sikkert som muligt i forhold til både ondsindede brugere, og brugere der vil dele billeder af tvivlsom karakter.

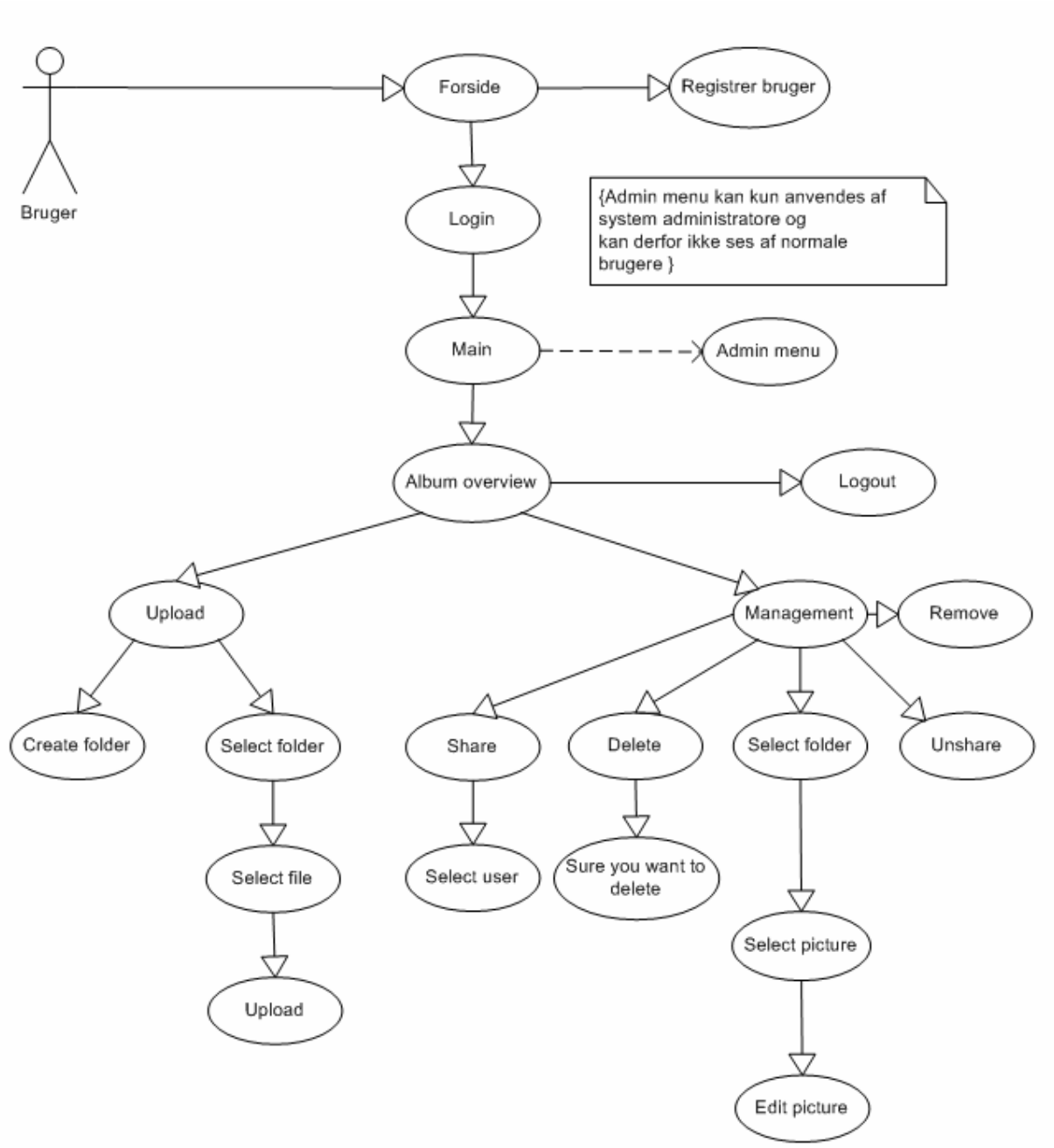
Brainstorm

Jeg har valgt at inkludere den brainstorm jeg startede med i begyndelsen af dette projekt. Jeg mener at den illustrerer nogle af de ting, jeg har valgt at fokusere på i udviklingen af mit system. Både med hensyn til sikkerheden, funktionaliteten og designet.

- Programmet skal være overskueligt og simpelt.
- Alt personfølsomt data i login cookien skal være krypteret.
- Hver gang der vises en ny side skal der kontrolleres om brugeren stadig er logget ind. Derved sikrer jeg løbende, at han er den, han udgiver sig for at være.
- Billederne kan gemmes på 2 måder enten i en BLOG i selve databasen eller som en fil og et link i databasen der så peger til placeringen af billedet.
- Hvis metoden med linket vælges så vær opmærksom på at billederne skal placeres uden for webserveren for at hindre en bruger i at se billeder, der ikke er hans.
- Hvis billederne derimod bliver gemt direkte i databasen som en BLOG skal de så krypteres?
- Når en bruger deler en mappe skal både han og modtageren kunne vælge at fjerne delingen.
- Det er kun ejeren, der skal kunne slette et billede fra systemet. Han skal dog ikke kunne slette billedet ”fysisk” fra harddisk, da en bruger ikke skal have lov til at slette filer gennem systemet.
- Jo mere velkendt man kan lave mappe inddelingen des lettere vil det virke for brugeren.
- En bruger skal kunne tilføje en kommentar til hvert billede. Derved vil andre der får delt billede kunne få en forklaring af billedet.
- En bruger skal kunne redigere i informationerne til et billede, efter han har uploadet det.

Use Case Diagram

Af: Peter Bech



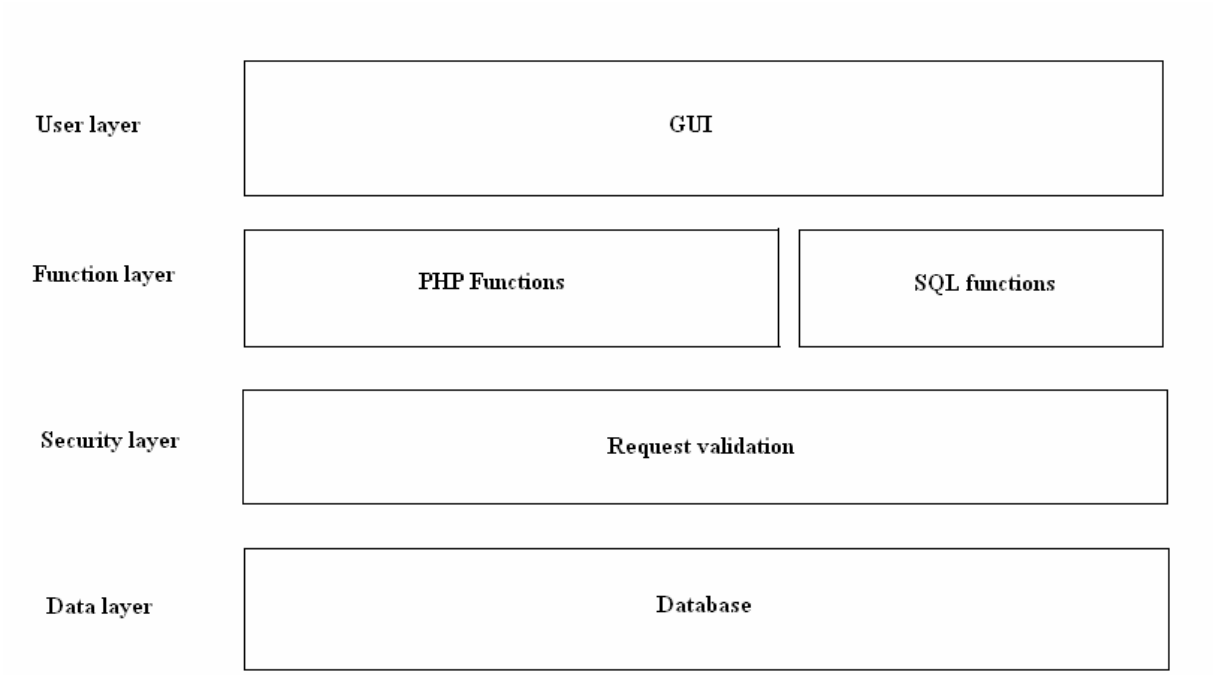
Se bilag 2 for forklarende Use Case scenarier.

Design

Designet af systemet skal danne grundlag for at systemet bliver sikkert, funktionelt og overskueligt. Det er vigtigt, at det bliver lavet på en overskuelig og anvendelig måde for at skabe overblik under implementeringen. Det skal vise samspillet mellem de forskellige dele af systemet. Dvs. vise hvordan de enkelte klasser og funktioner arbejder sammen med hinanden og databasen.

Arkitektur

For at skabe et overblik af systemet, har jeg lavet en lagdelt grafisk præsentation. Den skal illustrere hvordan systemet er bygget op.

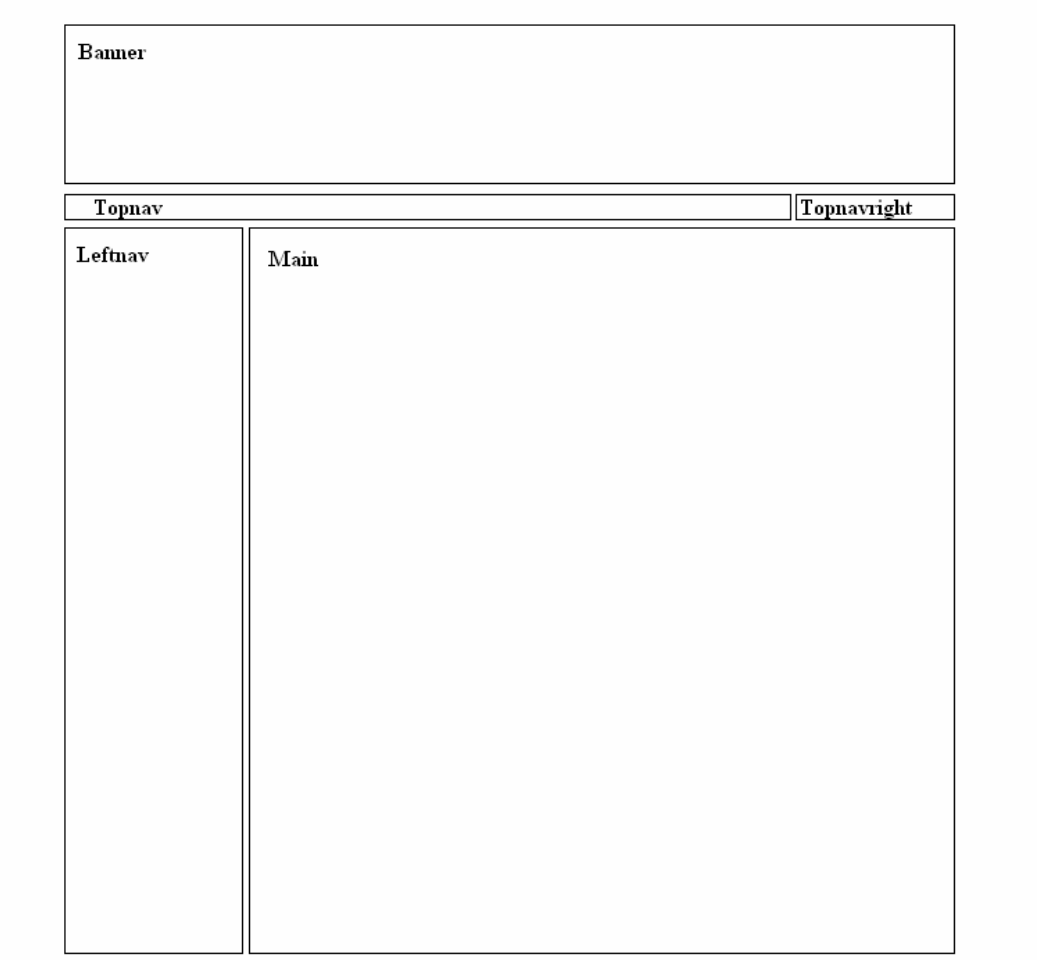


Hvert lag kan sende data til laget under sig og returnere til laget over sig. Dette er gjort for at øge sikkerheden i systemet, da jeg ikke er interesseret i at en bruger sender data til databasen uden at det er blevet kontrolleret. Hvis dette lykkes for ham, vil han formodentligt kunne benytte sig af f.eks. SQL injection og derved kompromittere systemet. Nedenfor vil jeg gennemgå hvert enkelt lag og dets funktion.

User layer:

Dette lag vil blive bygget op ved hjælp af html og CSS. Det er hvad bruger kommer til at se, når han anvender systemet. Det er derfor vigtigt, at der tænkes over interfacet, så det bliver intuitivt at anvende. Dette gøres for at lette brugers bevægelse gennem systemet og holde det så simpelt som muligt.

Til at designe layoutet på siden har jeg valgt at benytte mig af CSS. Dette giver flere fordele. Det bliver lettere at styre systemets udseende og lettere at ændre senere, hvis dette skulle ønskes. Det giver samtidig mulighed for at ændre f.eks. scroll bars og lignende hvis dette skulle have ens interesse. Jeg har valgt et simpelt design til systemet som set nedenfor.



Banneret giver sig selv.Her vil der blive vist et billede med systemets navn.

Nedenfor banneret er der en navigations bar, som jeg har valgt at dele op i to dele: topnav og topnavright. Den venstre halvdel topnav bliver ikke brugt i selve systemet, som det er nu. Men tanken bag den er, at den i senere udvidelse enten til kunne blive brugt til at holde links til andre

dele af systemet eller blive brugt til at hjælpe en bruger med at navigere rundt i systemet ved at give ham en sti over hvor han befinder sig i systemet. F.eks.

Rediger_billed\Billede_folder_navn\Billed_navn

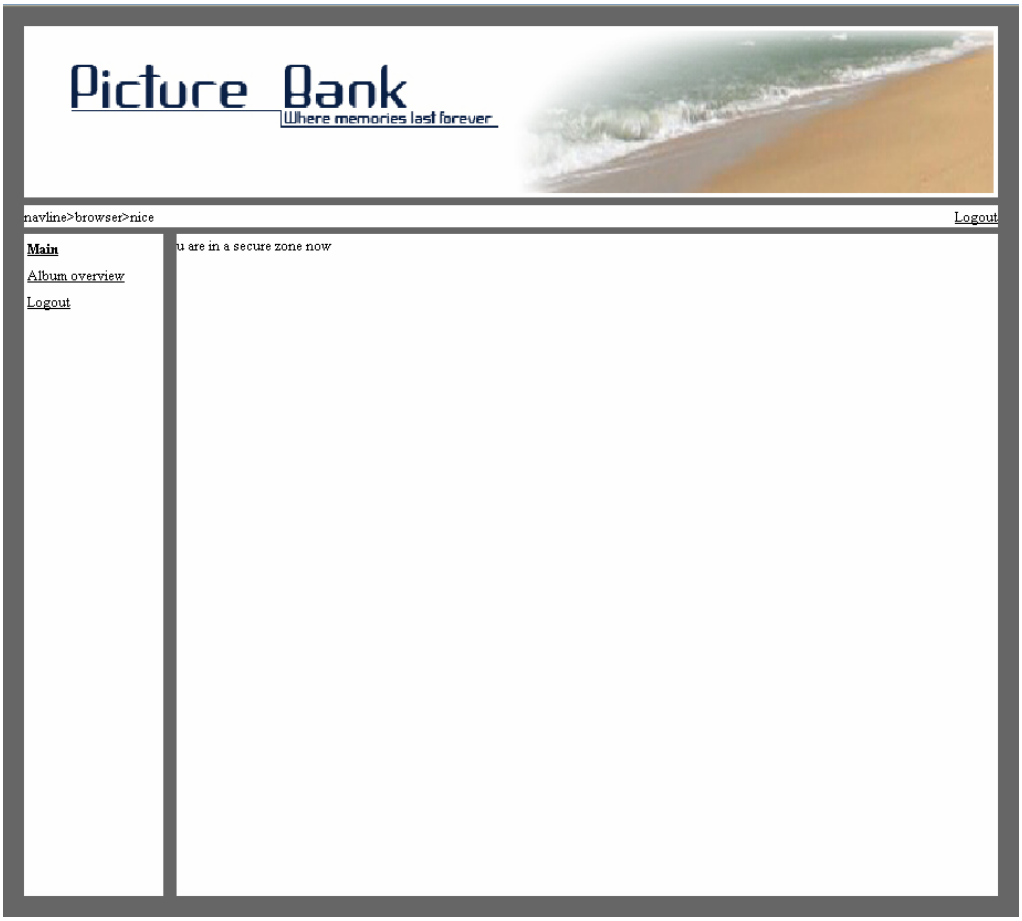
Den højre halvdel vil derimod blive brugt i systemet. Den vil indeholde et link til at logge ud af systemet. Det vil sætte en bruger i stand til at logge ud fra et hvilket som helst sted i systemet.

Nedenfor vil selve hovedmenuen være. Jeg har valgt at kalde den leftnav. Den vil fungere som nagivations bar for hele systemet. Det er her en bruger kan vælge hvad han/hun ønsker af foretage sig. Ved siden af leftnav er main placeret. Main er præsentations delen af gui'en og vil blive brugt til at vise billeder eller funktioner som f.eks. upload af billeder.

Jeg har valgt at hele layoutet af siden skal bygges op som % angivelser som eks:

```
position:absolute;
left:15%;
top:2%;
height:26%;
width:70%;
```

På den måde får jeg en mere fleksibel side, der vil kunne benyttes ved forskellige skærmopløsninger og stadig se ud, som jeg havde til hensigt. Nedenfor ses det endelige layout.



Til dette lag hører også den cookie, som jeg vil bruge til at kontrollere om en bruger er logget ind i systemet. Grunden til at cookien tilhører dette lag er, at brugeren har direkte adgang til den. Derfor udgør den også en mulig sikkerhedsrisiko. Derfor er det vigtigt at minimere mængden af informationer den indeholder. Derved vil den fortælle en mulig ondsindet bruger så lidt om systemet som muligt. Jeg har valgt at min cookie skal indholde følgende informationer:

```
-Kundenummer (ingen)
-Brugernavn (md5)
-Password (sha1)
-Userlevel (md5)
```

Dette er meget brugerfølsomme data, hvis de skulle blive stjålet af en anden bruger. Derfor kan disse data selvfølgelig ikke gemmes som klartekst i cookien. Jeg har derfor valgt at benytte 2 forskellige krypterings metoder til at kryptere disse informationer (se parenteserne ovenover).

Grunden til at jeg har valgt at kryptere med 2 forskellige algoritmer er ønsket om at vanskeliggøre processen for en ondsindet bruger. For en detaljeret beskrivelse af hvordan login fasen fungerer se afsnittet ”login”

Function layer

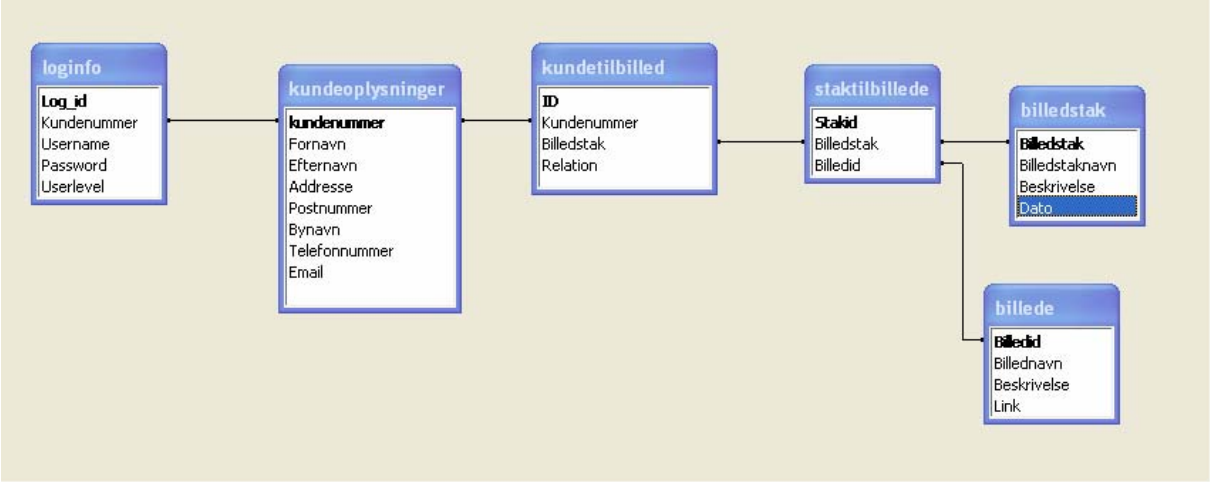
Dette lag består af to dele, PHP funtions og SQL functions. Det er dette lag, der laver al behandling af data. Som f.eks. at sende requests til databasen og returnere arrays med oplysninger til User layer. De sendte requests skal dog først igennem Security layer, som er beskrevet nedenfor.Alle funktioner der er indeholdt i funktionslaget vil blive beskrevet i implementeringsafsnittet. Der henvises til dette.

Security layer

Security layer har til opgave at kontrollere alle forespørgsler og alt data der kommer til databasen fra systemet. Dette er ekstremt vigtigt, for at forhindre ondsindede brugere i misbrug. Måden jeg vil gøre dette på er ved at kontrollere alle variabler der bruges i forespørgsler til databasen. Dette kan jeg gøre, fordi jeg på den måde ved hvad jeg forventer en given variable indeholder. Hvis den ikke indeholder en forventet og dermed lovlig værdi, vil forespørgslen ikke få lov til at blive eksekveret. Til at bestemme hvad en given variable må indeholde vil jeg anvende regular expressions, som er særdeles effektiv til den opgave. Derved kan jeg præcisere hvad en given variabel må indeholde. De enkelte kontrolfunktioner vil blive forklaret i implementations afsnittet(se dbcontrol.php).

Database layer

Dette lag består af selve databasen. Det er her alle oplysninger vil blive gemt såsom placering af billeder, kundeoplysninger og deslige. Jeg har valgt at lave en simpel, men funktionel database, som jeg har normaliseret til 3NF (Normal form). Nedenfor er vist et diagram over det endelige database design.



Der er et sted, hvor jeg har valgt at afvige fra 3NF. I tabellen ”Kundeoplysninger” finder man Postnummer og Bynavn i samme tabel. Dette strider mod princippet om at dele relaterede kolonner i separate tabeller og give dem en primær nøgle som så bliver en foreign key i den gamle tabel. Man kan nemlig her tage Postnummer og Bynavn, lave en ny tabel og så bruge Postnummer som foreign key i tabellen ”kundeoplysninger”. Jeg har valgt ikke at gøre det, da jeg i denne sammenhæng finder det overflødigt. I en større database vil dette dog kunne gøres med nogen fordel.

Funktionalitet

I dette afsnit, vil jeg gennemgå den måde, jeg har valgt at designe de centrale dele af systemet.

Login

Login funktionaliteten af systemet består af to dele. I User Layer eller på brugerniveau, har vi cookien. Denne indeholder oplysninger til at identificere en bruger. Den indeholder som tidligere nævnt fire variabler.

```
-Kundenummer (ingen)
-Brugernavn (md5)
-Password (sha1)
-Userlevel (md5)
```

De samme variabler findes også i databasen. Her bliver de dog lagret på følgende måde af sikkerhedsmæssige årsager. .

```
-Kundenummer (ingen)
-Brugernavn (ingen)
-Password md5(sha1(password))
-Userlevel (ingen)
```

Som man bemærker, er det kun password jeg har valgt at kryptere i databasen, og endda både med md5 og sha1. Jeg er ikke interesseret i at levere mere information end højst nødvendigt i klartekst ,men vil samtidig heller ikke levere krypteret tekst direkte fra databasen. Jeg har valgt at løse problemet på denne måde, fordi cookien er på brugerniveau og dermed en sikkerheds risiko. Den kan blive hijacket eller ændret på anden måde for at en ondsindet bruger på den måde kan tiltvinge sig adgang til systemet. Det er svært at sikre sig mod hijacking af cookies, hvor en ondsindet bruger opsnapper cookien og derved kan udgive sig for at være brugeren. En måde at gøre dette på er ved at lagre ip adressen og kontrollere på denne ved hver request der sendes til siden. Dette er dog ikke implemeneret og derfor er dette system sårbart over for cookie hijacking. Hvad angår ændring af brugerinformation i cookien, har jeg valgt at lave brugerkontrol hver gang en bruger foretager en handling i systemet bliver han kontrolleret på følgende måde:

```
if($_COOKIE[user] == md5($login_data[Username])&& md5($_COOKIE[pass])
== $login_data[Password])
{
}
else
{
            header("Location: /main.php?pathid=0");
}
```

På denne måde sikrer jeg at en bruger, hvis oplysninger ikke stemmer overens med dem i databasen vil blive logget ud af systemet og sendt tilbage til forsiden. Umiddelbart har en ondsindet bruger ikke interesse i at ændre brugernavn og password, hvis han i stedet bare kan ændre kundenummeret, og derved skifte bruger identitet. Dette sikrer jeg også i mod ved ovenstående kontrol. $Login_data bliver sat ved følgende funktion.

```
$login_data = check_login($_COOKIE[kundenummer]);
```

Denne funktion bliver kaldt, hver gang en bruger foretager en handling. Dette vil medføre at hvis en ondsindet bruger prøver at ændre kundenummeret i cookien så vil $login_data ændre sig. Dette medfører at brugerinformationerne i cookien og brugerinformationerne i $login_data ikke længere er ens. Bruger vil derfor blive logget ud af systemet og returneret til forsiden, når han foretager en handling.

Check_login funktionen returnerer også Userlevel, som er den variabel, der angiver hvorvidt en bruger er en administrator. Denne variabel vil også have interesse for en ondsindet bruger, da administrator adgang for en sådan, altid vil være at noget at stræbe efter. Jeg har en administrator menu, som kun vil blive vist hvis følgende validerer:.

```
if($_COOKIE[userlevel]==md5($login_data[Userlevel])&&login_data[Userlevel] == 1)
{
?>
            <tr><td><a href=main.php?pathid=99>Admin tools</a></td></tr>
<?PHP
}
?>
```

Han vil derfor ikke få vist administrator menuen, uanset om han ændrer sin userlevel eller ej. (Medmindre hans userlevel i databasen også er 1, og derved vil han i forvejen være en administrator).

Skulle en ondsindet bruger alligevel finde en work around, vil der være endnu et sikkerheds check, som ser ud som nedenstående.

```
if($_COOKIE[user]  ==  md5($login_data[Username])  &&  md5($_COOKIE[pass])  ==
$login_data[Password] && $_COOKIE[userlevel] ==
md5($login_data[Userlevel]) && $login_data[Userlevel] == 1)
            {
                        header("Location: PHP/administrator.php");
            }
            else
            {
                        header("Location: /main.php?pathid=0");
            }
```

Derved vil han blive logget ud af systemet, hvis oplysningerne i cookien ikke stemmer overens med dem fra databasen.

Upload og lagring billeder

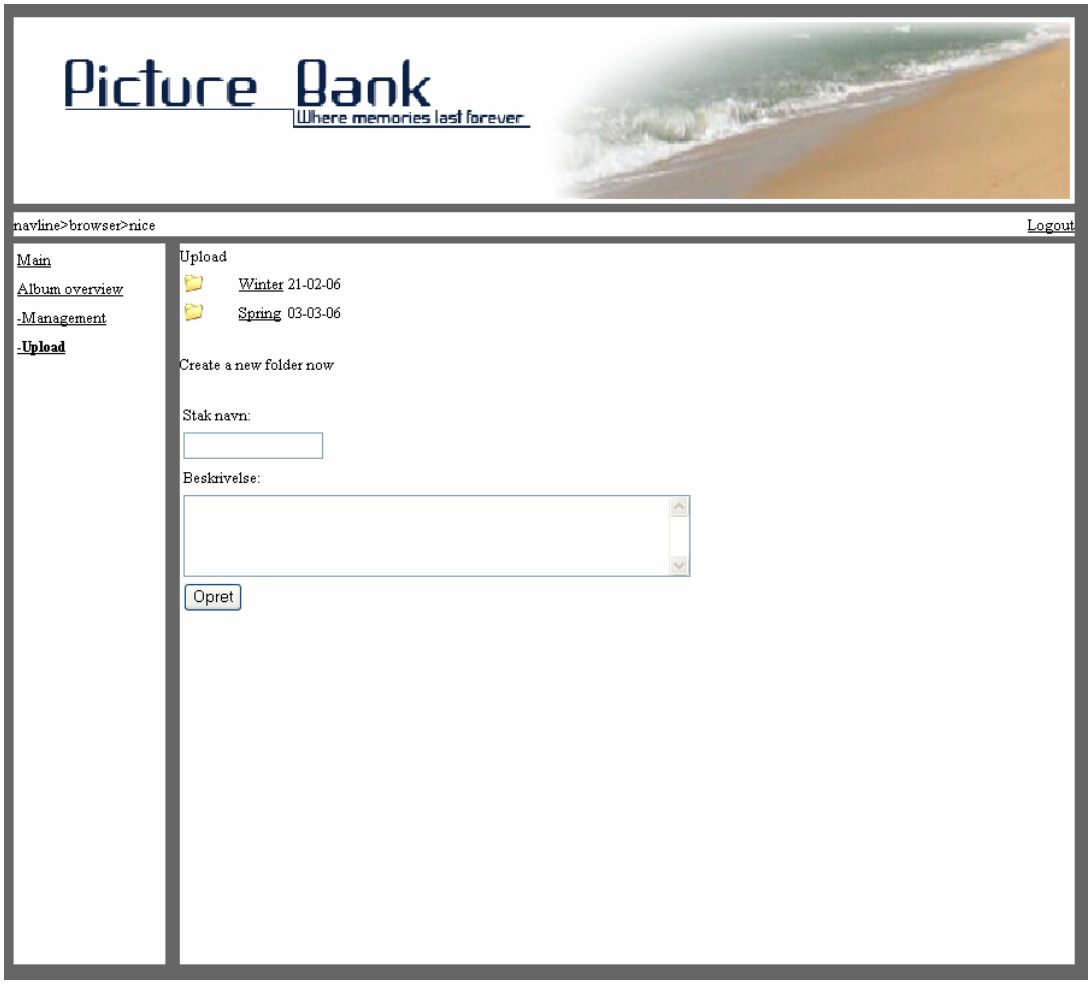
Til at lagre billeder har jeg kigget på 2 muligheder (se brainstorm) .Jeg har valgt at lagre billederne på serveren og så gemme et link til det enkelte billede i en database. Jeg mener at gemme billeder direkte i database, strider i mod en databases fundamentale principper, der bygger på, at alt data skal være simpel og entydig.
Jeg har i denne prototype valgt at gemme billederne inde på selve webserveren, på følgende måde:

```
$path = $_SERVER['DOCUMENT_ROOT'];
$folder = "/archives/";
$dirname = $folder . $stackid;
$link = $path . $dirname;
$uploaddir = $link
move_uploaded_file($filetmpname, $uploaddir . '/' . $filename)
```

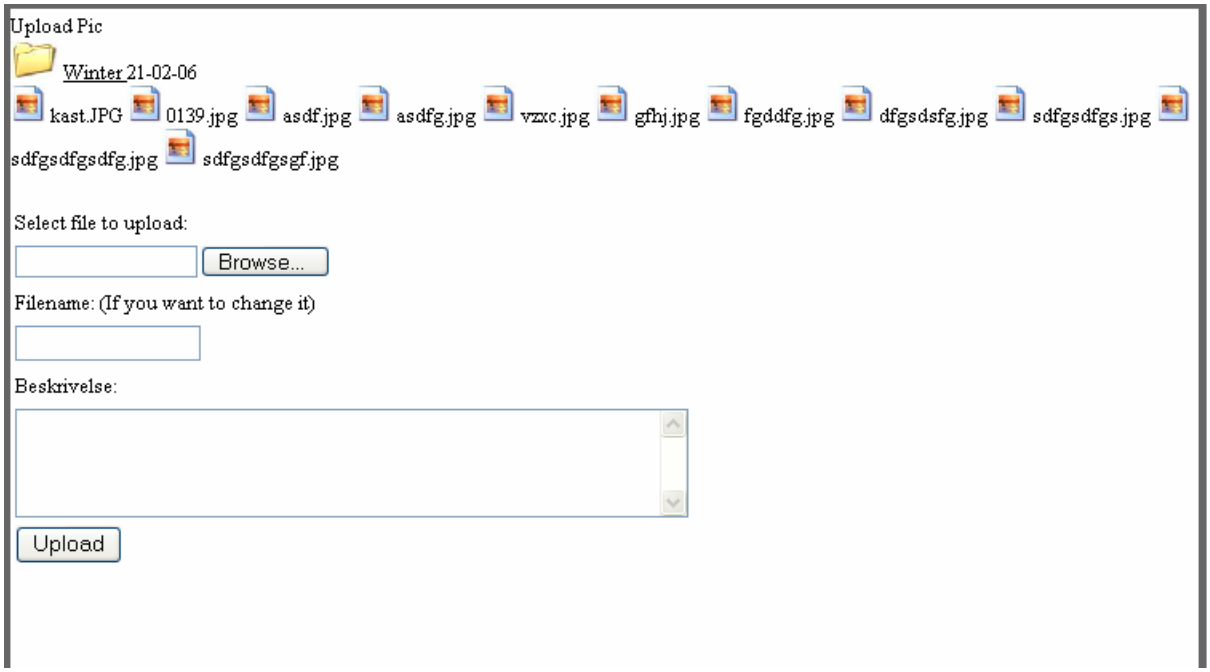
Dette udsagn er placeret som statement i en if sætning. Returnerer dette udsagn sandt vil følgende funktion blive eksekveret, der så vil oprette alle de relevante informationer i tabellerne.

```
$full_link = $uploaddir . '/' . $filename;
$uploadprogress = create_billed($filename, $beskrivelse, $full_link, $stackid);
```

Det bør bemærkes, at det bestemt ikke er hensigtsmæssigt at lagre filerne på selve webserveren. Dette vil medføre at en hvilken som helst bruger, vil kunne få adgang til dem ved at browse rundt. Det er derfor af afgørende betydning, at billederne ved en udvidelse af systemet bliver placeret udenfor selve webserveren, da dette ellers vil medføre en alvorlig sikkerhedsbrist.
Selve upload siden er blevet designet i 2 dele. På den første del har brugeren mulighed for at vælge en mappe eller billedstak om man vil, hvor han eller hun kan uploade et billede til. Eller brugeren kan oprette en helt ny mappe. Denne mappe / billedstak vil svare til en mappe på serveren. Denne side ser i den endelige prototype således ud.



Når en bruger vælger en mappe, vil han blive sendt videre til næste side, hvor han har mulighed for at vælge et billede som han ønsker at uploade. Han kan samtidig ændre navn på det eller tilføje en beskrivelse. Dette vil han dog også kunne gøre senere. Selve designet ser således ud, når en bruger har valgt en mappe at uploade et billede til:



Deling af billeder

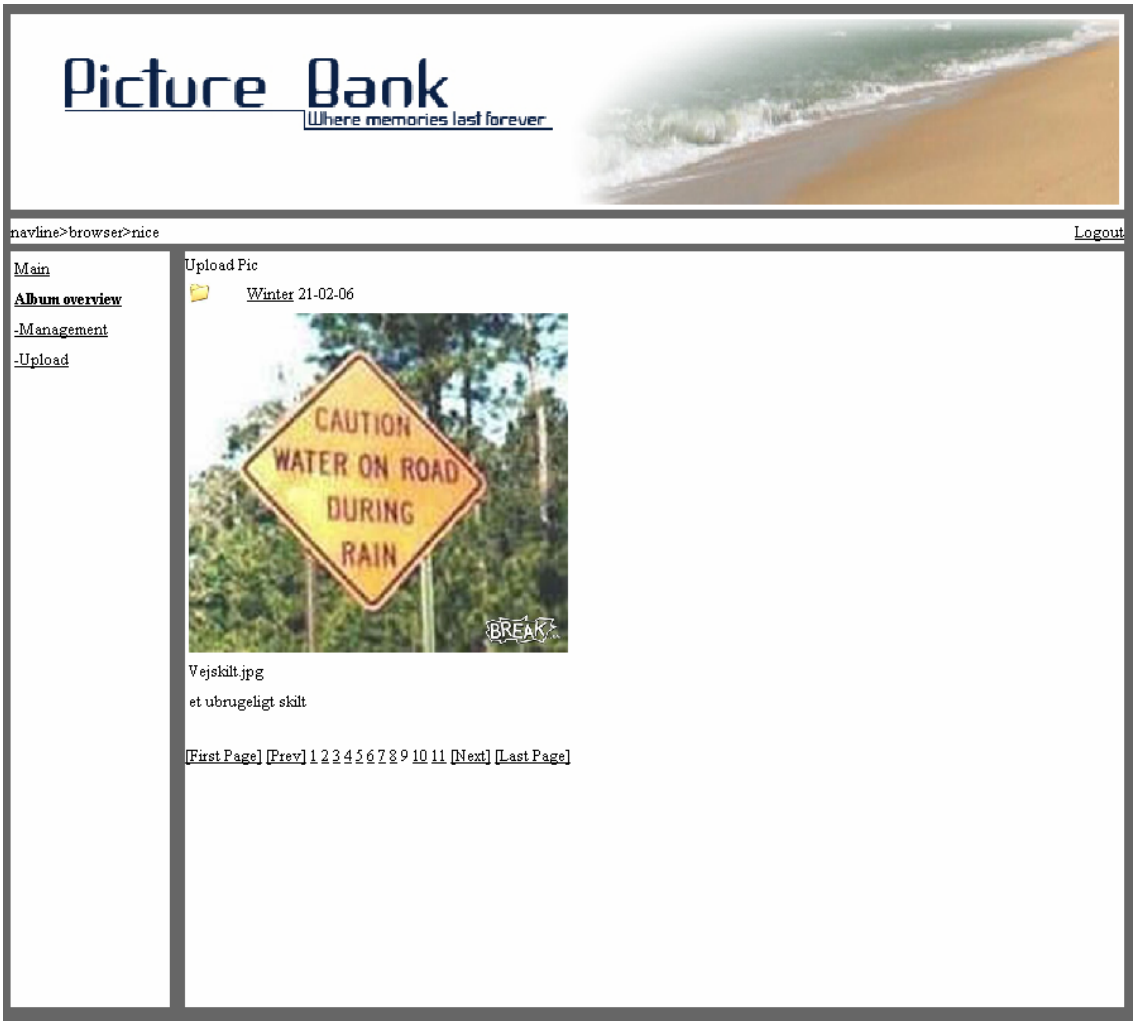
For at en bruger kan dele en billedmappe med en anden bruger kræver det, at han ejer rettigheder over den pågældende mappe. Dette sikre jeg mig ved kun at vise de mapper i formen som brugeren har relation '1' over. Det vil sige, at han ejer mappen . Jeg benytter mig af funktionen:

```
user_stacks($_COOKIE[kundenummer], 1);
```

Som det kan ses tager denne funktion et 1 tal som argument, hvilket er relationen (for en nærmere beskrivelse se implementering). En bruger vil nu kunne dele mappen og hele dens indhold med en anden bruger. Dette sker ved at brugeren som skal have tildelt mappen får den tilføjet i tabellen "kundetilbillede", Dette sker med relation 0, der er tegn på at mappen er en mappe, som han har fået tildelt. Han vil derfor hverken kunne slette eller redigere noget i denne mappe. Han vil dog være i stand til at se billederne som den indeholder ved hjælp af visningen.

Visning af billeder

Visningen af billeder vil foregå ved hjælp af paging. Som det er nu vil der kun kunne vises et billede per side. Men funktionerne der tager sig af pagingen, er udvidet til at kunne returnere flere billeder og dermed flere billeder per side. Ved visning af et billede vil der blive vist navn, og en eventuelt beskrivelse, hvis en sådan er indtastet til det givne billede. Her under ses et eksempel på den endelige implementering.



Det bør bemærkes af billederne som det er nu vil blive vist ved en størrelse på 60 % af deres originale størrelse. Se fejl og fremtidige forbedringer for mere information.

Sletning af billeder

Sletning af billeder er en lidt omstændelig affære. Jeg vil ikke tillade at en bruger skal kunne slette filer på selve serveren. En af grundene til dette er, at en bruger kunne finde på at dele billeder med andre, som har en tvivlsom karakter. Hvis en bruger uden videre kan slette disse igen vil de være svære at frembringe, i tilfældet af at de skal bruges til en senere retssag. I stedet har jeg valgt at begrænse mig til, at en bruger kun kan slette en hel billedstak fra databasen. Den vil dog stadig ligge fysisk på serveren. En billedstak vil blive fjernet ved hjælp af følgende 3 sql sætninger:

```
$query = "DELETE FROM kundetilbilled WHERE Billedstak = '$stack'";
$query = "DELETE FROM billedstak WHERE Billedstak = '$stack'";
```

```
$query = "DELETE FROM staktilbillede WHERE Billedstak = '$stack'";
```

På denne måde fjerner vi alle relationer, og billedstakken vil for brugeren se ud, som om den er fjernet helt. Som man kan se bliver billederne ikke fjernet fra tabellen ”billeder”. Dette er der en grund til. På den måde vil en administrator kunne implementere en funktion, så han kan se hvilke billeder der ligger på serveren og fjerne disse efter f.eks. 5 år. For at se forslag til hvorledes en sådan funktion vil kunne fungere se ”mulige udvidelser”.

Implementering

I dette afsnit vil jeg beskrive de funktioner, jeg har lavet i hver klasse. De vil blive gennemgået klasse for klasse. Ved hvert afsnits start vil jeg kort forklare, hvad formålet er med hver enkelt klasse.

administrator.php

Administrator.php, er den side som kun en given administrator har adgang til. Den indeholder ikke nogen funktioner i øjeblikket. Der er lavet et sikkerheds check, der kontrollerer at en bruger virkelig har administrator rettigheder. Dette sikrer, at siden ikke kan blive vist til en tilfældig bruger, der indtaster det direkte link i sin browser. Hvordan dette check fungerer, er forklaret i afsnittet ”login” i design delen af rapporten.

album.php

Album.php, er den side, der tager sig af præsentationen af billeder. Den benytter sig af funktioner fra:

```
dblogin.php
dbpicture.php
dbupload.php
```

Selve siden er delt op i to dele, der trigger på om en stack er valgt eller ej. Er der ikke valgt en stack, vil album.php vise alle tilgængelige mapper(stacks). Når brugeren så vælger en stack vil siden reloade med en sat stack som følger:

```
http://localhost:xxxx/PHP/album.php?stack=26
```

Den satte stack vil først blive kontrolleret, for at sikre at brugeren har rettigheder til at se den valgte stack. Men også for at sikre , at han ikke har prøvet at skrive noget andet end et stacknummer ind i denne variable. Validerer dette check ikke vil han blive logget ud af systemet uden advarsel. Validerer dette check vil siden, per default, vise første billede i mappen. Men da siden her benytter sig af paging, vil der, hvis der er flere billeder i mappen, komme en menu i bunden hvor brugeren kan bladre i billederne. For at denne menu fungerer er jeg nødt til at benytte mig af en variable mere. Derfor kommer mit link til at se således ud:

```
http://localhost:xxxx/PHP/album.php?stack=26&page=2
```

Page er variablen, der fortæller hvor i billedstakken vi befinder os, så min paging menu fungerer korrekt.

dbcontrol.php

Denne fil indeholder funktioner til at kontrollere alle variabler, der bliver anvendt i forbindelse med min database. Jeg benytter `eregi`, som ikke er case sensitiv, til at kontrollere et udsagn. Derved er det ikke nødvendigt at tilføje high case karakterer til de enkelte regular expression (regex).

`validate_text($text)`

Denne funktion bliver brugt til at kontrollere bl.a. fornavn og efternavn. Den anvender et regex der ser således ud:

```
"^[a-z]{1,32}$"
```

`validate_addresse($addresse)`

Denne funktion benytter jeg til at validere en adresse. Denne skal kunne indeholde tal, bogstaver og mellemrum. Derfor ser det regex således ud:

```
"^[0-9a-z ]{1,32}$"
```

`validate_postnummer($postnummer)`

Funktionen har til formål at validere et postnummer. Et postnummer består af 4 tal hvilket får dette regex til at se således ud:

```
"^[0-9]{4}$"
```

`validate_telefonnummer($telefonnummer)`

Et telefonnummer består som minimum af 8 tal, hertil kan komme 2 tal mere for landekode og et ´+´. Derfor kommer mit regex til at se sådan ud:

```
"^[0-9\+]{8,12}$"
```

`validate_email($email)`

For at validere en e-mail har jeg valgt følgende regex:

```
"^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-
z0-9-]+)*$"
```

Dette regex tillader en e-mail adresse der må indeholde tal, bogstaver, _ , . . Den skal indeholde et @ og et punktum efterfulgt af bogstaver og tal.

`validate_user($user)`

Denne funktion anvender jeg til at validere et username, det tillader en bruger at benytte tal, bogstaver, underscore og bindestreg i brugernavnet. Det regex jeg benytter ser således ud:

```
"^[_a-z0-9-]{1,32}$"
```

`validate_kundenummer($kundenummer)`

Denne funktion bliver brugt til at validere et kundenummer. Kundenummeret bliver genereret af systemet og består udelukkende af tal. Derfor ser det regex som jeg anvender således ud:

```
"^[0-9]{1,12}$"
```

`validate_stackname($stackname)`

Et stackname er navnet på en bruger-oprettet mappe (stack). Dette navn må indeholde tal, bogstaver og mellemrum. Dette gør brugeren i stand til at navngive en mappe med f.eks. dato og sted. Det regex som jeg benytter til at validere et mappenavn med ser således ud:

```
"^[a-z0-9 ]{1,100}$"
```

`validate_beskrivelse($beskrivelse)`

En beskrivelse bliver benyttet til at beskrive en mappe eller et billede. Denne funktion benyttes til at validere et sådanne:

```
"^[a-z0-9\.,:() ]{1,250}$"
```

`validate_stack($stack)`

En stack er et autogeneret nummer som benyttes til at identificere en billedstak. Dette nummer består af tal. Der får derfor mit regex til at se således ud:

```
"^[0-9]{1,11}$"
```

`validate_billedid($billedid)`

Et billedid bliver benyttet til at identificere et billede. Dette id er ligesom ovenstående autogeneret og derfor ser det anvendte regex ud som stack-regex:

```
"^[0-9]{1,11}$"
```

`validate_billednavn($billednavn)`

Billednavn er navnet på det billede som brugeren vælger at uploade, eller det navn han tildeler billedet hvis han ønsker at ændre navnet. Et billednavn må bestå af tal, bogstaver og punktum. Derfor har jeg valgt at mit regex skal se således ud:

```
"^[a-z0-9.]{1,100}$"
```

Man kunne vælge at udspecificere det mere ved at sige at navnet på et billede skal ende på et punktum og så 3 karakterer. Det har jeg dog valgt ikke at gøre da jeg her benytter en anden form for kontrol. (Se fileprocess.php)

`validate_relation($relation)`

En relation er den variable der benyttes til at bestemme om en bruger er ejer af en billedstak. Til dette benyttes 0 eller 1. Jeg har dog valgt at medbringe 2 også i tilfældet af en senere udvidelse. Det regex jeg anvender ser således ud:

```
"^[0-2]{1}$"
```

`get_stack_relation($kundenummer, $stack)`

Denne funktion bliver benyttet i album.php til at kontrollere at den stack som er blevet valgt virkelig har et tilhørsforhold til brugeren. Man kan forestille sig at en ondsindet bruger ville prøve at ændre dette nummer for at tiltvinge sig adgang til billedmapper, som han ikke har rettigheder til at se.

dblogin.php

Dblogin.php, indeholder funktioner til brug ved login og kontrol gennem systemet. Den indeholder 2 funktioner som er beskrevet nedenfor.

`get_user($user, $pass)`

Tager $user og $pass som argument. Den benyttes når en bruger logger ind i systemet. Hvis username og password findes som en række i tabellen loginfo, vil det sige at begge

dele er indtastet korrekt. Den vil så returnere kundenummer, username, password og userlevel. Disse vil blive skrevet til den cookie, der vil blive anvendt gennem resten af systemet

`check_login($kundenummer)`

Denne funktion vil blive kaldt hver gang en bruger trykker på et link i systemet, efter han / hun er logget ind. Den vil returnere username, password og userlevel, til det kundenummer, som bliver anvendt som argument.
Disse vil blive anvendt for at kontrollere at en bruger stadig er logget ind og anvender det korrekte kundenummer.

dbmanagement.php

Indeholder alle funktionerne som vil blive benyttet til management delen af systemet. F.eks. Hver gang en bruger vil dele eller slette et billede, vil han benytte funktioner fra denne fil. Denne klasse er en af de centrale dele af systemet og indeholder mange af de funktioner som er grundlæggende for systemet.

`function get_usernames()`

Når en bruger vil dele en mappe med en anden bruger, benytter han en dropboks med alle brugernavne i systemet.
Denne funktion leverer alle brugernavne til den dropboks. Det bør dog bemærkes at dette ikke er hensigtsmæssigt i et større system, så der er man nødt til at finde på en anden løsning. (Se fejl og mangler)

`share($owner_kundenummer, $reciver_kundenummer, $stack)`

Denne funktion sørger for at dele en mappe med en anden bruger. Den tager owner_kundenummer, reciever_kundenummer og stack som argumenter. Den kontrollerer så om owner_kundenummeret virkelig er ejer af billedmappen. Er dette tilfældet vil den tilføje

billedstakken med nummeret ”stack” til modtageren (reciever_kundenummer). Da der ikke er implementeret fejl koder i denne del, vil den returnere til:

```
header("Location: /PHP/management.php");
```

Dette vil den gøre også selv om det ikke lykkes at dele mappen. (se management.php for mere information)

`function remove($kundenummer, $stack)`

Denne funktion bruges af en bruger, der har fået tildelt en mappe, som han ikke længere ønsker at have adgang til. Den kontrollerer at brugeren har fået mappen delt og at han ikke er ejer. Er dette tilfældet vil den fjerne delingen i tabellen ”kundetilbilled”. Der er ligeledes ikke implementeret fejl håndtering i denne del, derfor vil den ligeledes returnere til

```
header("Location: /PHP/management.php");
```

`delete($kundenummer, $stack`

Denne funktion bruges til at slette en mappe. Den tager kundenummer og stack som argumenter, hvor stack er den mappe der ønskes slettet. Funktionen kontrollerer først at brugeren ejer den mappe, han ønsker at slette. Er dette tilfældet, vil den fjerne alle rækker i ”kundetilbilled” der indeholder den pågældende billedstak (stack). Den vil efterfølgende fjerne rækken fra billedstak og til sidst fjerne alle rækker i ”staktilbillede” der indeholder denne billedstak.
(se evt. sletning af billeder under funktionalitet.)

`shared_stacks($kundenummer)`

Denne funktion returnerer alle mapper og kundenumre som en bruger deler med andre.

`get_username($kundenummer)`

Returnerer et brugernavn til et givet kundenummer. Denne funktion bruges sammen med shared_stacks så en bruger kan se hvilke mapper han deler med hvem.

`Unshare($owner_kundenummer, $reciver_kundenummer, $stack)`	Denne funktion bruges til at fjerne en deling af en mappe med en given bruger. Der er ikke implementeret fejlhåndtering til denne funktion så den returnerer til : `header("Location: /PHP/management.php");`

dbpicture.php

Denne fil indeholder funktioner til visning af billeder. Det er denne klasse der indeholder funktionerne der tager sig af paging. Til at page bruger jeg 2 funktioner: get_billedid_paged og pager. De tager begge variablen $rows. Denne er ikke blevet implementeret, men sættes default i album.php til 1. Det er denne variable, der styrer hvor mange billeder der skal være pr side.
(Pagingen bygger på følgende script fundet på nettet se under referencer for link)

`get_billedid_paged($stack, $rows, $page)`	Som en del af pagingen, tager denne funktion en given stack og returnerer alle billedid'er der tilhører denne billedstak og som er inden for det fastsatte limit.
`pager($rows, $stack, $pageNum)`	Denne funktion tager sig af selve pagingen den sørger for at skrive selve navigations menuen og de tilhørernde links.
`get_billedid($stack)`	Denne funktion tager en stack som argument og returnerer alle tilhørende billedid'er. Præcis som ovenstående dog uden paging
`get_billede($billedid)`	Denne funktion tager et billedid og returnerer Billednavn, Beskrivelse og Link.

dbpictureedit.php

dbpictureedit.php indeholder en enkelt funktion. Denne funktion sørger for at opdatere et billede, hvis en bruger ønsker at ændre i navn eller beskrivelse af et billede. Denne klasse bliver kaldt af en

form, som benytter sig af den globale $_POST variable. Det er derfor vigtigt at kontrollerer alle vores variabler efter vi har hentet dem:

```
$billednavn = $_POST['billednavn'];
$beskrivelse = $_POST['beskrivelse'];
$stackid = $_POST['stack'];
$edit_billed = $_POST['edit'];
```

Efter at variablerne er blevet valideret, kontrollerer funktionen, at brugeren har ejerrettigheder over billedet. Er dette tilfældet vil Billednavn og Beskrivelse blive opdateret og bruger vil blive returneret til:

```
header("location: /PHP/pictureedit.php?stack=$stackid&edit=$edit_billed&rtn=1")
```

Det er værd at bemærke at variablen rtn er til fejlhåndtering.

dbregister.php

Denne klasse indeholder alle funktioner til at oprette en person i databasen som en bruger. Den består af 2 funktioner som vil blive beskrevet nedenfor.

`create_user($fornavn, $efternavn, $addresse, $postnummer, $by, $telefonnummer, $email , $user, $pass, $repass)`	Tager sig af fejlhåndteringen, validerer alle if sætningerne kalder den nedenstående funktion.
`make_user($fornavn, $efternavn, $addresse, $postnummer, $by, $telefonnummer, $email , $user, $pass)`	Denne funktion tager sig af oprettelsen af brugeren i databasen.
`check_username($user)`	Denne funktion bruges til at kontrollere om et brugernavn er i brug i forvejen.

Det bør bemærkes at e-mail adressen ikke bliver valideret, da min database ikke vil accepterer @. Derfor er det pt. ikke muligt at skrive en e-mail adresse ned i databasen. Derfor er e-mail validering slået fra, så den ikke kræver en e-mail adresse skrevet ind i feltet for at få det valideret.

dbupload.php

dbupload.php indeholder de funktioner der bliver anvendt i forbindelse med upload af en fil. Samt et par funktioner til designet af upload.php.

`create_stack($kundenummer, $stackname, $beskrivelse)`	Denne funktion opretter en ny mappe / billedstak, og sætter brugeren med det givne kundenavn som ejer af den nye billedstak.
`create_billed($billednavn, $beskrivelse, $link, $stack)`	Denne funktion bruges sammen med fileprocess.php til at uploade en fil til serveren. Denne funktion sørger for at billedet bliver oprettet korrekt i databasen.
`user_stacks($kundenummer, $relation)`	Denne funktion returnerer en brugers billedstakke med en given relation.
`get_stackinfo($billedstak)`	Denne funktion returnerer Billedstaknavn, Beskrivelse og dato for en given billedstak.

fileprocess.php

Fileprocess.php er den fil der bliver kaldt, når en fil skal uploades til serveren. Den indeholder en masse kontrol checks, som en fil skal igennem inden den bliver uploadet til serveren. Bl.a. er det her jeg kontrollerer om den fil, der bliver uploadet har en tilladt filtype. Dette sker ved hjælp af følgende kode:

```
$valid = array (".jpg",".gif",".png",".bmp");
$type = strtolower(strstr($filename, '.'));
if (!in_array($type, $valid))
{
        $uploadprogress = "3"; //invalid file type
}
```

Herved sikre vi os, at der kun bliver uploadet filer af godkendte formater til vores webserver. Jeg har implementeret fejlhåndtering i upload.php som denne funktion returnerer til, ved hjælp af følgende:

```
header("location: /PHP/upload.php?stack=$stackid&file=$uploadprogress");
```

hvor $file er den variable der indeholder den returnerede kode og $uploadprogress er fejlkoden. Det bør dog bemærkes at $uploadprogress = 1 er tegn på en succesfuld upload af filen.

index.php

Dette er den første side en bruger møder. Den indeholder en loginboks og et link til en side, hvor personen kan oprette sig som bruger. Denne side er beregnet til, i en udvidelse, at give en kort introduktion til ideen bag systemet. Samt forklare funktionaliteten af systemet. Med andre ord være et blikfang for en person, som er en potentiel bruger. Selve index.php benytter sig af funktioner fra dblogin.php, til at håndtere login processen. For nærmere forklaring se afsnittet ”login” under funktionalitet.

main.php

Dette er hovedsiden i systemet, når en bruger er logget ind. Den fungerer som oversigt for hele systemet. Alle links til andre sider er inkluderet i menuen til venstre (leftnav). Her køres brugerkontrol på hvert link inden brugeren sendes videre til den ønskede side. Dette forgår ved hjælp af et pathid, et eksempel herpå kan ses nedenfor:

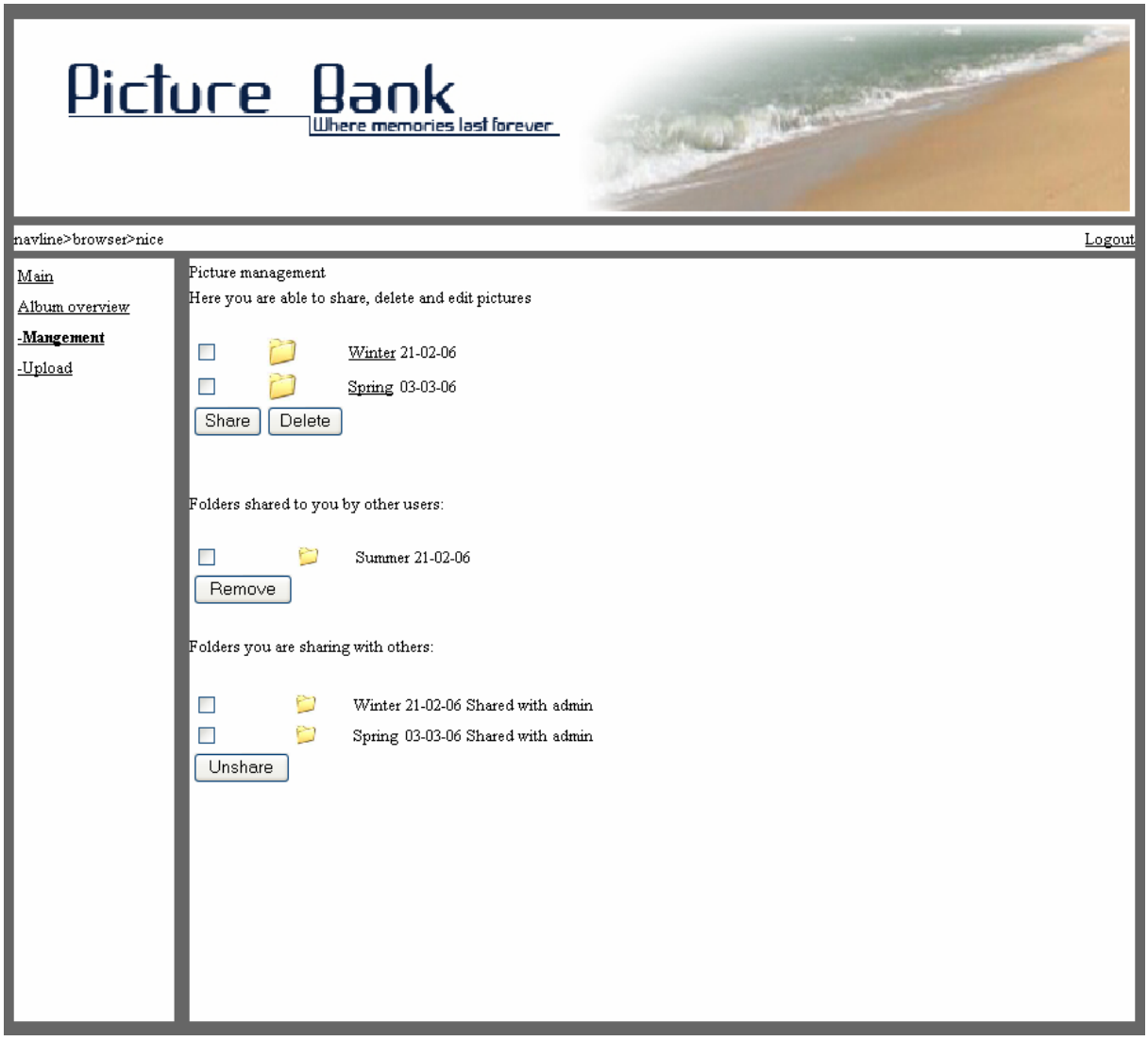
```
elseif($pathid == 3)  //Picturemanagement
{
if($_COOKIE[user] == md5($login_data[Username])&& md5($_COOKIE[pass])
== $login_data[Password])
           {
                      header("Location: PHP/management.php");
           }
           else
           {
                      header("Location: /main.php?pathid=0");
           }
}
```

Hvis username og password ikke stemmer overens, vil brugeren blive sendt tilbage til main med pathid=0. Dette vil logge ham ud af systemet. Derved sikrer jeg, at vores bruger hele tiden er den, som han udgiver sig for at være.

management.php

Dette er den endelige frontend løsning, til hele billed management delen af mit system. Det er herfra en bruger kan slette, dele og redigere i sine billeder og delinger. Den benytter sig af mange funktions kald til at håndtere visningen af de enkelte mapper. Men alle hoved funktionerne vil være

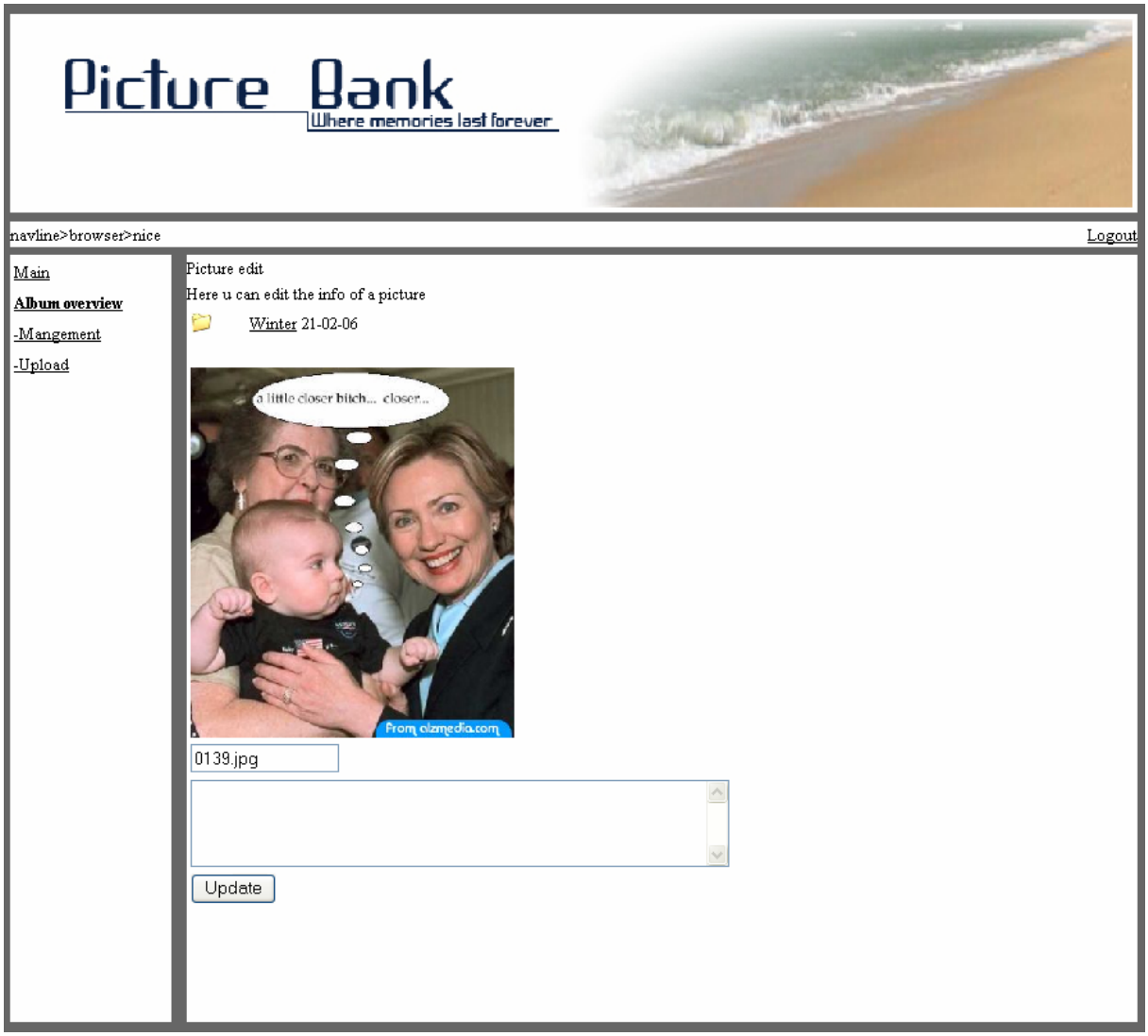
at finde i dbmanagement.php. Da der sker mange ting på denne side, vil den switche på en række variabler, alt efter hvilken funktionalitet brugeren ønsker at anvende. Der er ikke i denne prototype implementeret fejlhåndtering af eventuelle fejl på denne side. Den endelige side er kommet til at se således ud:



pictureedit.php

pictureedit.php, fungerer som frontend del for en bruger, der ønsker at redigere data til et givent billede. Den indeholder en funktion, der viser det billede man skal til at redigere, og en form hvor man kan ændre navn og beskrivelse. Er dette allerede indtastet, vil disse informationer også være at finde i formen. Formen kalder dbpictureedit.php, som tager sig af selve updateringen, når der bliver

trykket på update. Der er implementeret fejlhåndtering, af det returnerede svar fra dbpictureedit.php. Den endelige implementering af denne er kommet til at se ud på følgende måde:



register.php

Denne side indeholder formen, som en person skal benytte, hvis han eller hun ønsker at blive oprettet en bruger i systemet. Den indeholder fejl beskeder, som den vil bringe sammen med en ny register form, f.eks. en bruger har indtastet data som ikke er blevet valideret korrekt. Dette kunne f.eks. være bogstaver i postnummeret, som vil medføre at bruger er nødt til at prøve igen.

upload.php

Indeholder den grafiske brugerflade til upload af billeder. Den består af to dele, som switcher på en if sætning som ser ud som følgende:

```
if($stackid == "0")
{
}
else
{
}
```

Hvad denne if sætning gør, er at switch på bruger har valgt en mappe eller ej. Har brugeren ikke valgt en mappe (dette er tilfældet når han trykker på linket upload ude i menuen til venstre) vil han blive præsenteret for alle de mapper han har skriverettigheder til, samt en form til at oprette en ny mappe. Når bruger klikker på en mappe, altså vælger hvilken mappe han vil uploade et billede til, vil han blive sendt ned i den anden del af if sætningen. Denne del vil nu vise ham den valgte mappe og indhold samt en form til at vælge et fil upload, og en form der sætter ham i stand til at indtaste et nyt navn til billedet samt en beskrivelse.

index_css.css

Dette er den CSS fil som hele designet bygger på, den indeholder konfigurationen til alle de 's som jeg benytter mig af gennem projektet. For mere forklaring se under design i afsnittet user layer.

Database

Som tidligere beskrevet er databasen normaliseret til 3NF, nedenfor er de sql sætninger som er blevet brugt at lave hver enkelt tabel. Disse kan blive brugt til at genskabe databasen, men de viser også hvilke værdier de enkelte felter kan indeholde.

billede

```
CREATE TABLE `billede` (
  `Billedid` int(12) NOT NULL auto_increment,
  `Billednavn` varchar(50) default NULL,
  `Beskrivelse` varchar(400) default NULL,
  `Link` varchar(100) NOT NULL,
```

```
  PRIMARY KEY  (`Billedid`),
  UNIQUE KEY `Billed` (`Billedid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Billedstak

```
CREATE TABLE `billedstak` (
  `Billedstak` int(11) NOT NULL auto_increment,
  `Billedstaknavn` varchar(100) NOT NULL,
  `Beskrivelse` varchar(400) default NULL,
  `Dato` varchar(8) default NULL,
  PRIMARY KEY  (`Billedstak`),
  UNIQUE KEY `Billedstak` (`Billedstak`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Kundeoplysninger

```
. CREATE TABLE `kundeoplysninger` (
  `kundenummer` int(11) NOT NULL auto_increment,
  `Fornavn` varchar(32) default NULL,
  `Efternavn` varchar(32) NOT NULL,
  `Addresse` varchar(100) NOT NULL,
  `Postnummer` int(4) NOT NULL,
  `Bynavn` varchar(32) NOT NULL,
  `Telefonnummer` varchar(15) NOT NULL,
  `Email` varchar(100) NOT NULL,
  PRIMARY KEY  (`kundenummer`),
  UNIQUE KEY `kundenummer` (`kundenummer`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

kundetilbillede

```
. CREATE TABLE `kundetilbilled` (
  `ID` int(11) NOT NULL auto_increment,
  `Kundenummer` int(11) NOT NULL,
  `Billedstak` int(11) NOT NULL,
  `Relation` int(1) NOT NULL default '1',
  PRIMARY KEY  (`ID`),
```

```
  UNIQUE KEY `ID` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Loginfo

```
CREATE TABLE `loginfo` (
  `Log_id` int(11) NOT NULL auto_increment,
  `Kundenummer` int(11) NOT NULL,
  `Username` varchar(32) NOT NULL,
  `Password` varchar(32) NOT NULL,
  `Userlevel` int(1) NOT NULL default '0',
  PRIMARY KEY  (`Log_id`),
  UNIQUE KEY `Kundenummer` (`Kundenummer`),
  UNIQUE KEY `Username` (`Username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Staktilbillede

```
CREATE TABLE `staktilbillede` (
  `Stakid` int(11) NOT NULL auto_increment,
  `Billedstak` int(11) NOT NULL,
  `Billedid` int(11) NOT NULL,
  PRIMARY KEY  (`Stakid`),
  UNIQUE KEY `stakid` (`Stakid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Test

I denne del af rapporten vil jeg prøve at teste mit system Jeg vil prøve at finde de fejl og mangler jeg har i mit system, så de kan blive vurderet.

Test af register:

Kontrol af brugerinput i registerformen.
I denne test er jeg nødt til at se bort fra e-mail feltet. Grunden til dette er, at jeg ikke er i stand til at gemme en e-mail adresse i databasen. Derfor vil jeg få en fejl hvis jeg forsøger på dette. Derfor er testen ikke rettet mod dette felt. Jeg tester ved at indtaste ikke forventet input i formen for at se om jeg kan fremprovokere en fejl, der ikke er taget højde for i fejlbehandlingen.
Dette har jeg dog ikke været i stand til, da alle variabler bliver kontrolleret ved hjælp af mine tidligere opstillede regular expressions.

Test af login og cookie:

Den første test af login jeg har valgt at foretage, er at tjekke for en mulighed for sql injection. Dette gør jeg ved at indsætte følgende udsagn i henholdsvis user og pass, som er post variablerne fra min login form.

```
'or1=1--
"or1=1--
or1=1--
'or'a'='a
"or"a"="a
') or ('a'='a
```

Ideen bag er at teste om jeg kan få en sql kommando til at returnere et sandt udsagn, hvorved man kan være heldig at blive lukket ind i systemet. Det kunne dog ikke lade sig gøre. Grunden til dette er at user bliver checket i validate_user inden sql sætningen bliver udført. Password derimod bliver krypteret inden det bliver checket så derfor vil sql injection slet ikke virke i dette felt heller.

Til at teste cookien, og dermed kontrol funktionerne til at sikre at en bruger ikke ændrer oplysninger i systemet har jeg prøvet at ændre først kundenummer, hvilket resulterede i, at jeg blev logget ud af

systemet. Efterfølgende prøvede jeg at ændre mit userlevel. Dette ændrede ikke noget og jeg kunne fortsætte min færden gennem systemet. Jeg kunne dog stadig ikke se linket til administrator menuen. Jeg prøvede så at indtaste linket direkte til administrator menuen, hvilket gjorde at jeg blev logget ud af systemet.

Test af billedvisning:

Til at teste billedvisningen vil jeg prøve at uploade et stort billede og se hvordan billedvisningen reagerer. Dette medfører følgende resultat:

Som vi kan se bliver siden ”forskubbet”. Dette er ikke meningen og bør derfor ændres i en udvidelse.

Da billedvisningssiden bruger stack som variable (anført nedenfor) kan det være interessant at se, hvad der sker, hvis jeg ændrer tallet til et andet stack nummer, for at se om jeg kan få adgang til en anden stack på den måde.

```
http://localhost:xxxx/PHP/album.php?stack=26
```

Ved at benytte mig af sql injection sker der ikke noget, da funktionen ikke bliver eksekveret, fordi stacknummeret ikke kommer igennem validerings fase. Efterfølgende prøver jeg at ændre stack nummer til et andet stack nummer, som brugeren har adgang til. Dette sender ham direkte videre til den pågældende stack. Dette er ok og det vil jeg ikke betragte som en fejl. Da jeg efterfølgende prøver at indtaste et stack nummer, som brugeren ikke har tilknytning til, bliver jeg logget ud af systemet. Grunden er at jeg kører et tjek på om brugeren har tilknytning til stacken. Har han ikke det vil han blive logget ud af systemet.

Denne side benytter sig af en variabel mere, nemlig page:

```
http://localhost:xxxx/PHP/album.php?stack=26&page=9
```

Der kører ikke validering på denne variable som systemet er nu. Dette åbner mulighed for sql injection. Jeg har dog ikke været i stand til at bedømme det mulige skadeomfang, men vurderer det til at være af mindre risiko. Der bør dog i en senere udvidelse køres et validerings check på denne variabel.

Test af billede management:

Ved deling, sletning og unshare af et billede testes der også. Funktionerne benytter sig af en hidden post variable. Denne Post variabel (stack), vil ikke kunne blive benyttet til et eventuelt sql injection angreb, da der køres validering af selve variablen. Det vil heller ikke være muligt direkte at tiltvinge sig adgang til andre mapper da der kører kontrol af tilhørsforholdet på den givne stack variabel. Det bør bemærkes at dette tilhørsforhold ikke kontrollerer selve ejerforholdet. Derved kan en bruger med et tilhørsforhold til en stack kunne ændre, slette eller på anden vis misbruge dette.

Derfor bør der bestemt blive implementeret en løsning på dette i en fremtidig udvidelse af systemet.

Ydermere bør det bemærkes, at denne stack variabel kun kan indeholde et stackid ad gangen. Det er derfor ikke muligt at dele, slette eller unshare mere end en mappe ad gangen. Forsøger man på dette vil det kun være den sidst valgte mappe, der vil blive påvirket.

Endelig kan bemærkes, at deler man den samme mappe til en person flere gange, vil alle disse delinger blive fjernet hvis man unshare en af dem.

Test af billede upload:

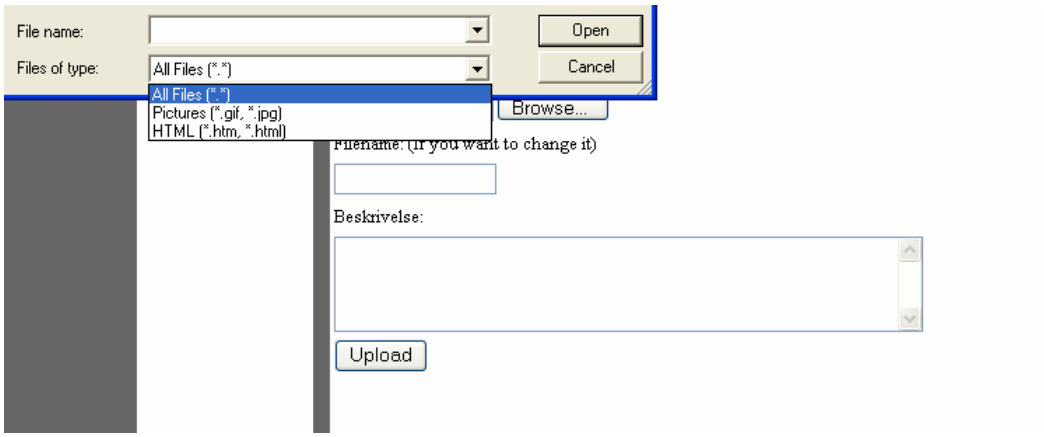
Den første funktion jeg støder på, i forbindelse med testen af upload delen af mit system, er ”opret en ny mappe”. Denne funktion kører validering på både mappe navn (stak navn) og på beskrivelse. Den vil dog ikke oprette en mappe uden af begge dele er indtastede. Hvis dette ikke er tilfældet kommer den med en fejlmeddelelse:

```
Failed to create new folder
```

Når man har indtastet begge dele og trykker på ”opret”, vil mappen blive oprettet. Den vil dog ikke blive vist, før man trykker på upload linket igen, da den ikke opdaterer selve siden før. Dette kan man forstille sig opdateret i en fremtidig version. Det udgør dog ikke en potentiel sikkerhedsrisiko og er derfor af mindre vigtighed.

Efter at have valgt hvilken mappe, eller stack om man vil, som jeg ønsker at uploade et billede til, kommer jeg videre til den egentlige upload del. Selve valget af mappe kører igen på en hidden post variabel, der indeholder den valgte stack, som jeg kører kontrol af tilhørsforhold og validering på, inden den bliver benyttet.

Jeg kan nu browse efter en fil i en standard browser, men denne har dog flere valg muligheder end den burde have og viser kun nogle af de formater, som mit system tillader. Nedenstående billede illustrerer dette:



Selve min upload procedure tillader filer af følgende typer at blive uploadet:

```
$valid = array (".jpg",".gif",".png",".bmp");
```

Dette inkluderer ikke html filer, men tillader derimod .png og .bmp filer. Jeg kan forestille mig at dette bliver ændret i senere udvidelse.

Jeg har ikke yderligere været i stand til at finde flere fejl ved upload delen, da der kører validering på mine variabler og indtastede værdier.

Fejl, mangler og mulige udvidelser

Blandt andet med udgangspunkt i mine tests, vil jeg i følgende afsnit opsummere de fejl jeg har i mit system. Ydermere vil de ting som jeg gerne ville have implementeret, men som jeg ikke nåede, blive nævnt.

- I billedvisningen bliver billederne kun vist ved en størrelse på 60 % af deres originale størrelse. Samtidig vil store billeder skævvride siden. Jeg kan forestille sig, at billederne i en fremtidig version, vil blive tegnet i en boks. Denne boks skal have en prædefineret størrelse. Hvert billede skulle så tegnes inde i denne boks og derved vil de ikke skævvride min
- Efter en bruger har ”slettet” et billede fra serveren, vil oplysningerne om dette billede stadig være at finde i tabellen ”billede”. Selve billedet vil også stadig ligge på serveren. Det er derfor nødvendigt med en måde at holde styr på disse ”slettede” billeder. En mulig løsning kan være at krydsreferere billedid fra tabellen ”billed” med billedid fra tabellen ”staktilbillede”. Derved vil man kunne finde alle de billeder, der ikke er tilknyttet en stak og på den måde afgøre hvilke billeder man kan slette.
- Den måde jeg på nuværende tidspunkt håndterer valg af brugere, der skal have tildelt en mappe, vil ikke være hensigtsmæssigt i et størrer system. Jeg kan forestille mig, at jeg implementerer en søge funktion eller en friends liste, eller begge dele.
- Af sikkerhedsmæssige årsager vil det være nødvendig at rykke placeringen af billeder væk fra selve web-serveren. Dette skaber et alvorligt sikkerhedshul som det er lige nu. Jeg har dog valgt at bibeholde dette, fordi det har lettet min implementering. Men understreger endnu engang, at det er strengt nødvendigt at denne lagring finder sted udenfor selve web-serveren. Løsningen til dette problem er at ændre stien, som billederne bliver skrevet til. PHP er fuldt ud i stand til at gøre dette, og min database vil også kunne håndtere dette som den er nu.
- Der mangler at blive valideret på min paging variabel. Det vil være fornuftigt at implementere, da det ellers åbner mulighed for sql injection.

- Jeg er ikke i stand til at gemme et @ i min sql database, som det er lige nu. Dette skal indarbejdes i en mulig udvidelse.
- Det er, som systemet er indrettet nu, muligt at vælge flere mapper til deling, sletning eller unsharing. Det er dog kun det sidst valgte, der vil blive sendt videre til behandling. Enten skal jeg udvide systemet til at kunne håndtere flere samtidige valg, eller også skal jeg sætte en begrænsning, så kun en mappe kan vælges ad gangen. Her vil den sidste løsning være klart lettest at implementere.
- I øjeblikket er der en mulighed for at dele den samme mappe flere gange med en given bruger. Dette bør ikke være muligt i senere udvidelse.
- Når jeg opretter en ny mappe, opdaterer siden ikke, så jeg kan se den nyoprettede mappe , før jeg trykker på upload linket igen.
- Felterne i fil browseren til upload af billeder skal have de rigtige fil formater som valgmuligheder.
- Mine paging funktioner åbner nu mulighed for at page flere billeder per side ved hjælp af rows. Antallet af rows vil svare til antallet af billeder, der vil være på en side. I en fremtidig udvidelse vil det være oplagt at benytte sig af denne mulighed.
- Det vil være hensigtsmæssigt at implementere en tabel der indeholder kundenummer og billedeid, da denne senere vil kunne blive brugt til at forbinde et bruger med et bestemt billede i tilfælde af at dette bliver nødvendigt.

Konklusion

Igennem hele projektet har jeg forsøgt at holde sikkerhed og simpelt funktionelt design højt. Dette har været sværere end det umiddelbart lyder da der er mange ting at tage hensyn til, når man designer et system som Picturebank. Forholdet mellem de enkelte delelementer i systemet spiller en stor rolle. Man skal derfor hele tiden huske, at være konsekvent i sin validering af variabler da man ellers hurtigt åbner sikkerhedshuller i sit system. Ligeledes er det vigtigt hele tiden at følge en bruger gennem systemet og kontrollere alle de inputs, han kan lave til systemet. En af måderne jeg har gjort dette er ved nøje at kontrollere hans tilhørsforhold til alle mapper.
Jeg har valgt at lagre filerne i mapper på harddisken i mappestruktur genereret af mit system. Det er i denne forbindelse yderst vigtigt, at denne mappe struktur bliver lavet udenfor web-serveren da det ellers vil medføre en alvorlig sikkerhedsbrist i systemet.
Jeg har ligeledes valgt ikke at give en bruger tilladelse til fysisk at slette billeder fra harddisken. Dette er der to grunde til:

1. Jeg mener det er uhensigtsmæssigt at implementerer en funktion der kan slette filer på harddisken og efterfølgende lade den modtage input fra brugerniveau uden særlige sikkerheds foranstaltninger.
2. Da der er en reel fare for at et system som dette vil kunne blive brugt til at udveksle billeder af tvivlsom karakter, vil det være uhensigtsmæssigt hvis en bruger kan fjerne dem og på denne måde slippe for eventuel retsforfølgelse

Jeg har gennem hele udviklingen ligeledes forsøgt at bygge dette projekt op om et simpelt men funktionelt design. Dette har jeg prøvet at opnå ved at opbygge en navigations form, som lægger sig tæt op af windows egen. Derved håber jeg at en bruger vil kunne finde genkendelige elementer og derved hurtig blive fortrolig med systemet.

Referencer

Til php hjælp har jeg benyttet mig af nedenstående side:

http://www.php.net

Til CSS vejledning har jeg benyttet mig af:

http://www.w3schools.com/css/

Til at implementerer paging benyttede jeg mig af følgende, som jeg dog selv redigerede så den passede til mig behov.

http://www.php-mysql-tutorial.com/php-mysql-paging.php