LogPlan

Teknikfagsrapport

Udarbejdet af:
Emmanuel Hategekimana, Erik B. Christensen, Marcus Christiansen og William A. Carlsen
ZBC Vordingborg HTX
Henrik Bruhn

Indledning

Projektarbejde er en stor del af en HTX-uddannelse og det handler ikke kun om at lave et produkt eller at dokumentere det. En stor del af projektarbejde er også administration og organisering, men det er ikke noget studerende typisk har nogen erfaring eller dybdegående undervisning i. Og det kan medføre, at projekter går dårligt, ikke på grund af dårligt arbejde, men på grund af dårlig planlægning. Og derudover er projektstyring kernestof i både Teknikfag og Teknologi[1], så det kan forbedre ens karakterer, hvis man inkluderer dokumentation af ens projektstyring, f. eks. tidsplan og logbog, i ens rapport.

Problemformulering

Derfor kunne værktøjer til at assistere studerende i disse dele af projektarbejdet være en god idé. Bedre projektstyring hjælper både med at lave et bedre produkt og med at opfylde fagenes kernestof, hvilket begge kan betyde bedre karaktere. Og dem, som allerede har gode karaktere, kan brude mindre tid på deres projektstyring for samme kvalitet.

Hvordan kan man hjælpe HTX-studerendes projektstyring og dokumentation heraf vha. værktøjer til at gøre projektstyring og -dokumentering nemmere?

Brugsscenarier og målgruppeanalyse

Brugere af vores program er HTX-studerende i deres projektarbejde. HTX har projekter i mange af deres fag. Dette inkluderer: Teknologi, Teknikfag, Kommunikation og it, og Programmering. I teknologi og teknikfag er projektstyring et relevant fagområde og eleverne bliver dermed undervist i det. Men for at undersøge hvorvidt eleverne også bruger projektstyring, er man nødt til at lave en brugerundersøgelse.

Brugerundersøgelse

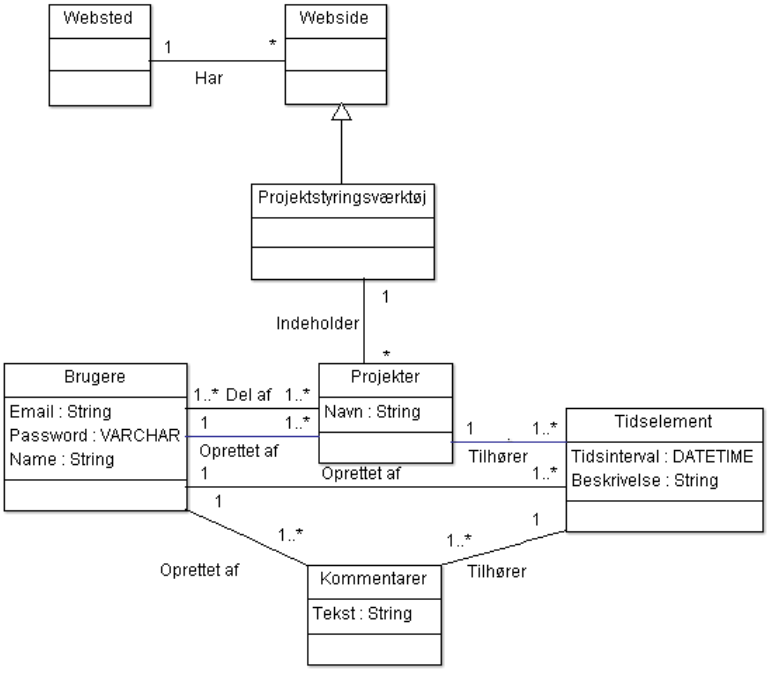
Vi undersøgte denne problemformulerings aktualitet vha. et interview med en 3. g'er. Til spørgsmålet "Hvor ofte laver du dokumentation over jeres gruppearbejde? F. eks. logbog eller tidsplan." svarede interviewpersonen, at han i hans gruppes projekter ikke laver logbog så meget, men de har nogle gange haft fat i tidsplanlægning. Men til gengæld bliver tidsplanerne ikke brugt så meget. Derefter spurgte vi hvorvidt den sidste periode i hans projekter er mere presset og stresset end resten. Hertil svarede han ja. Vores sidste spørgsmål var hvorvidt han mente, at projektstyring er på lig fod med andre fagligheder i et fag. F. eks. om evnen til at programmere lige så vigtig i et programmeringsprojekt som evnen til at styre projektet. Hertil svarede han, at det er vigtigere at bruge sin tid rigtigt, da ens programmering evner kun kan blive brugt, hvis man har tid til det.

[1] https://www.uvm.dk/gymnasiale-uddannelser/fag-og-laereplaner/laereplaner-2017/htx-laereplaner-2017

Man kan altså konkludere at mindst en gruppe på HTX kunne bruge et redskab der kunne hjælpe dem med at lave deres projektstyring og deres dokumentation af deres projektstyring.

Overordnede begreber

Der skal laves et projektstyringsværktøj, og det skal ligge på et websted. Webstedet består af flere forskellige websider, som hver især står for en del af værktøjet. Brugere kan med værktøjet oprette projekter, og tilføje andre brugere til sine projekter. Alle brugere som er del af et projekt kan i projektet oprette tids-elementer. Disse elementer skal indikere tidspunkter brugerne skal arbejde på deres projekt i. Derfor gives tids-elementerne et tidspunkt/dato, og en beskrivelse, der fortæller hvad brugerne har planlagt at lave på projektet i dette tidsinterval. Derudover kan brugerne knytte kommentarer til tids-elementerne, hvilket kan bruges til at føre logbog over hvad der faktisk blev lavet i tidsintervallet.



Figur 1: Diagram over sammenhæng mellem begreber

Beskrivelse af begreber

Vi har her udarbejdet en tabel over hvilke navneords-begreber vi bruger i forhold til vores produkt, og beskrevet disse, og en tabel over hvilke udsagnsords-begreber vi bruger i forhold til vores produkt, og beskrevet disse.

Navneord	Betydning

Websted	Webstedet er den webadresse hvor vores projektstyringsværktøj ligger på
Webside	Webstedet består af websider som hver indeholder en del af vores projektstyringsværktøjer.
Projektstyringsværktøjet, eller “LogPlan”	Selve det produkt vi udarbejder. Det ligger på tværs af de websider der er på webstedet.
Projekt	Projekter er noget som brugere kan oprette med projektstyringsværktøjet, og bruges til at sammenkoble hvilke brugere der arbejder sammen, og hvilke tids-elementer der hører til sammenkobling.
Tids-element	Tids-elementer hører til et projekt, og beskriver en tidsperiode. Dermed har tids-elementer et tidsinterval, men de har også en beskrivelse, der beskriver hvad brugerne har planlagt at lave på projektet i tidsintervallet
Bruger	Er den person som opretter en konto på vores projektstyringsværktøj.
Kommentarer	De tekster brugere tilføjer til tid-element for at beskrive hvad der faktisk skete i tidsintervallet. En slags logbog

Udsagnsord	Betydning
Tilføje	Det at brugere kan tilføje andre brugere til projekter som brugeren er en del af.
Oprette	Brugere opretter projekter i opret projekt, når de er logget ind.
Kommentere	Brugere kan kommentere tids-elementer i projekter som de er en del af.

Løsningens arkitektur

LogPlan skal kunne køre på et standard webhotel med Apache 2.4.29, PHP 7.3.8 og MariaDB 10.4.6.

Udviklingsplan

#	Fase	Subfase	Indhold	Flade
1	Login og registrer	- Opret bruger - Login til bruger	- Brugeren kan oprette sig selv som bruger - Brugeren kan logge ind efter de har oprettet sig	- Frontend - Frontend
2	Projekter	- Opret projekter	- Brugeren kan oprette en eller flere projekter	- Frontend
3	Kooperative projekter	- Tilføj andre brugere til projekter	- Brugeren kan tilføje andre til sine projekter, og de andre kan se alle projekter som de er en del af	- Frontend
4	Tidsplan	- Tilføje tids-elementer til projektet	- Brugeren kan tilføje elementer med mål og tidsperiode	- Frontend
5	Logbog	- Skriv logbogskommentarer - Se logbogskommentarer	- Brugere kan skrive logbogkommentarer inde på hver af tidsplanselementerne - Brugere kan se hvem der har skrevet, hvilke kommentarer	- Frontend - Frontend

Overordnede behov og krav

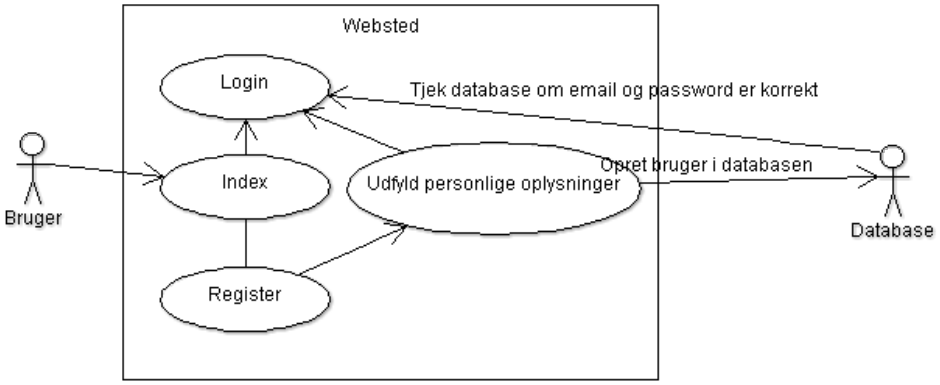
Der er følgende krav til vores produkt:

- Kan visualisere tids-elementer i en tidsplan
- Brugere skal kunne være en del af flere projekter
- Der skal være mulighed for at flere brugere er en del af samme projekt
- Brugere kan se alles kommentarer i projekter som de er en del af, men kun kunne redigere sine egne

Beskrivelse af faser

Iteration 1, Opret og login

Den første iteration af vores websted har bare login og registrering. Vi har valgt at starte med dette, fordi at den måde vi har lavet databasen på, krævede at der var brugere til at lave projekter og tilføje andre brugere til dem.



Figur 1: Use-case diagram over første iteration.

Use case 1: Registrer bruger

Formål:
Brugeren af webstedet kommer ind på index.php, og navigerer til registreringssiden, register.php. Brugeren udfylder formen på websiden, og opretter en bruger, som bliver indsendt til databasen.

Aktører:
Der er to aktører i denne iteration, en bruger på webstedet, og databasen, som agerer ved hjælp af requests eller inputs fra webstedet.

Startbetingelser:
Brugeren har åbnet index.php.

Beskrivelse:

Bruger	**Database**
Klikker på “Register”	
	Åbner register.php
Udfylder felterne på siden og klikker på “Create”	
	Opretter en bruger i databasen, der har de

	informationer som brugeren har tastet ind

Slutbetingelse:
Databasen er opdateret med den nye bruger som er oprettet.

Use case 2: Login

Formål:
Brugeren af webstedet har lige oprettet en bruger på register.php, og er dermed blevet sendt til login.php. Her logger brugeren ind.

Aktører:
Der er to aktører i denne iteration, en bruger på webstedet, og databasen, som agerer ved hjælp af requests eller inputs fra webstedet.

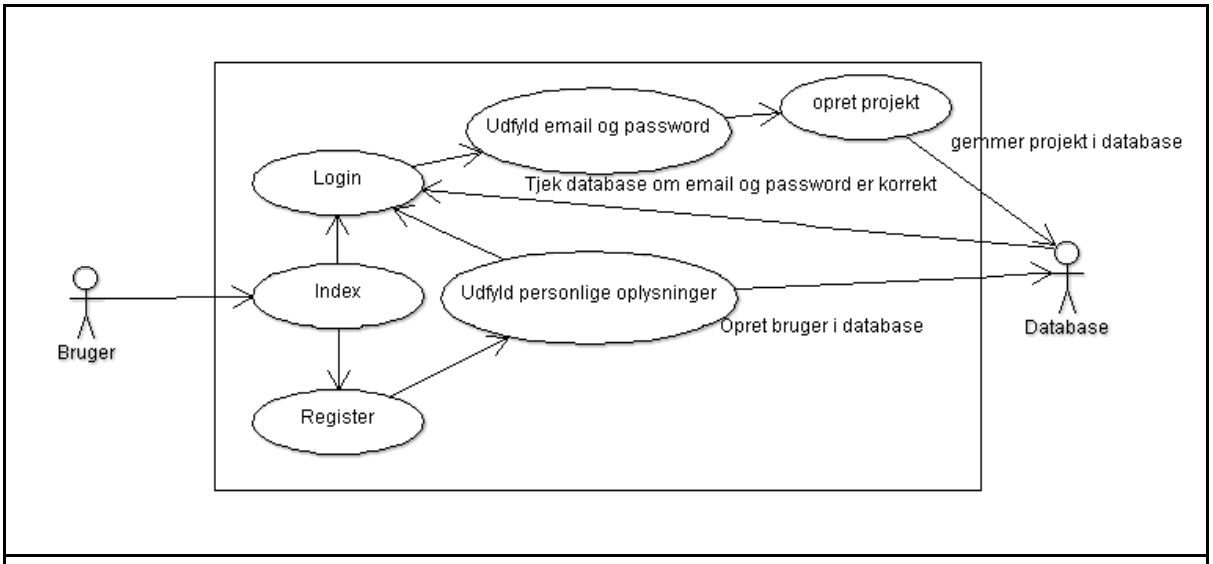
Startbetingelser:
Brugeren har oprettet en bruger på register.php.

Beskrivelse:

Bruger	**Database**
Udfylder den korrekte mail og adgangskode, og klikker på “Login”	
	Tjekker om der er en bruger i databasen med den korrekte mail-adresse, og ser derefter om den indtastede adgangskode matcher den hashede adgangskode i databasen. Hvis dette er tilfældet sendes man til siden main.php.

Iteration 2, opret projekter

Anden iteration er implementeringen af projekter. Dette omhandler hovedsageligt om at give brugeren mulighed for at oprette projekter efter de er logget ind.



Figur IT2: Usecase-diagram over anden iteration.

Use case 1: Opret projekt

Formål:
Der skal selvfølgelig også være et formål ved at logge ind. Og det er blandt andet at lave projekter. Projekter er grundlaget for resten af projektets funktioner, men gør i sig selv ikke noget.

Aktører:
Der er 2 aktører. Brugeren, der opretter projekterne, og databasen, der indsætter projektet i databasen.

Startbetingelser:
Brugeren er logget ind

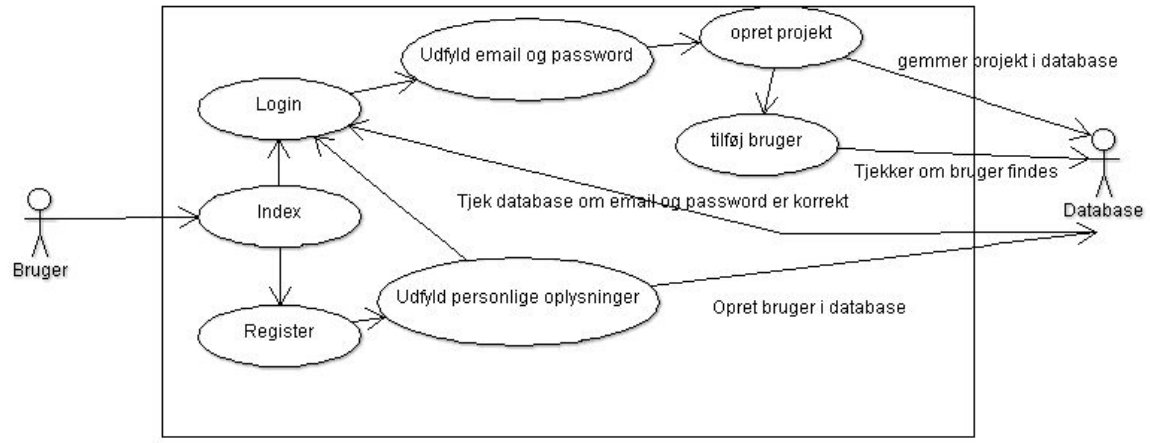
Beskrivelse:

Bruger	**Database**
Brugeren indtaster projektets navn	
	Databasen opretter projektet i databasen

Slutbetingelse:
Brugeren har oprettet et projekt

Iteration 3, kooperative projekter

Tredje iteration er udvidelse af projekter. Dette omhandler at give medlemmer af et projekt mulighed for at tilføje andre brugere. Projektets opretter er automatisk medlem af projektet.



Figur IT3: Usecase-diagram over tredje iteration.

Use case 1: Tilføj andre brugere til projekter

Formål:
Projekter giver kun mening, hvis der også er gruppearbejde og det kan kun ske hvis der er mere end en i projektet, så denne iterations formål er at tilføje andre til en eksisterende projekt

Aktører:
Der er 2 aktører. Brugeren, der anmodere om at tilføjelse af anden bruger, og databasen, der tjekker om den anden brugeren eksisterer og tilføjer brugeren til projektet

Startbetingelser:
Begge brugere og registreret og et projekt er oprettet

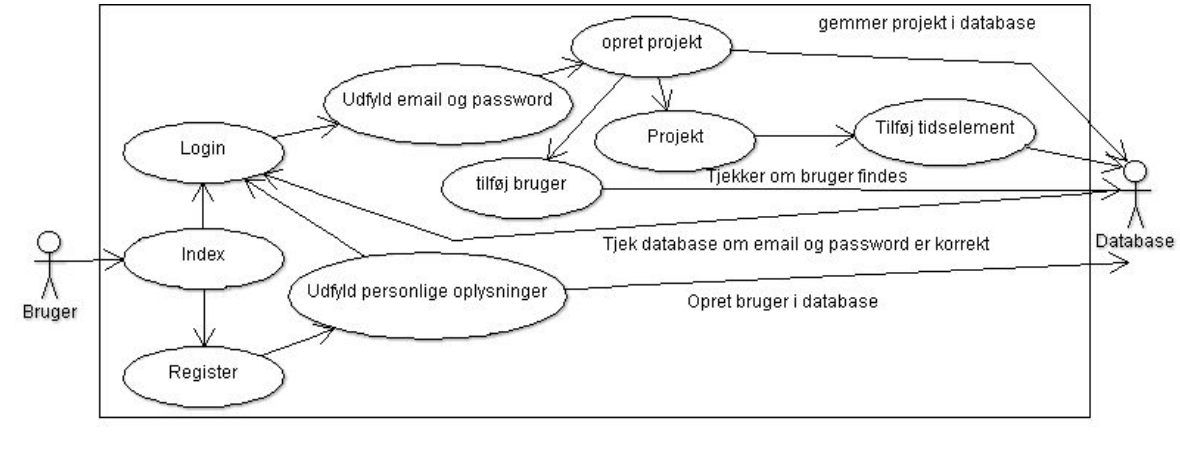
Beskrivelse:

Bruger	**Database**
Brugeren indtaster anden brugers navn	
	Databasen tjekker om brugeren eksisterer
	Databasen tilføjer anden bruger til projektet

Slutbetingelse:
Bruger nr. 2 er tilføjet til projektet.

Iteration 4, Tidsplan

Fjerde iteration er endnu en udvidelse af projekter. Denne omhandler implementeringen tidselementer.



Figur IT4: Usecase-diagram over fjerde iteration.

Use case 1: Tilføj andre brugere til projekter

Formål:
Indtil videre har vores produkt ikke haft nogen aktuel funktion, der kunne hjælpe med projektstyring. Men med tilføjelsen af tidselementer kan vores produkt faktisk bruges til noget.

Aktører:
Der er 2 aktører. Brugeren, der anmoder om at tilføje tidselementer til et projekt og databasen, der gør det.

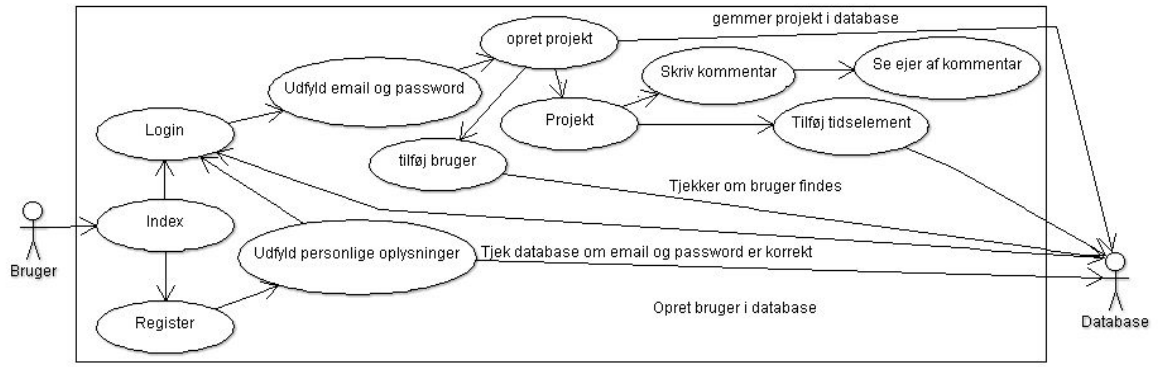
Startbetingelser:
Der er et projekt og en bruger

Beskrivelse:

Bruger	**Database**
Brugeren anmoder om at tilføje tidselementer til et projekt	
	Databasen tilføjer de tidselementer til projektet.

Slutbetingelse:
Tids-elementerne er blevet tilføjet

Iteration 5, Logbog



Figur IT5: Usecase-diagram over femte iteration.

Use case 1: Skriv logbogskommentarer

Formål:
Der skal være mulighed for at kunne skrive en logbogkommentar på hvert tidselement.

Aktører:
Der er 2 aktører. Brugeren, der anmoder om at tilføje logbogskommentarer til et tidselement og databasen, der gør det.

Startbetingelser:
der eksisterer et tidselement

Beskrivelse:

Bruger	**Database**
Brugeren skriver en kommentar	
	Databasen tilføjer kommentaren til et valgt tidselement.

Slutbetingelse:
Der er tilføjet en logbogskommentar

Use case 2: Se logbogskommentarer

Formål:
Der skal være mulighed for alle brugere at vælge et tidselement og se de kommentarer der er blevet skrevet i dem.

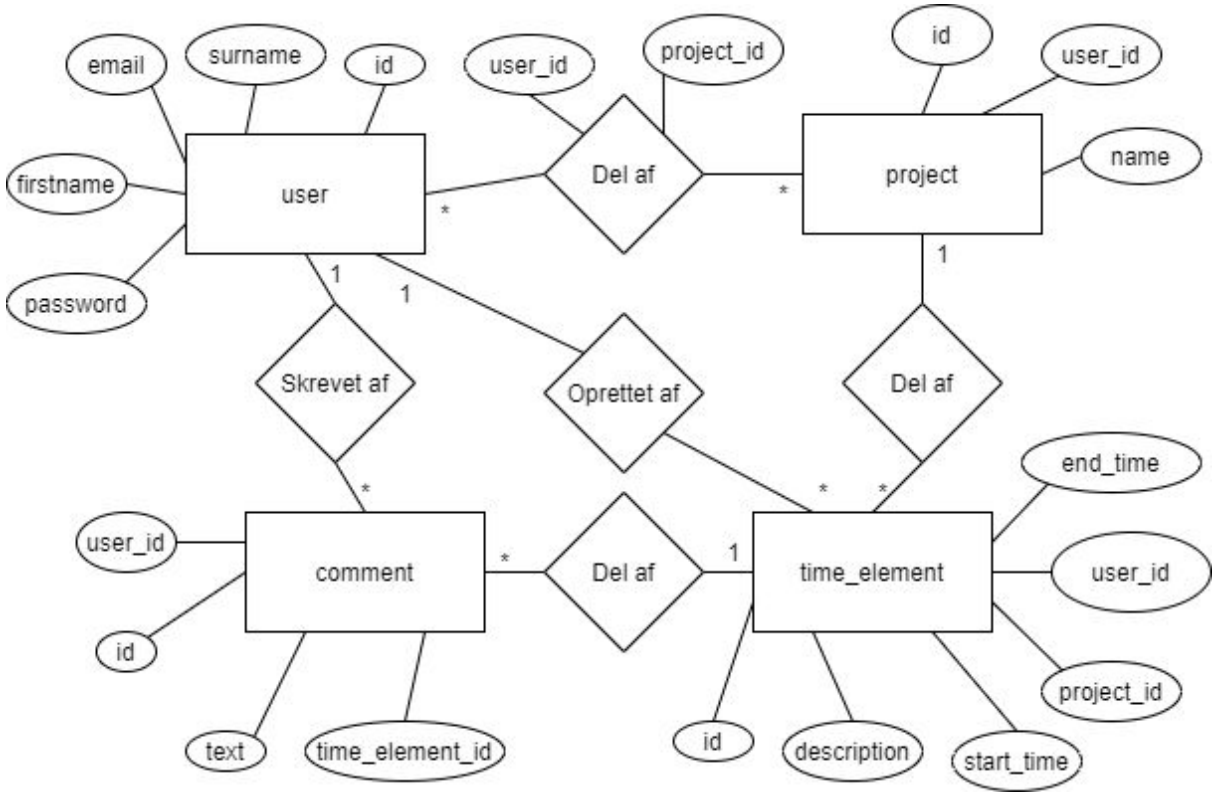
Aktører:
Der er 2 aktører. Brugeren, der anmoder om kommentarerne på et tidselement og databasen, der henter det.

Startbetingelser:
Brugeren har oprettet en bruger på register.php.

Beskrivelse:

Bruger	**Database**
Vælger et tidselement	
	Henter alle kommentarer under det valgte tidselement

Dokumentation af database



Normalformer

user

user overholder første normalform da alle attributterne er en-værdisattributter og den har også en primærnøgle, hvilket i dette tilfælde er “id”. Da primærnøglen ikke af sammensat af flere attributter opfylder den anden normalform. Desuden afhænger ingen af attributterne indirekte af primærnøglen, hvilket gør at den også opfylder tredje normalform. Se figur user.

	id	email	firstname	surname	password
Edit Copy Delete	1	erikbuur@hotmail.com	Erik	Christensen	$2y$10$0ZK22PlVpdktbNg3BUQZwu.r6hlG.wOhwN6lTkcNa1z...
Edit Copy Delete	2	julemand@jul.dk	Julemand	Julendal	$2y$10$wk5g6wHVDYaSMCqTy.BJUuS8lNYsrCCeeP/9HFbjG2b...
Edit Copy Delete	3	olivergeneser1@gmail.com	Oliver	Geneser	$2y$10$lVjElyri2LKWkpxN2mdVUutHNGLTwjir2Cnc9rQZWYh...
Edit Copy Delete	4	SoberVampire@gmail.com	Emiel	Regis	$2y$10$Nbk0c.ySdu1GqUxY8l.UO.8Q4QwVdgmGUxxEpcpkN8O...

Figur user: Eksempeldata for tabellen *user*.

project

project overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, “id”, primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen. Se figur project.

	id	name	user_id
Edit Copy Delete	44	Programmering - Sortering	4
Edit Copy Delete	45	Teknikfag - Eksamensprojekt	4
Edit Copy Delete	46	Kemi - Titrering	4

Figur project: Eksempeldata for tabellen *project*.

comment

comment overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, “id”, primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen. Et eksempel på noget data i denne tabel er vist i figur comment.

	id	user_id	time_element_id	text
Edit Copy Delete	28	2	1	Write your LogComment here...
Edit Copy Delete	43	1	2	Write your LogComment here...
Edit Copy Delete	48	1	13	William er gennemsnitlig
Edit Copy Delete	49	1	13	Jeg har færdiggjort iteration 5 i dag

Figur comment: Eksempeldata for tabellen *comment*.

time_element

time_element overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, "id", primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen. Se figur time_element.

	id	project_id	description	user_id	start_time	end_time
Edit Copy Delete	7	44	Begynd på sorteringsalgoritmer. Oliver - mergeS...	4	2020-02-03 00:00:00	2020-02-03 00:00:00
Edit Copy Delete	10	45	Idegenrer, undersøg og planlægl	4	2020-02-27 00:00:00	2020-02-27 00:00:00
Edit Copy Delete	11	45	Lav spørgsmål til brugerundersøgelse og begynd på ...	4	2020-03-03 00:00:00	2020-03-03 00:00:00
Edit Copy Delete	12	46	Lav forsøg og nedskriv data	4	2020-02-12 00:00:00	2020-02-12 00:00:00
Edit Copy Delete	13	46	Beskriv og visualiser tidligere indsamlet data	4	2020-02-17 00:00:00	2020-02-17 00:00:00

Figur time_element: Eksempeldata for tabellen *time_element*.

user_project

Denne tabel er ikke for at indeholde data selv, men for at vise en sammenhæng mellem data fra tabellen user og tabellen project. Der er kun to kolonner i tabellen, en for user_id og en for project_id. Primærnøglen i tabellen er kombinationen af de to kolonner, og dermed må der ikke være nogen rækker der har ens værdi i begge kolonner, men det samme user_id må gerne gå igen, og det samme project_id må gerne gå iden. Siden det eneste der er i tabellen er primærnøglen, overholder denne også alle tre normalformer.
Et eksempel på data i tabellen kan ses på figur user_project.

	user_id	project_id
Edit Copy Delete	1	10
Edit Copy Delete	1	22
Edit Copy Delete	1	23
Edit Copy Delete	2	10
Edit Copy Delete	2	14
Edit Copy Delete	2	22
Edit Copy Delete	3	13
Edit Copy Delete	3	14

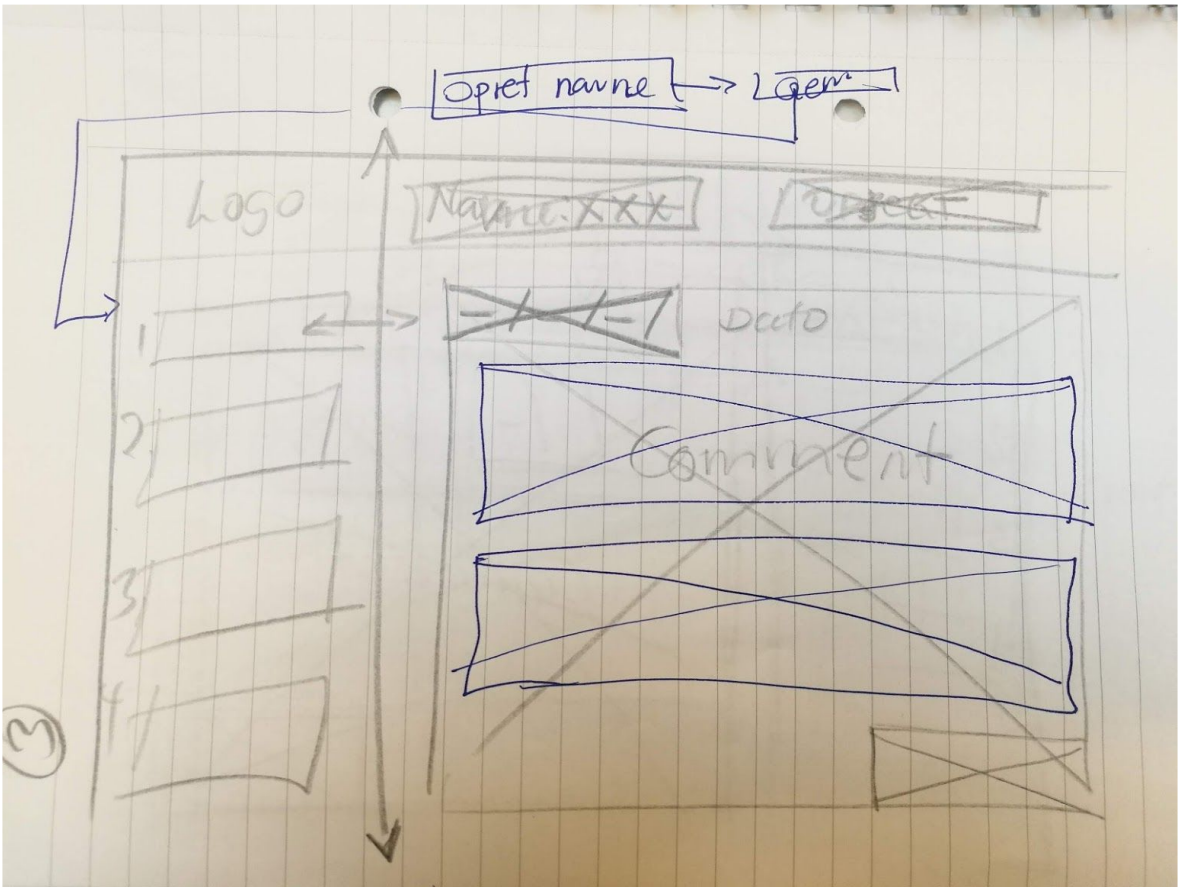
Figur user_project: Eksempeldata for tabellen *user_project*.

Sikkerhed

Hypertext Transfer Protocol Secure, forkortet til HTTPS er en krypteringsmetode, der bruges til at kryptere HTTP. Den sørger for at, det kun er afsender og modtager der kan læse indholdet af en meddelelse og eliminere på den måde chancen for “man-in-the-middle” angreb. Mange webbrowsere bruger denne protocol automatisk. I vores tilfælde er der indsat en en fil ved navn “.htaccess” i vores FTP servers root. Denne fil sørger for at vores browser bruger HTTPS protokolen

Visuel planlægning

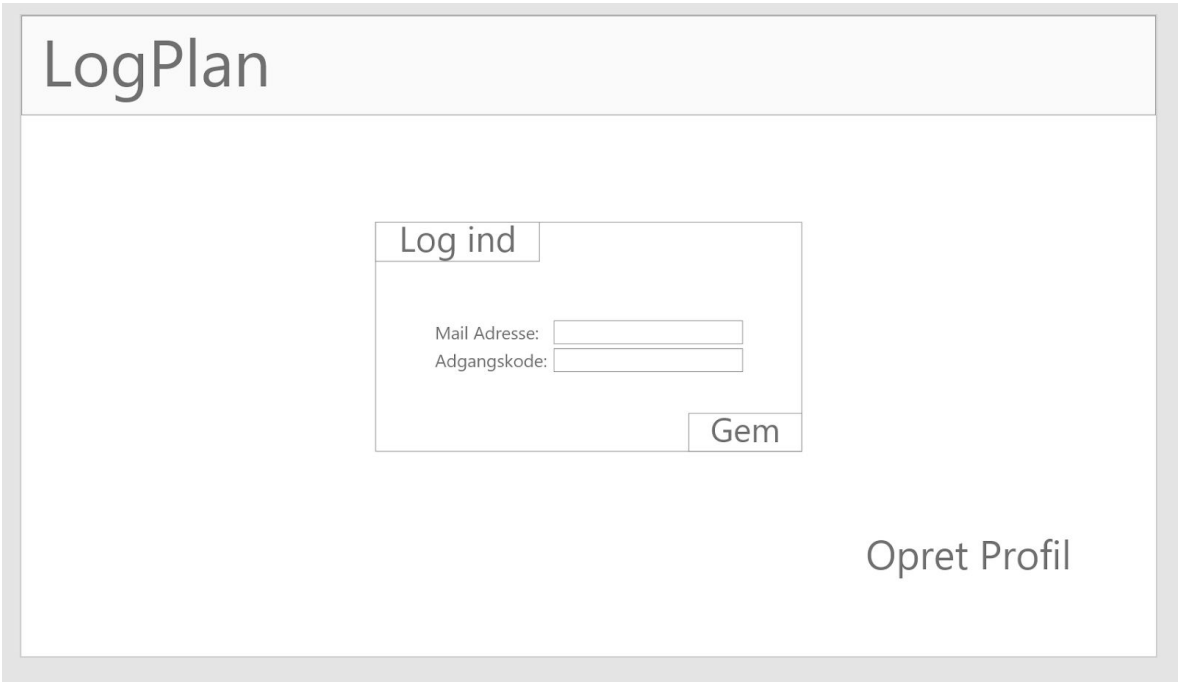
Den visuelle planlægning bestod af 3 skridt. Først tegnede vi løse sketches til hurtigt at skabe udkast til design, så gruppen kan vurdere dem, uden at for meget arbejde ville gå tabt, hvis designet ikke bliver godtaget. Se figur UDKAST.



Figur UDKAST: Her kan ses et udkast til et design hvor et tids-elements indhold skal kunne ses samtidigt med alle projektets andre tidselementer. Dette design blev kasseret, da koden først blev planlagt og deri var der ikke mulighed for at disse funktioner kunne være tilgængelige på en webside.

Når et endeligt design blev valgt lavede vi derefter mockups til hver webside. Mockups'ne var tilnærmelsesvis bare pæne sketches, men gav stadig en bedre idé om størrelsesforhold. Derudover klarlægger mockups også hvilke elementer der skal være på en hjemmeside. Se figur MOCKUP. Flere sketches og mockups kan findes i bilag. I vores planlægning af webstedets udseende brugte vi ikke wireframes til at bestemme passende afstande i og mellem elementer, da det ville havde kræve mere arbejde i forhold til at teste afstandene under kodningen.

Derudover er webstedets udseende kun designet til fuldskærm og vil derfor reagere visuelt dårligt på at vinduet bliver mindre.



Figur MOCKUP: Her kan ses vores mockup til login.php. Mockup'en har hverken farve eller korrekte afstande, men giver overblik over elementer og deres placering i forhold til hinanden.

Dokumentation af implementation

Det websted vi har implementeret består af flere php-filer, og herunder beskriver vi hver fils funktion.

Kommentarer om implementation

Fejl er en naturlig del af tilværelsen og dermed stødte vi selvfølgelig ind i nogle problemer under implementationen. Disse problemer vil vi klargøre her.

Under programmeringen af webstedets funktionalitet brugte vi html input'et datetime-local til at brugeren kunne angive tidselementers varighed. Dette gav god mening da den giver mulighed for at specificere både datoen for elementet og tidspunktet på dagen. Under implementationen af css opdagede vi dog at html input'et datetime-local ikke er understøttet af browseren Firefox. Vi stødte dermed ind i et dilemma, da databasen og koden var lavet til at modtage to instanser af datetime-local til at dokumentere hvornår og hvor længe brugerens arbejdessessioner var, men dette vil ekskludere alle brugere af Firefox. Vi valgte at skifte datetime-local ud med date, da vi prioriterede tilgængeligheden af vores websted frem for funktionen. Da dette problem først blev opdaget sent, er der stadig nogle "levn" fra vores tidligere design.

Vi mødte også et problem under implementation af navigationen på webstedet, hvilket fandt sted sammen med implementationen af css. Hovedsageligt består navigation af <a>-tags rundt om et <p>-tag, men da noget navigation blev lavet sammen med webstedets funktionalitet kunne navigationen ikke ændres til <a>-tags. Det omhandler time_element.php, hvor back-knappen er lavet ved hjælp af en <form> og tilhørende php-kode. Et <a>-tag vil i dette tilfælde gå over php-koden og forbigå de operationer koden ellers ville have gjort. Dette problem manifesteres kun ved at back-knappen kun fungerer, hvis der klikkes på teksten "back", hvor andre knapper kan klikkes på hvor end brugeren ønsker.

Derudover stødte vi også på et lidt specielt problem. Under implementationen blev vi bedre til at kode css og dermed blev hver side pænere end den næste. Dette har skabt inkonsekvent detalje design. Dette blev ikke rettet, da der kun blev sat tid af til at implementere css en gang, og der vil opstå tidspres hvis hver webside skulle istandsættes igen

config.php

Denne php-fil vises ikke for brugeren, men inkluderes i filerne for de andre websider. Dette gør at koden i denne fil eksekveres når denne fil bliver inkluderet i en anden.
Koden har den funktion at logge ind på webstedets korresponderende database. Dette gøres ved at definere serverens navn, databasens navn, samt brugernavn og adgangskode

til databasen. Disse informationer bruges nu til at skabe en forbindelse til databasen med new mysqli-funktionen.

index.php

Denne php-fil beskriver den webside man først kommer ind på når man åbner vores websted. Denne webside består af to knapper der fører en til henholdsvis login eller registrer. Knapperne er html-inputs af typen submit, og de ligger i en form der bruger metoden post. Der bruges en if-sætning til at tjekke om der serveren har modtaget en request af metoden post når siden loader, og derefter tjekkes det hvilken af inputene der blev kaldt. Afhængigt af hvilket af inputene der blev brugt, bruges header-funktionen til at sende brugeren videre til enten register.php eller login.php.

I starten af filen startes også en session, som bruges til at gemme variable på tværs af websiderne. Der bruges også funktionen session_unset-funktionen, så man er logget ud når man kommer til index-siden, da vores log-ud knapper typisk kun sender brugeren tilbage på index-siden, og ikke andet.
Config.php inkluderes også, som beskrevet foroven.

login.php

På denne webside er der to tekst-input og et submit-input i en form med metoden post. Når siden loades, tjekkes om de to tekstinputs er tomme. Hvis de begge er udfyldt, bruges sql's SELECT funktion, med specifikationerne FROM og WHERE, til at finde det password i databasen, som passer til den bruger der har den indtastede email. Nu bruges funktionen password_verify til at tjekke om det indtastede password matcher den hashede version der ligger i databasen. Hvis dette er tilfældet, sættes en sessionvariabel til brugerens email, og brugeren bliver sendt videre til main.php.

Udover denne mekanisme, startes sessionen, så der kan oprettes sessionvariable, og config.php inkluderes også.

register.php

På denne webside er der fire tekst-input og et submit-input, som alle ligger i den samme form med metoden post. Hver af tekst-inputene bruges til en af fire informationer der skal logges i databasen når der oprettes en bruger. Disse fire informationer er email, fornavn, efternavn og kode. På samme vis som i index.php, tjekkes efter en server-request af metoden post, og derefter om der blev postet noget fra submit-inputtet. Hvis dette er tilfældet, tjekkes om den email der blev indtastet allerede er oprettet som bruger. hvis dette er tilfældet, gives en fejlmeddelelse. Hvis emailen ikke bliver brugt, gemmes de indtastede informationer i variable, kode bliver dog hashet først, med funktionen password_hash.

Nu laves et sql-statement der bruger INSERT funktionen med specifikationerne INTO og VALUES der beskriver hvilken tabel i databasen der skal indsættes data i, og hvilke værdier der skal indsættes.
Efter dette er udført, bliver brugeren sendt over til login.php med header-funktionen.

main.php

I main.php startes sessionen og config.php inkluderes som i de tidligere beskrevne filer, men nu tjekkes også om der er sat en sessionvariabel for email. Hvis der ikke gør, sendes man tilbage til index.php, da dette er måden der tjekkes om man er logget ind.
Vi definerer også en funktion her kaldet userID($email, $conn). Den modtager parametrene $email og $conn, hvor $email er den email man vil finde user-id'et for, og $conn er den connection til den database der skal ledes i. Funktionen Bruger et sql SELECT statement, til at finde id FROM tabellen user WHERE email er lig den email der gives som parameter til funktionen. Funktionen returnerer id'et for brugeren med denne email.

Nu bruges en if-sætning til at tjekke om serveren har modtaget en request af metoden post, og hvis dette er tilfældet, skal det nu tjekkes hvilken af de tre forms i filen der blev brugt. Dette gøres med if- eller else if-sætninger, der bruger isset-funktionen til at tjekke om der er blevet postet noget fra hver af de tre formers submit-input.
Hvis inputtet med navnet newProject er sat, skal der nu oprettes et nyt projekt i databasen, i tabellen project. For at gøre dette findes først brugerens id ved at bruge funktionen userID med den email der gemmes som sessionvariabel, som parameter. Også den forbindelse der defineres i config.php er en variabel.
Navnet projektet skulle have hentes fra tekstinputtet med navnet projectName, nu indsættes dataen som en ny række i project tabellen i databasen, med sql-funktionen INSERT.

Hvis inputtet med navnet open er sat, findes id'et på det projekt der skal åbnes. Dette gøres ved at tage hvad der blev postet fra inputtet med navn openid. Nu sættes en sessionvariabel med navnet project til at være lig dette id, og header bruges til at sende brugeren til project.php.

Hvis inputtet med navnet dlt er sat, findes id'et på det projekt der skal slettes. Dette gøres ved at tage hvad der blev postet fra inputtet med navn dltid. Nu bruges sql-statementet "DELETE FROM project WHERE id=$commid;", hvor commid er det id der blev fundet fra dltid-inputtet. Efter dette statement er blevet eksekveret, er koden der køre for server-requests færdig.

Nu findes bruger-id'et for den bruger der er logget ind. Det gøres med userID-funktionen, og sessionvariablen for email, og id'et gemmes som variablen $userid. Nu bruges følgende sql-statement: "SELECT * FROM project INNER JOIN user_project ON project.id = user_project.project_id WHERE user_project.user_id = $userid;". Det kobler tabellerne project og user_project, hvor id fra tabellen project er lig project_id fra tabellen user_project. Derefter vælger den alle rækker hvor user_id fra tabellen user_project er lig det tidligere fundne user-id.

Efter disse rækker er valgt og gemt under variablen $result, bruges en if-sætning til at tjekke om der er flere end 0 rækker. Hvis der er, oprettes en html-div, hvorefter der køres en while-løkke, der kører for hver række i resultatet. I løkken oprettes først en ny div på siden. Nu gemmes resultatrækkens værdi for name og id i hver deres variabel. Nu bruges id'et til at lave et nyt sql-statement der SELECT'er user_id fra tabellen project hvor id er lig det id der blev fundet. Dette gemmes under variablen $creator. Nu bruges echo-funktionen til at skrive navnet på projektet, og til at oprette en form med et submit-input med navnet open, og et hidden-input med navnet openid og en værdi lig projektets id.

Nu bruges en if-sætning til at tjekke om variablen $creator er lig variablen $userid. Hvis dette er sandt, bruges echo-funktionen til at skrive en form ud med et submit-input med navnet dlt, og et hidden-input med navnet dltid, og en værdi lig projektets id. Til sidst i løkken lukkes den div der bliver oprettet først i løkken. Efter løkken lukkes den div der bliver oprettet først i if-sætningen. Denne del sørger for at man kun kan tilgå sletknappen til projekter, hvis man selv har oprettet dem, specifik kode kan ses på figur SletKode.

```
$creator = $row2['user_id'];
        echo "<h1>$name</h1> <form method='POST'> <input type='submit'
name='open' value='Open' /><input type='hidden' value='$id'
name='openid'/></form></div>";
        if($creator == $userid){
          echo "<div class='deletThis'><form method='POST'><input
type='submit' name='dlt' value='Delete'><input type='hidden'
name='dltid' value='$id'></form></div>";
```

Figur SletKode

project.php

Først i filen startes sessionen, der oprettes forbindelse til databasen, og der tjekkes om brugeren er logget ind. Derefter er der defineret to funktioner. userID som er magen til funktionen af samme navn der blev beskrevet i main.php, og projectName, der på samme måde som userID finder ud fra en email, finder navnet på et projekt ud fra et projekt-id.

Hvis serveren har modtaget en request af metoden post, tjekkes også her hvilket input der bliver brugt. Der er fire forskellige af disse submit-inputs, men da to af dem er i den samme form, er der kun tre forms i alt. Hvis det er submittet med navnet add, tages værdien af inputtet med navn mail, og gemmer dette i en variabel. Nu bruges userID-funktionen på dette, og hvis der bliver returneret null, bruges echo til at skrive en fejlbesked. Nu bruges et sql-statement til at tjekke om der er sammenhæng mellem en bruger af det fundne id, og projektet man arbejder i, i tabellen user_project. I så fald bruges echo til at fortælle brugeren at den indtastede email er en del af projektet i forvejen. Hvis brugeren af den indtastede mail ikke er med i projektet, bruges sql-funktionen INSERT til at indsætte en række i tabellen user_project, der beskriver en sammenhæng mellem brugeren med den indtastede mail, og projektet der bliver tilføjet. Hele dette stykke er for at sikre at man ikke kan tilføje brugere til et projekt som de allerede er en del af. Koden til det kan ses på figur Emailinput.

```
if ($_SERVER["REQUEST_METHOD"] == "POST")  {
  if(isset($_POST['add'])){
    $input = $_POST['mail'];
    $inputid = userID($input, $conn);

    if($inputid == null){
      echo "<div class='error'>user does not exist</div>";
    } else{
      $proid = $_SESSION['project'];
      $sql = "SELECT * FROM user_project WHERE user_id='$inputid' AND
project_id='$proid';";
      $result = $conn->query($sql);
      if($result->num_rows > 0){
        echo "allerede del af projekt";
      } else{
        $sql = "INSERT INTO user_project (user_id, project_id) VALUES
('$inputid', '$proid');";
        $conn ->query($sql);
      }
    }

  } else if(isset($_POST['time-element'])){
```

Hvis submit-inputtet med navnet time-element bliver brugt, gemmes værdierne for de to inputs ‘start’ og ‘end’, i hver deres variabel. Derefter findes brugerens id med funktionen userID, og projektets id fra sessionsvariablen findes også, og gemmes i hver deres variabel. Nu bruges et sql-statement til at indsætte disse data i en ny række i tabellen time-element.

Hvis submit-inputtet med navnet descript bliver brugt, findes værdien fra inputtet time_element_id, og der sættes en sessionvariabel med navnet time-element til dette. Derefter bruges header-funktionen til at sende brugeren til time_element.php.

Hvis submit-inputtet med navnet descript bliver brugt, findes værdien fra inputtet time_element_id, og der bruges en sql-statement til at slette rækken i tabellen time_element der har id lig, værdien.

Inden i html’en, bruges funktionen projectName til at finde projektets navn ud fra sessionvariablen om projektets id, hvilket bliver echo’et inden i et html h1-tag.

Nu bruges et sql-statement til at finde alle rækker i tabellen time_element, med project_id lig det id der er i sessionvariablen for project. Der bruges en if-sætning til at tjekke om mængden af rækker der er blevet valgt, er større end 0. Hvis dette er tilfældet oprettes nogle html-elementer, og der kører nu en while-løkke, der kører en mængde gange lig mængden af valgte rækker.

I løkken findes først værdierne for start_time og id på rækken, som gemmes i hver deres variabel. Nu bruges echo-funktionen til at udskrive start_time, og en form med tre inputs. Et submit-input med navnet descript, et submit-input med navnet dlt, og en hidden-input med navnet time_element_id og værdien sat lig det id der tidligere blev fundet.

Nu slutter while-løkken, og html-tags'ene lukkes.

time_element.php

Denne fil starter med session_start, config.php, og tjek efter login, ligesom de tidligere filer. userID-funktionen er også defineret, da den bruges.

Først gemmes sessionvariablen for time-element i variablen $timeid. Derefter tjekkes efter posts. Hvis inputtet back blev brugt, bruges header til at sende brugeren tilbage til project.php. Hvis det derimod er adddes der blev kaldt, gemmes først værdien fra text-areaet dscrpt, hvorefter sql-funktionen UPDATE bruges til at SET'te description i den række i tabellen time_element, hvor id er lig sessionvariablen for time_element, som tidligere blev gemt i en variabel.

Hvis inputtet er af navnet postComment, gemmes værdien fra inputtet commentText i en variabel, og userID bruges til at gemme id'et på brugeren i en variabel. Efter dette bruges et sql-statement med funktionen INSERT til at indsætte en række i tabellen comment, med dataen fra disse to variable, og tidselementsid'et der blev fundet først på siden.

Hvis inputtet er af navnet dlt, findes værdien fra inputtet dltid, og der bruges et sql-statement af funktionen DELETE til at slette rækken i tabellen comment, hvis id er lig værdien fra dltid.

Efter dette bruges sql-funktionen SELECT til at vælge rækken i tabellen time_element hvor id er lig $timeid. Værdierne i kolonnerne description, start_time og end_time, gemmes nu i hver sin variabel.

Nu bruges et nyt sql-statement til at vælge alle rækker i tabellen comment, hvor time_element_id er lig $timeid. Dette resultat gemmes i en variabel.

Nu kommer html-elementerne i filen, og de tidligere fundne start_time og end_time bliver vist i de korrekte tags ved at echo'e dem. Der bruges også echo til at oprette et textarea i en form, med navnet dscrpt, og inde i textarea-feltet echo'es det der blev taget fra kolonnen description. Det smarte ved at vise description i et texarea, er at man kan redigere direkte i det uden at skulle åbne en ny webside, eller kun kunne skrive tillæg eller lignende. Hvordan det blev kodet kan ses på figur Textarea

```
<?php
echo "<textarea name='dscrpt'>$dscrpt</textarea>";
?>
```

Figur Textarea

Hvis der er mere end 0 rækker i resultatet for det seneste sql-statement, kører nu en while-løkke, der kører en gang for hver række i resultatet. I løkken Findes først værdierne for id, text og user_id i rækken, hvorefter et nyt sql-statement bruges til at vælge firstname og surname fra tabellen user, fra rækken hvor id er lig det tidligere fundne user_id. Dette gemmes i en anden resultatsvariabel. Nu gemmes disse to i hver deres variabel, og den bruger der er logget ind's id findes med userID-funktionen. Nu bruges echo-funktionen til at printe firstname og surname, samt text. Derefter bruges en if-sætning til at checke om user_id'et der blev fundet i rækken er lig user_id'et for brugeren der er logget ind. Hvis dette er tilfældet, bruges echo til at printe en form med to inputs, et submit-input med navnet dlt, og et hidden-input med navnet dltid, og værdi lig id fra rækken.

Nu afsluttes while-løkken. I tilfælde af at der ikke var nogle rækker, bliver der i stedet for at køre while-løkken, printet en besked med echo-funktionen om dette.

Implementationen af udseende

Selve kodningen af webstedet html og css foregik sådan: Først blev hvert websides funktionalitet programmeret uden nogen hensyn til html og css. Når hvert webside var færdig programmeret gik programmøren videre til den næste webside og designeren begyndte at istandsætte html-koden og implementere css i form af et stylesheet. Stylesheet'ets struktur er således: De første elementer man måder på siden ligger først i stylesheet'et. Altså, header'en ligger allerøverst og elementer der har at gøre med header'en ligger også i toppen af stylesheet'et. Derefter ligger stylesheet'ets klasser efter hvilket webside man møder dem. Klasser til time_element.php ligger altså efter klasser til login.php.

Klassen til webstedets header er gentaget, men på nogle websider tilføjes der elementer til header'en.

Organisation

Vi har brugt Github til at administrere vores kode under implementation, hvor vi havde en master-branch til færdige iterationer, en development-branch til iterationen under udvikling, og en styling-branch, til at skrive css i. Her er linket til repository'et:
https://github.com/VeryThankYou/LogPlan
Udover dette har vi benyttet os af vores iterationsplan, som vi har fulgt under udviklingen af programmet, så vi implementerede det et skridt af gangen.

Refleksion

Det første reflektionspunkt vi vil nævne er Rapport > Produkt. Under opgaven fokuserede vi hovedsageligt på implementationen af vores kravspecifikation, men dette gjorde at vores produkt i visse områder ikke er dokumenteret ordentligt eller tilstrækkeligt i vores rapport. Vi vil i næste opgave altså bruge flere kræfter på selve dokumenteringen af vores produktion i

stedet for selve produktionen. Dette vil muligvis manifestere i, at vi er mindre ambitiøse med vores produkt, for at kunne nå at implementere det til samme kvalitet som tidligere, og samtidigt dokumentere det til bedre kvalitet end tidligere. Heri ligger også at bruge mere tid på test og brugerundersøgelser. Vi har i denne opgave lavet en brugerundersøgelse på en person, hvilket ikke giver et tilstrækkeligt grundlag til at opstille generaliseringer. Derudover er der ikke lavet nogle

De næste reflektionspunkter er mindre, men stadig relevante. Det første er: Lav et kort over webstedets struktur sammen med kravspecifikationen, så det bliver fastlagt og klargjort fra starten af. ER-diagram mangler datatyper og use-case diagrammerne er for tekniske. Alle disse punkter kan opsummeres med: øvelse gør mester. Det er fejl der er opstået på grund af antagelser og misforståelser og er ikke svære at undgå næste gang, da vi nu ved hvordan vi laver dem ordentligt fra starten af.

Konklusion

Målet med vores produkt var at simplificere dokumentationen af en gruppes projektstyring, så gruppen kan få nemmere få højere karaktere. Både da projektstyring er kernestof i teknikfag og teknologi, men også da optimal projektstyring hjælper en gruppe med at arbejde optimalt.

Det er dog svært at konkludere på produktets brugbarhed, da den ikke er blevet testet. Dog kan implementationen af vores udviklingsplan konkluderes. Implementationen af vores udviklingsplan gik godt og alle iterationer nåede at blive implementeret. Dog opstod der divers problemer under implementationen af webstedets visuelle design. Dette har skabt tilfælde af inkonsekvent design og en enkelt overflødighed i databasen.

Bilag

Bilag 1, Original kravspecifikation

Problemformulering

Projektarbejde er en stor del af en HTX-uddannelse og det handler ikke kun om at lave et produkt eller at dokumentere det. En stor del af projektarbejde er også administration og organisering, men det er ikke noget studerende typisk har nogen erfaring eller dybdegående undervisning i. Og det kan medføre, at projekter går dårligt, ikke på grund af dårligt arbejde, men på grund af dårlig planlægning. Og derudover er projektstyring kernestof i både Teknikfag og Teknologi[2], så det kan forbedre ens karaktere, hvis man inkluderer dokumentation af ens projektstyring, f. eks. tidsplan og logbog, i ens rapport.

Derfor kunne værktøjer til at assistere studerende i disse dele af projektarbejdet være en god idé. Bedre projektstyring hjælper både med at lave et bedre produkt og med at opfylde fagenes kernestof, hvilket begge kan betyde bedre karaktere. Og dem, som allerede har gode karaktere, kan brude mindre tid på deres projektstyring for samme kvalitet.

Hvordan kan man hjælpe HTX-studerendes projektstyring og dokumentation heraf vha. værktøjer til at gøre projektstyring og -dokumentering nemmere?

Brugsscenarier og målgruppeanalyse

Brugere af vores program er HTX-studerende i deres projektarbejde. HTX har projekter i mange af deres fag. Dette inkluderer: Teknologi, Teknikfag, Kommunikation og it, og Programmering. I teknologi og teknikfag er projektstyring et relevant fagområde og eleverne bliver dermed undervist i det. Men for at undersøge hvorvidt eleverne også bruger projektstyring, er man nødt til at lave en brugerundersøgelse.

Brugerundersøgelse

Vi undersøgte denne problemformulerings aktualitet vha. et interview med en 3. g'er. Til spørgsmålet "Hvor ofte laver du dokumentation over jeres gruppearbejde? F. eks. logbog eller tidsplan." svarede interviewpersonen, at han i hans gruppes projekter ikke laver logbog så meget, men de har nogle gange haft fat i tidsplanlægning. Men til gengæld bliver tidsplanerne ikke brugt så meget. Derefter spurgte vi hvorvidt den sidste periode i hans projekter er mere presset og stresset end resten. Hertil svarede han ja. Vores sidste spørgsmål var hvorvidt han mente, at projektstyring er på lig fod med andre fagligheder i et fag. F. eks. om evnen til at programmere lige så vigtig i et programmeringsprojekt som

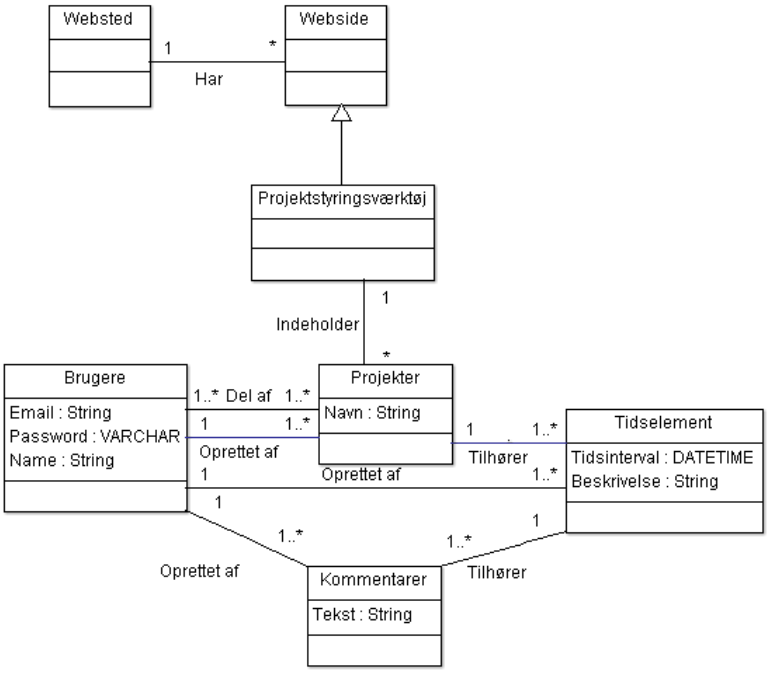
[2] https://www.uvm.dk/gymnasiale-uddannelser/fag-og-laereplaner/laereplaner-2017/htx-laereplaner-2017

evnen til at styre projektet. Hertil svarede han, at det er vigtigere at bruge sin tid rigtigt, da ens programmering evner kun kan blive brugt, hvis man har tid til det.

Man kan altså konkludere at mindst en gruppe på HTX kunne bruge et redskab der kunne hjælpe dem med at lave deres projektstyring og deres dokumentation af deres projektstyring.

Overordnede begreber

Der skal laves et projektstyringsværktøj, og det skal ligge på et websted. Webstedet består af flere forskellige websider, som hver især står for en del af værktøjet. Brugere kan med værktøjet oprette projekter, og tilføje andre brugere til sine projekter. Alle brugere som er del af et projekt kan i projektet oprette tids-elementer. Disse elementer skal indikere tidspunkter brugerne skal arbejde på deres projekt i. Derfor gives tids-elementerne et tidspunkt/dato, og en beskrivelse, der fortæller hvad brugerne har planlagt at lave på projektet i dette tidsinterval. Derudover kan brugerne knytte kommentarer til tids-elementerne, hvilket kan bruges til at føre logbog over hvad der faktisk blev lavet i tidsintervallet.



Figur 1: Diagram over sammenhæng mellem begreber

Beskrivelse af begreber

Vi har her udarbejdet en tabel over hvilke navneords-begreber vi bruger i forhold til vores produkt, og beskrevet disse, og en tabel over hvilke udsagnsords-begreber vi bruger i forhold til vores produkt, og beskrevet disse.

Navneord	Betydning
Websted	Webstedet er den webadresse hvor vores projektstyringsværktøj ligger på
Webside	Webstedet består af websider som hver indeholder en del af vores projektstyringsværktøjer.
Projektstyringsværktøjet, eller “LogPlan”	Selve det produkt vi udarbejder. Det ligger på tværs af de websider der er på webstedet.
Projekt	Projekter er noget som brugere kan oprette med projektstyringsværktøjet, og bruges til at sammenkoble hvilke brugere der arbejder sammen, og hvilke tids-elementer der hører til sammenkobling.
Tids-element	Tids-elementer hører til et projekt, og beskriver en tidsperiode. Dermed har tids-elementer et tidsinterval, men de har også en beskrivelse, der beskriver hvad brugerne har planlagt at lave på projektet i tidsintervallet
Bruger	Er den person som opretter en konto på vores projektstyringsværktøj.
Kommentarer	De tekster brugere tilføjer til tid-element for at beskrive hvad der faktisk skete i tidsintervallet. En slags logbog

Udsagnsord	Betydning
Tilføje	Det at brugere kan tilføje andre brugere til projekter som brugeren er en del af.
Oprette	Brugere opretter projekter i opret projekt, når de er logget ind.
Kommentere	Brugere kan kommentere tids-elementer i projekter som de er en del af.

Løsningens arkitektur

LogPlan skal kunne køre på et standard webhotel med Apache 2.4.29, PHP 7.3.8 og MariaDB 10.4.6.

Udviklingsplan

#	Fase	Subfase	Indhold	Flade
1	Login og registrer	- Opret bruger - Login til bruger	- Brugeren kan oprette sig selv som bruger - Brugeren kan logge ind efter de har oprettet sig	- Frontend - Frontend
2	Projekter	- Opret projekter	- Brugeren kan oprette en eller flere projekter	- Frontend
3	Kooperative projekter	- Tilføj andre brugere til projekter	- Brugeren kan tilføje andre til sine projekter, og de andre kan se alle projekter som de er en del af	- Frontend
4	Tidsplan	- Tilføje tids-elementer til projektet	- Brugeren kan tilføje elementer med mål og tidsperiode	- Frontend
5	Logbog	- Skriv logbogskommentarer - Se logbogskommentarer	- Brugere kan skrive logbogkommentarer inde på hver af tidsplanselementerne - Brugere kan se hvem der har skrevet, hvilke kommentarer	- Frontend - Frontend

Overordnede behov og krav

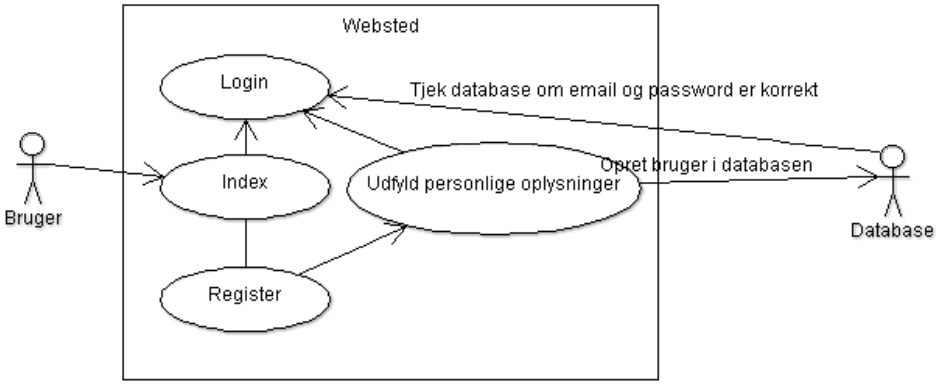
Der er følgende krav til vores produkt:

- Kan visualisere tids-elementer i en tidsplan
- Brugere skal kunne være en del af flere projekter
- Der skal være mulighed for at flere brugere er en del af samme projekt
- Brugere kan se alles kommentarer i projekter som de er en del af, men kun kunne redigere sine egne

Beskrivelse af faser

Iteration 1, Opret og login

Den første iteration af vores websted har bare login og registrering. Vi har valgt at starte med dette, fordi at den måde vi har lavet databasen på, krævede at der var brugere til at lave projekter og tilføje andre brugere til dem.



Figur 1: Use-case diagram over første iteration.

Use case 1: Registrer bruger

Formål:
Brugeren af webstedet kommer ind på index.php, og navigerer til registreringssiden, register.php. Brugeren udfylder formen på websiden, og opretter en bruger, som bliver indsendt til databasen.

Aktører:
Der er to aktører i denne iteration, en bruger på webstedet, og databasen, som agerer ved hjælp af requests eller inputs fra webstedet.

Startbetingelser:
Brugeren har åbnet index.php.

Beskrivelse:

Bruger	**Database**
Klikker på “Register”	
	Åbner register.php
Udfylder felterne på siden og klikker på “Create”	
	Opretter en bruger i databasen, der har de

	informationer som brugeren har tastet ind

Slutbetingelse:
Databasen er opdateret med den nye bruger som er oprettet.

Use case 2: Login

Formål:
Brugeren af webstedet har lige oprettet en bruger på register.php, og er dermed blevet sendt til login.php. Her logger brugeren ind.

Aktører:
Der er to aktører i denne iteration, en bruger på webstedet, og databasen, som agerer ved hjælp af requests eller inputs fra webstedet.

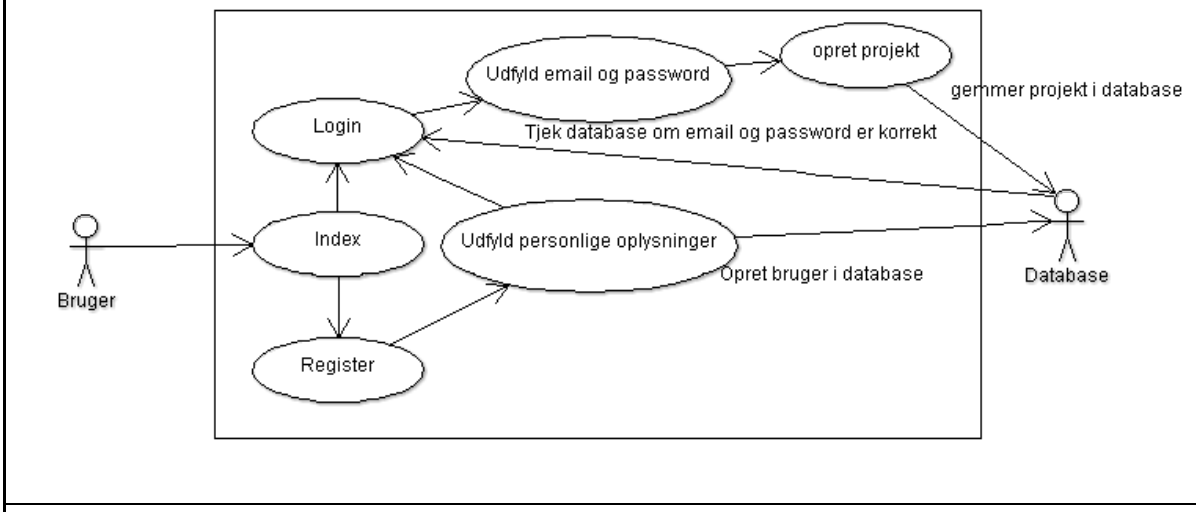
Startbetingelser:
Brugeren har oprettet en bruger på register.php.

Beskrivelse:

Bruger	**Database**
Udfylder den korrekte mail og adgangskode, og klikker på “Login”	
	Tjekker om der er en bruger i databasen med den korrekte mail-adresse, og ser derefter om den indtastede adgangskode matcher den hashede adgangskode i databasen. Hvis dette er tilfældet sendes man til siden main.php.

Iteration 2, opret projekter

Anden iteration er implementeringen af projekter. Dette omhandler hovedsageligt om at give brugeren mulighed for at oprette projekter efter de er logget ind.



Figur IT2: Usecase-diagram over anden iteration.

Use case 1: Opret projekt

Formål:
Der skal selvfølgelig også være et formål ved at logge ind. Og det er blandt andet at lave projekter. Projekter er grundlaget for resten af projektets funktioner, men gør i sig selv ikke noget.

Aktører:
Der er 2 aktører. Brugeren, der opretter projekterne, og databasen, der indsætter projektet i databasen.

Startbetingelser:
Brugeren er logget ind

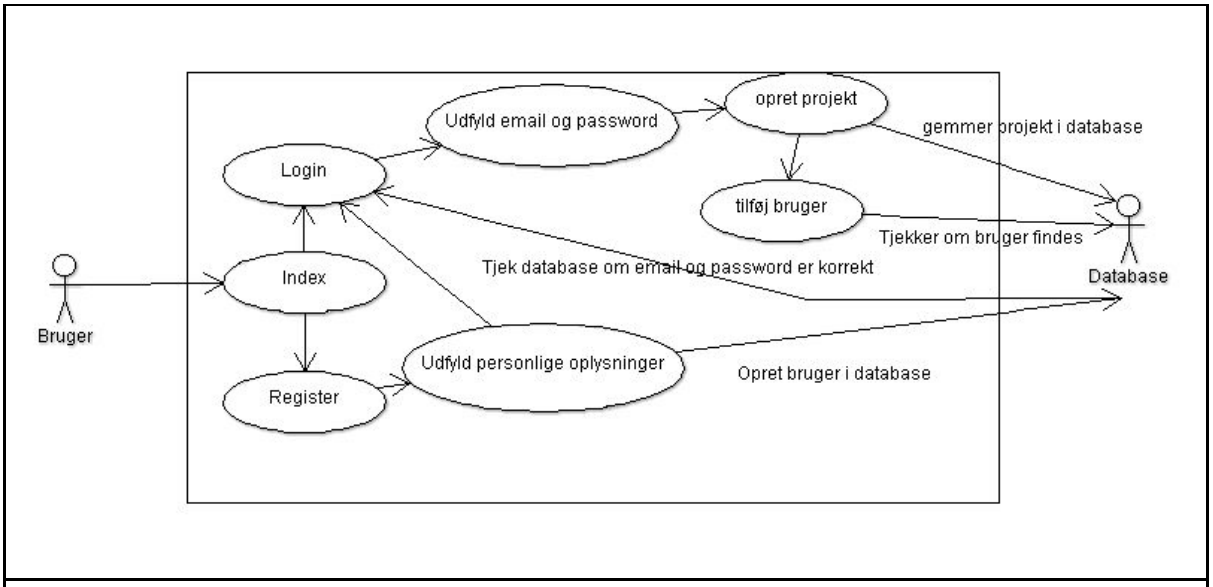
Beskrivelse:

Bruger	**Database**
Brugeren indtaster projektets navn	
	Databasen opretter projektet i databasen

Slutbetingelse:
Brugeren har oprettet et projekt

Iteration 3, kooperative projekter

Tredje iteration er udvidelse af projekter. Dette omhandler at give medlemmer af et projekt mulighed for at tilføje andre brugere. Projektets opretter er automatisk medlem af projektet.



Figur IT3: Usecase-diagram over tredje iteration.

Use case 1: Tilføj andre brugere til projekter

Formål:
Projekter giver kun mening, hvis der også er gruppearbejde og det kan kun ske hvis der er mere end en i projektet, så denne iterations formål er at tilføje andre til en eksisterende projekt

Aktører:
Der er 2 aktører. Brugeren, der anmodere om at tilføjelse af anden bruger, og databasen, der tjekker om den anden brugeren eksisterer og tilføjer brugeren til projektet

Startbetingelser:
Begge brugere og registreret og et projekt er oprettet

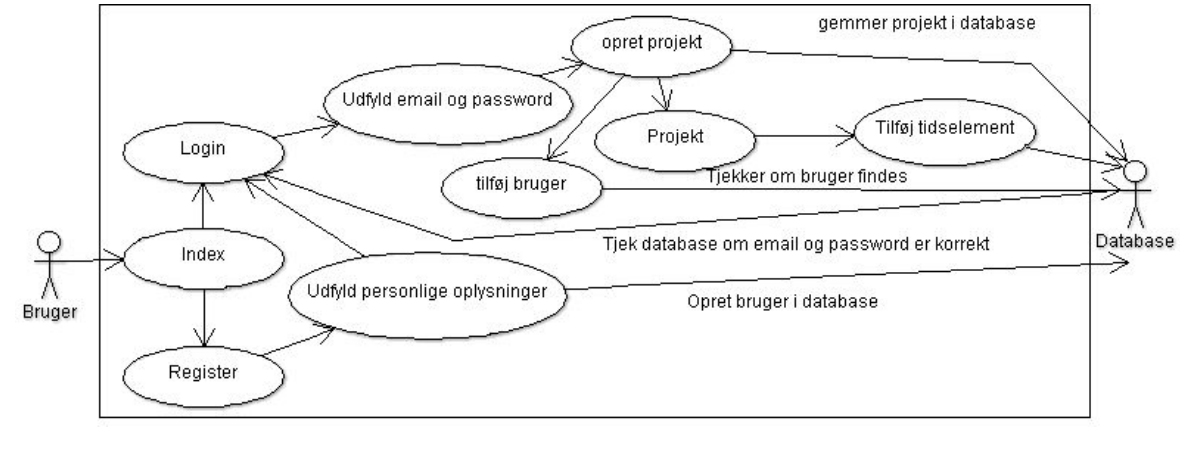
Beskrivelse:

Bruger	**Database**
Brugeren indtaster anden brugers navn	
	Databasen tjekker om brugeren eksisterer
	Databasen tilføjer anden bruger til projektet

Slutbetingelse:
Bruger nr. 2 er tilføjet til projektet.

Iteration 4, Tidsplan

Fjerde iteration er endnu en udvidelse af projekter. Denne omhandler implementeringen tidselementer.



Figur IT4: Usecase-diagram over fjerde iteration.

Use case 1: Tilføj andre brugere til projekter

Formål:
Indtil videre har vores produkt ikke haft nogen aktuel funktion, der kunne hjælpe med projektstyring. Men med tilføjelsen af tidselementer kan vores produkt faktisk bruges til noget.

Aktører:
Der er 2 aktører. Brugeren, der anmoder om at tilføje tidselementer til et projekt og databasen, der gør det.

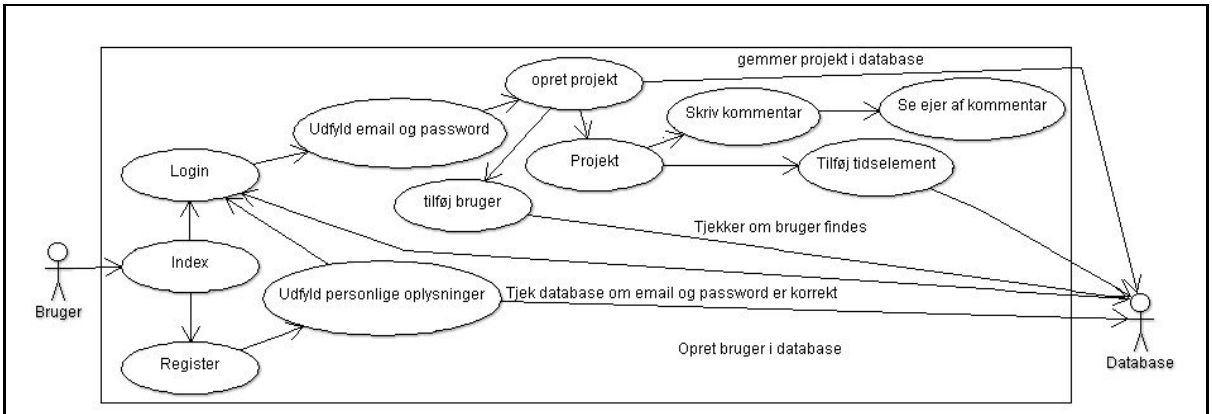
Startbetingelser:
Der er et projekt og en bruger

Beskrivelse:

Bruger	**Database**
Brugeren anmoder om at tilføje tidselementer til et projekt	
	Databasen tilføjer de tidselementer til projektet.

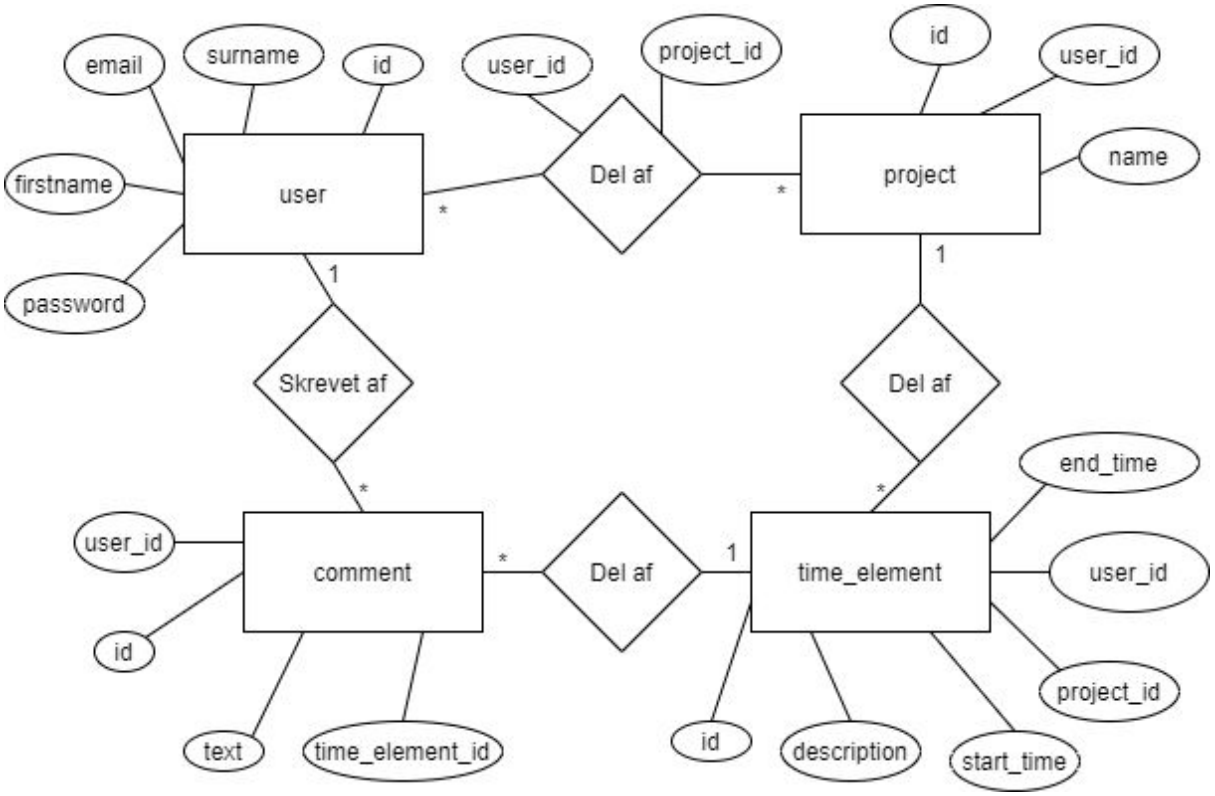
Slutbetingelse:
Tids-elementerne er blevet tilføjet

Iteration 5, Logbog



Figur IT5: Usecase-diagram over femte iteration.

Dokumentation af database



Normalformer

user
user overholder første normalform da alle attributterne er en-værdisattributter og den har også en primærnøgle, hvilket i dette tilfælde er “id”. Da primærnøglen ikke af sammensat af flere attributter opfylder den anden normalform. Desuden afhænger ingen af attributterne indirekte af primærnøglen, hvilket gør at den også opfylder tredje normalform.

project
project overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, “id”, primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen.

comment
comment overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, “id”, primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen.

time_element
time_element overholder også alle tre normalformer. Alle attributterne er en-værdisattributter, den har en primærnøgle, “id”, primærnøglen er ikke sammensat og ingen attributter afhænger indirekte af primærnøglen.

Bilag 2, Logbog

26-11-2019

Begyndte projektet.

Brainstorm om hvilket slags projektværktøj vi ville lave.

Lavede iterationsskema.

Planlagde database, lavede ER-diagram.

28-11-2019

Oprettet database
Lavet login og registreringsside

Første iteration, Login og registrer er færdig

03/12/2019

Arbejdet på anden iteration, der kan nu oprettes projekter i databasen.

Skitser til design er lavet

05/12/2019

Lavet kravspecifikation. Herunder usecase-diagrammer, usecases og iterationsbeskrivelser. Vi har lavet begrebsbeskrivelser, og overordnede krav for produktet. Vi har også beskrevet produktets arkitektur.

09/01/2020

Iteration 3 færdiggjort. Lavet usecases for iteration 5.

14/01/2020

Arbejdet på iteration 4.

16/01/2020

Arbejdet videre på iteration 4. Tilføjet afsnit om sikkerhed, begyndt på styling.

23/01/2020

Iteration 4 færdiggjort.

28/01/2020

Iteration 5 færdiggjort.

04/02/2020

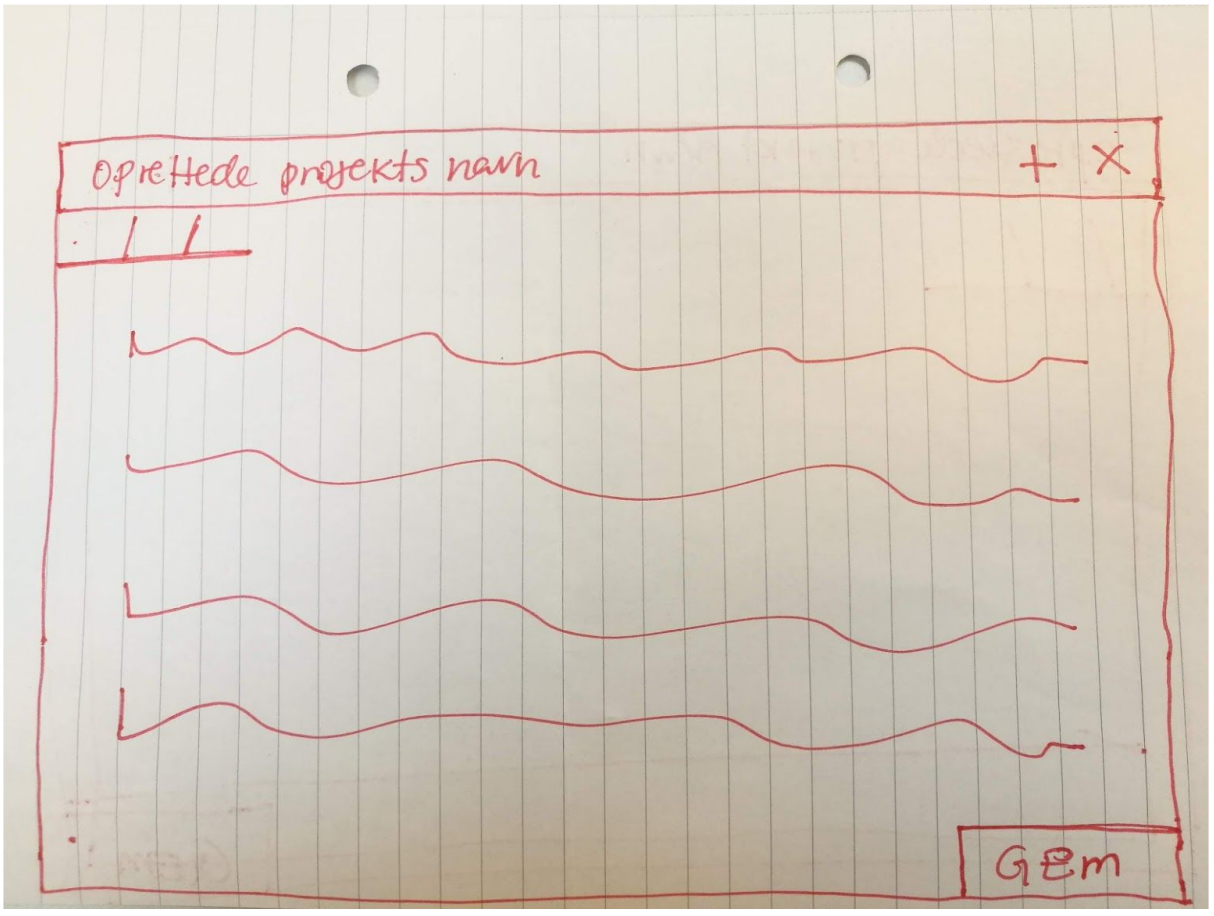
Begyndt på at beskrive implementeringen af systemet. Blev færdig med config.php, index.php og login.php.

06/02/2020

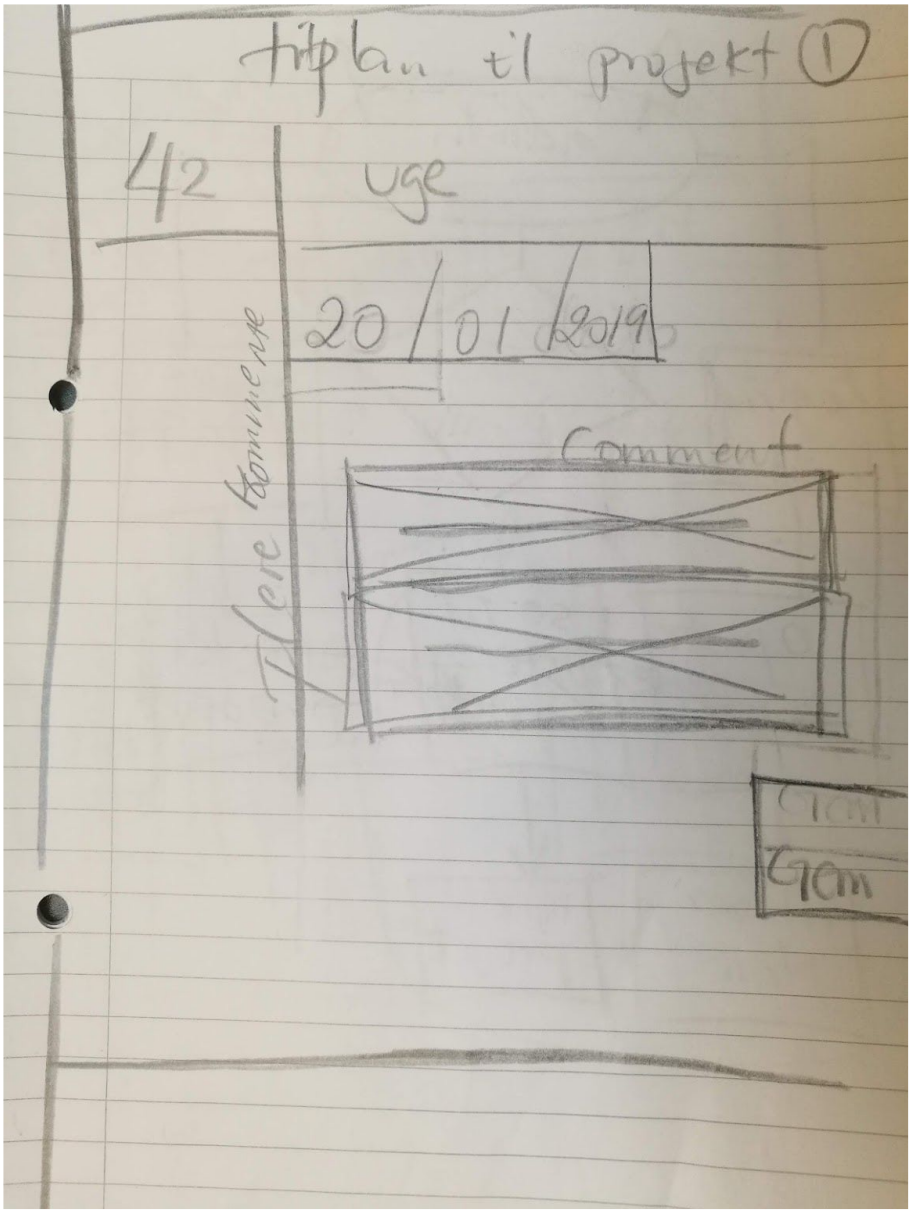
Fortsatte med at dokumentere implementation af systemet.

Bilag 3, Sketches og Mockups

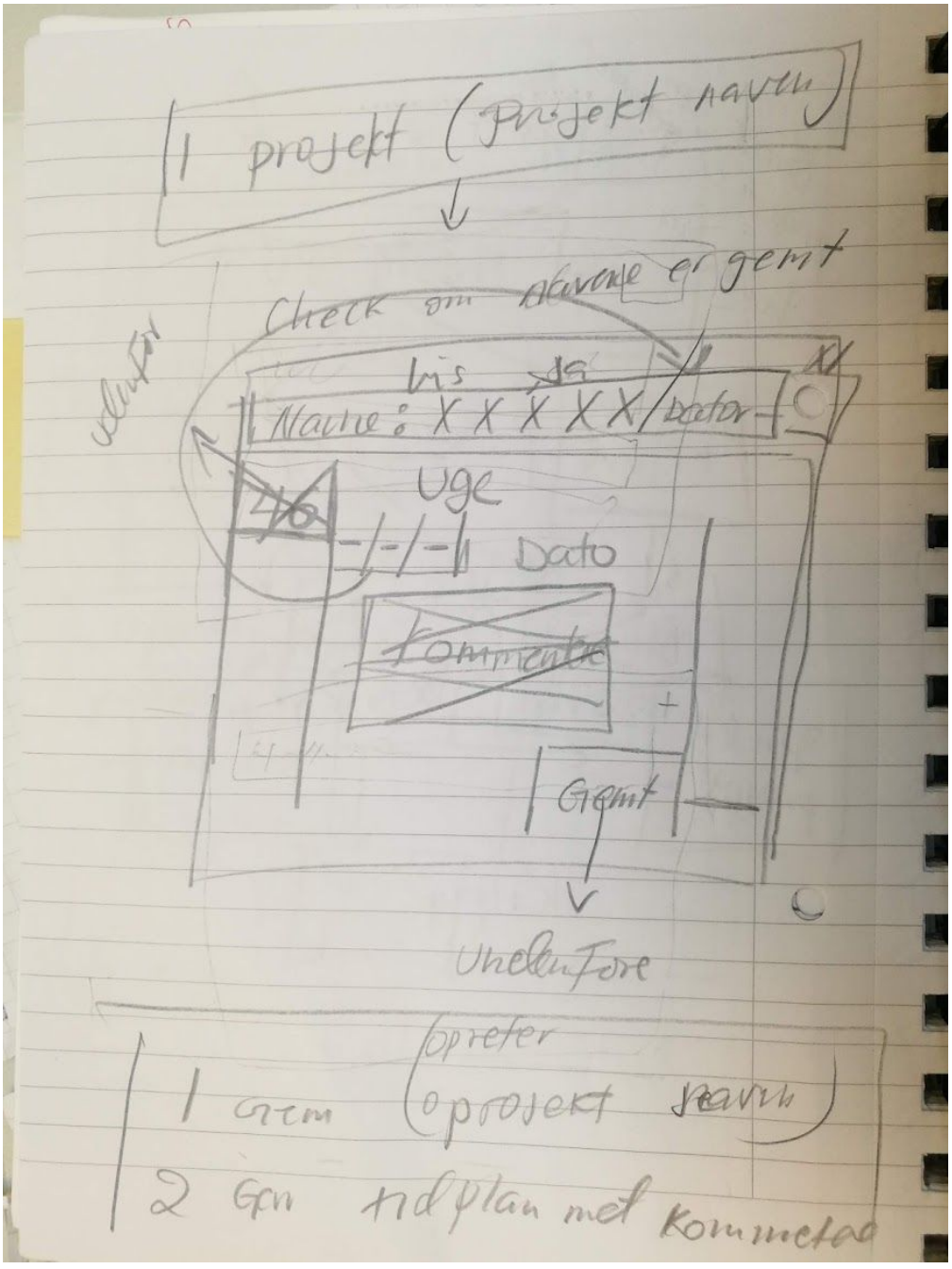
3.1



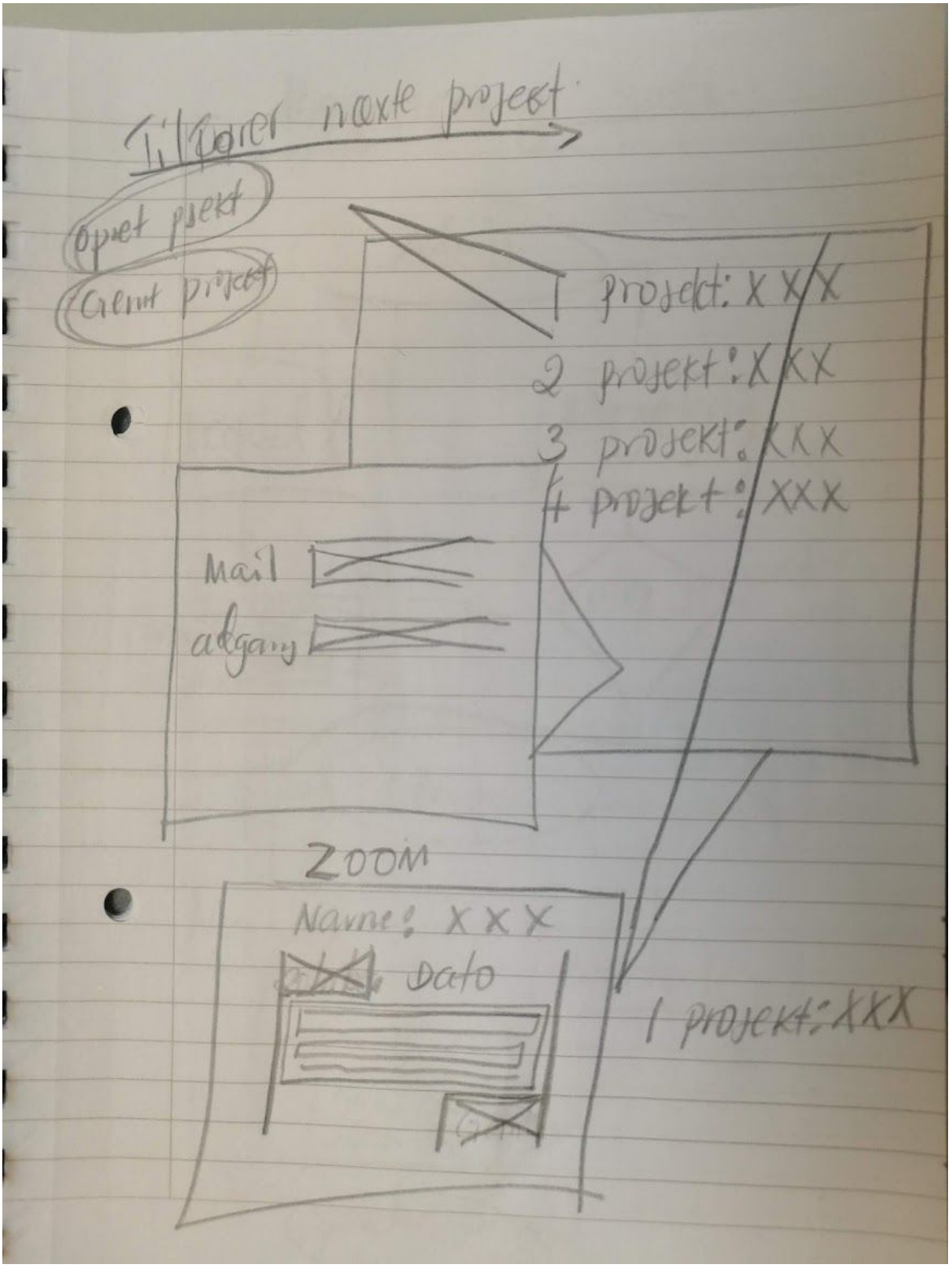
3.2



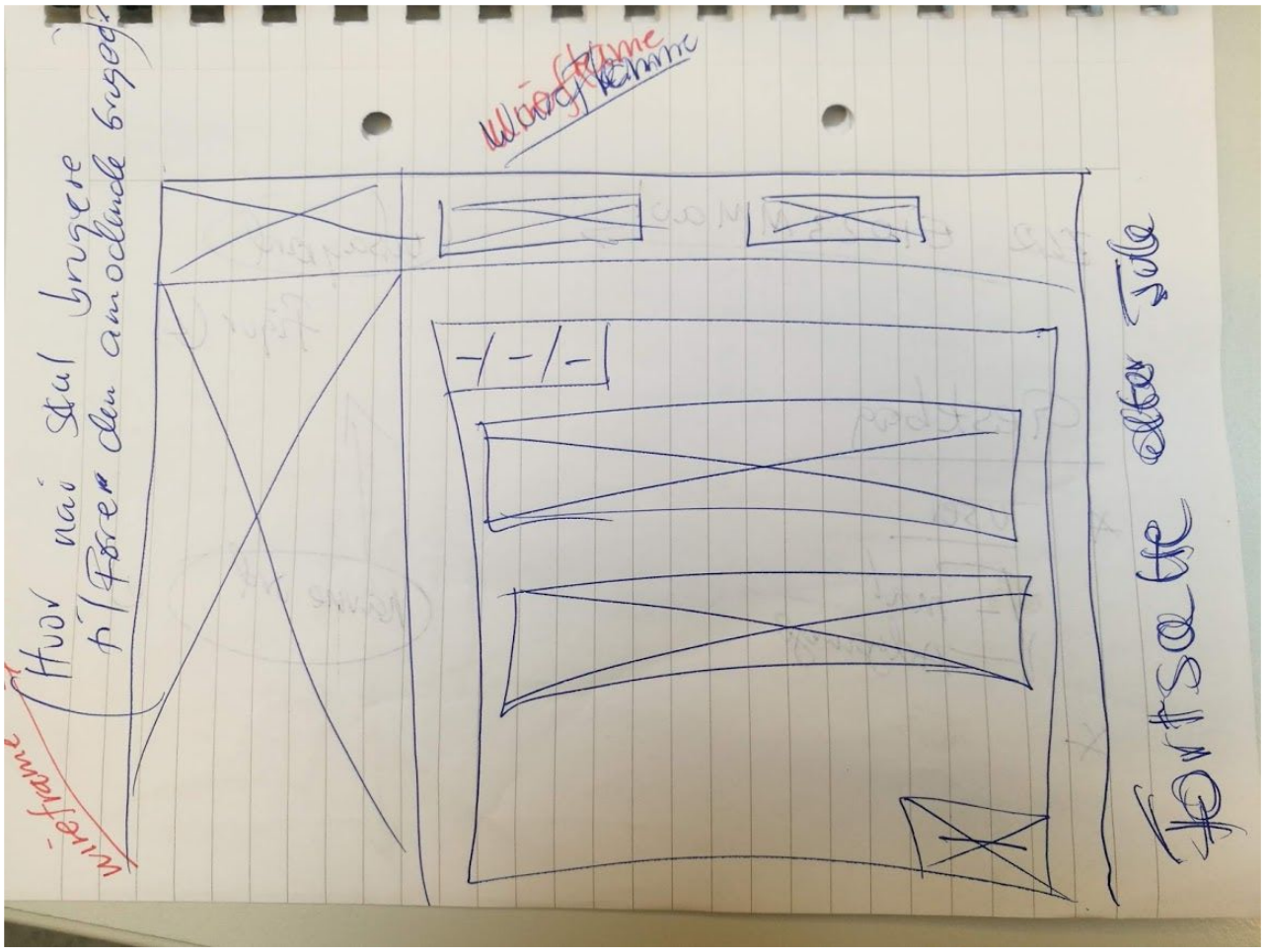
3.3


1 projekt (projekt navn)
Check om navnet er gemt
Vis
Navne: X X X X X
Dato
Uge
Dato
Gemt
Uheldtore
opreter
1 gem (projekt navn)
2 gm tidplan med kommetar

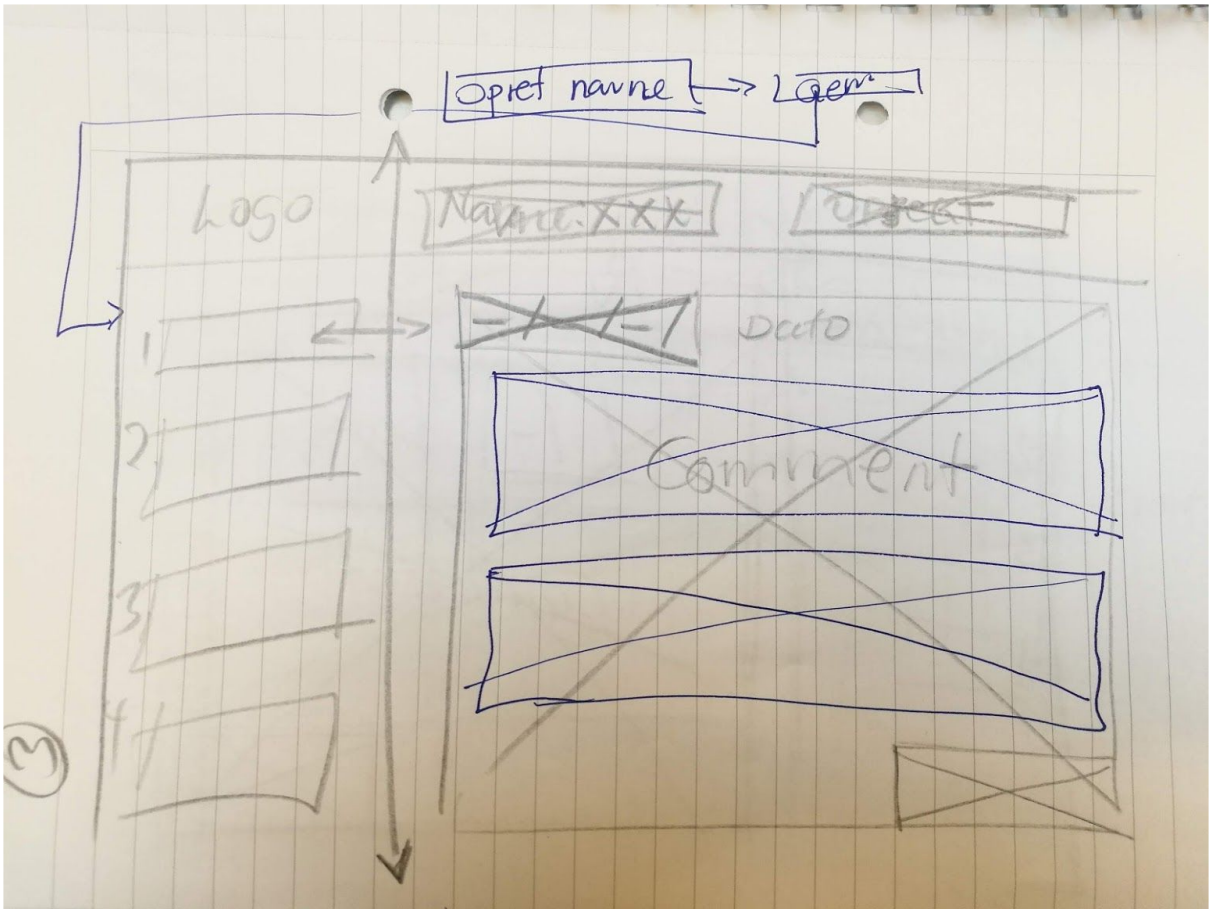
3.4


Tilføjer næste projekt
opret psekt
Slet projekt
1 prosdct: XXX
2 projekt: XXX
3 projekt: XXX
4 projekt: XXX
Mail
adgang
ZOOM
Navne: XXX
Dato
1 projekt: XXX

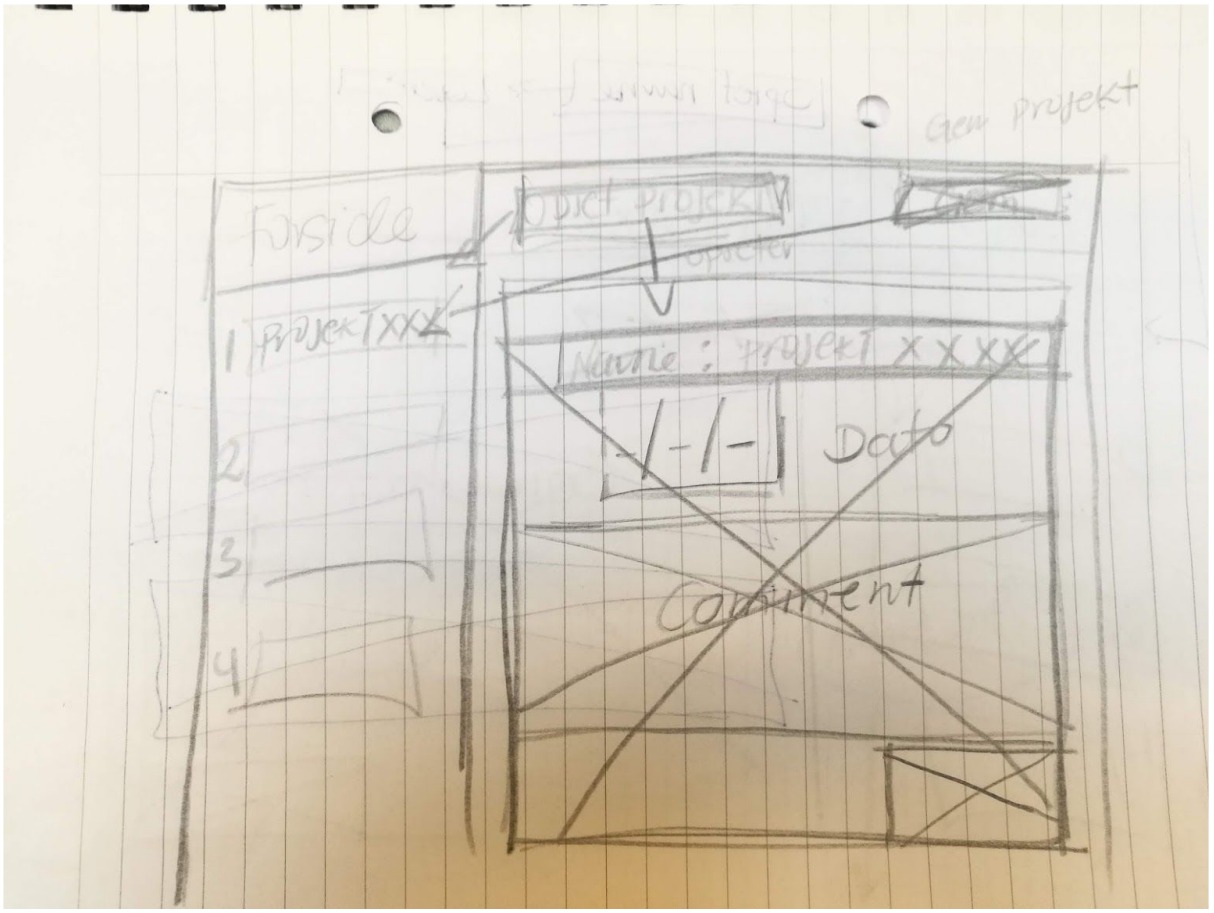
3.5



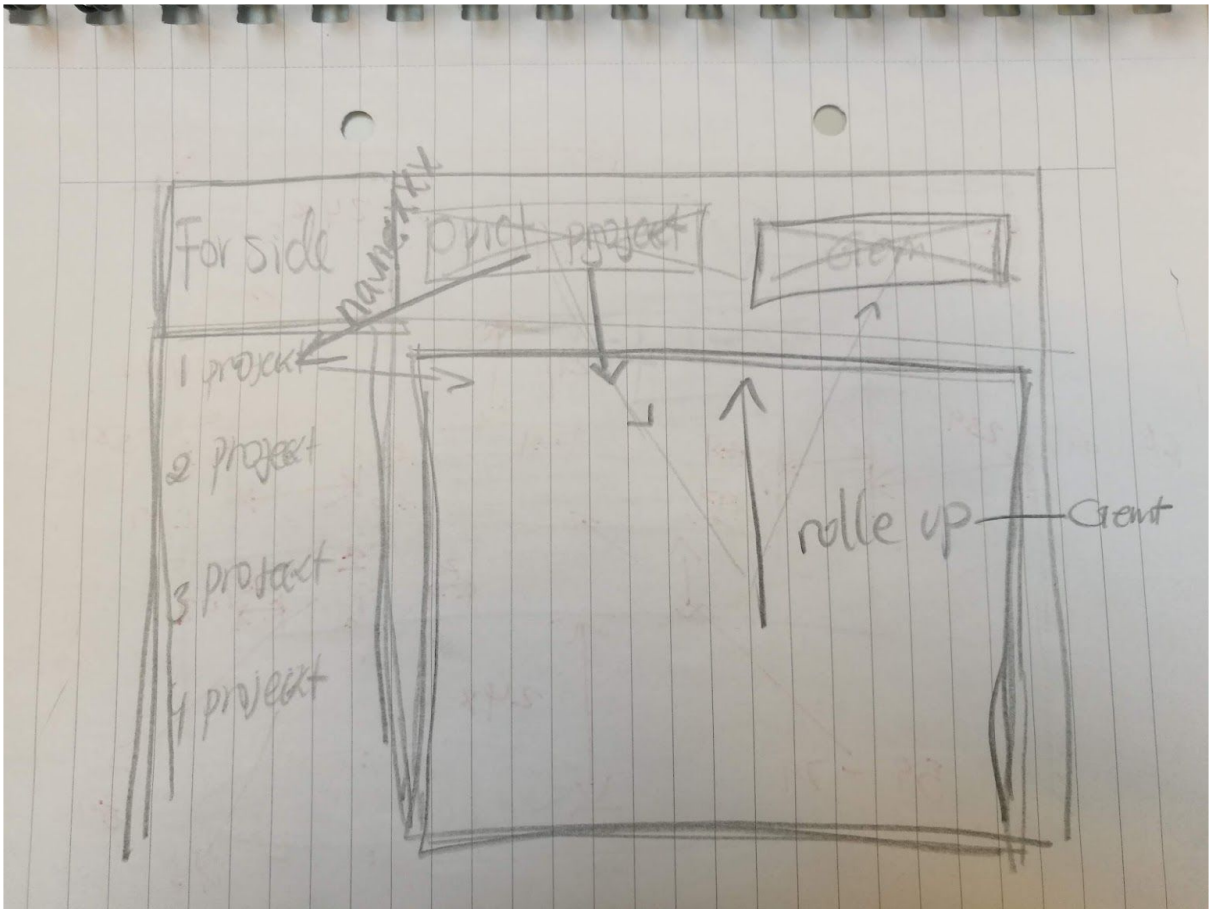
3.6



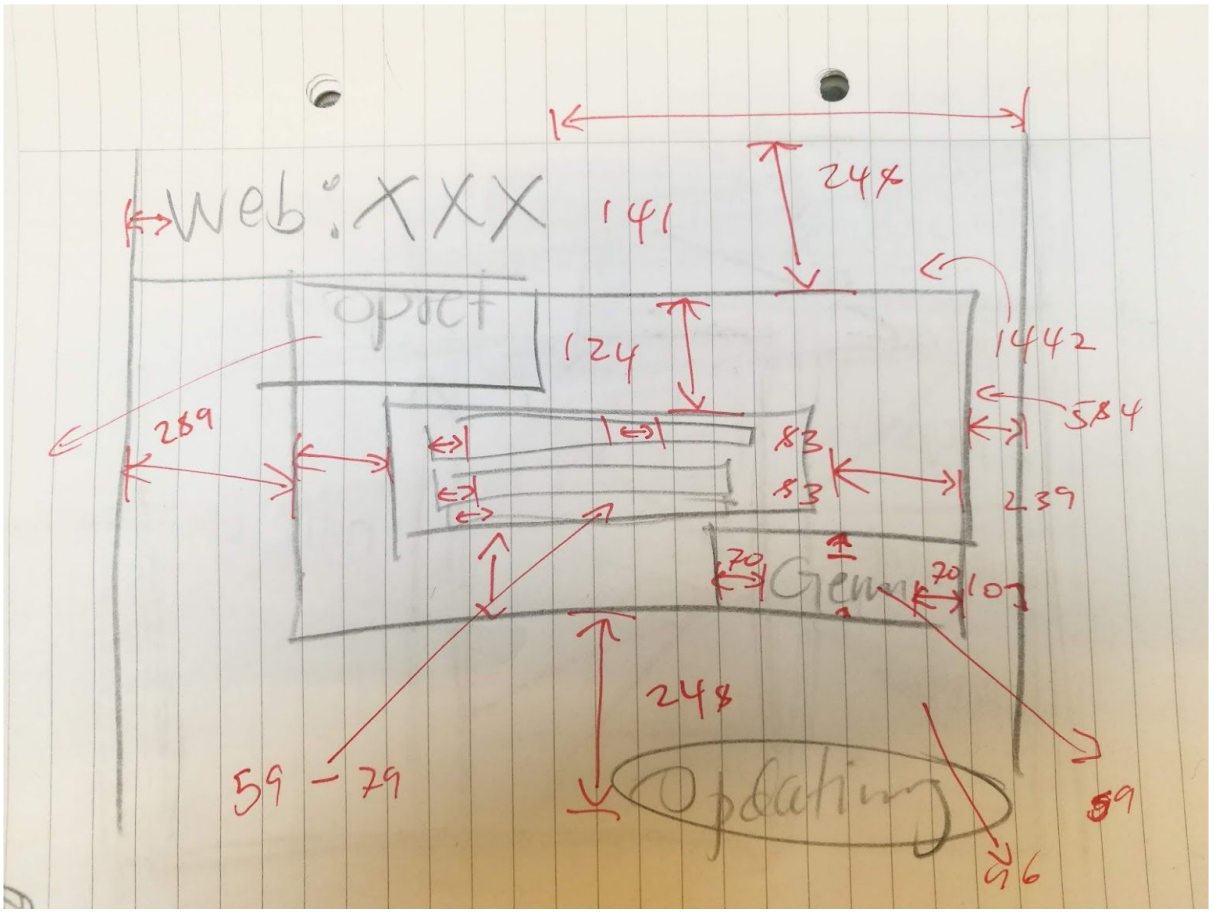
3.7



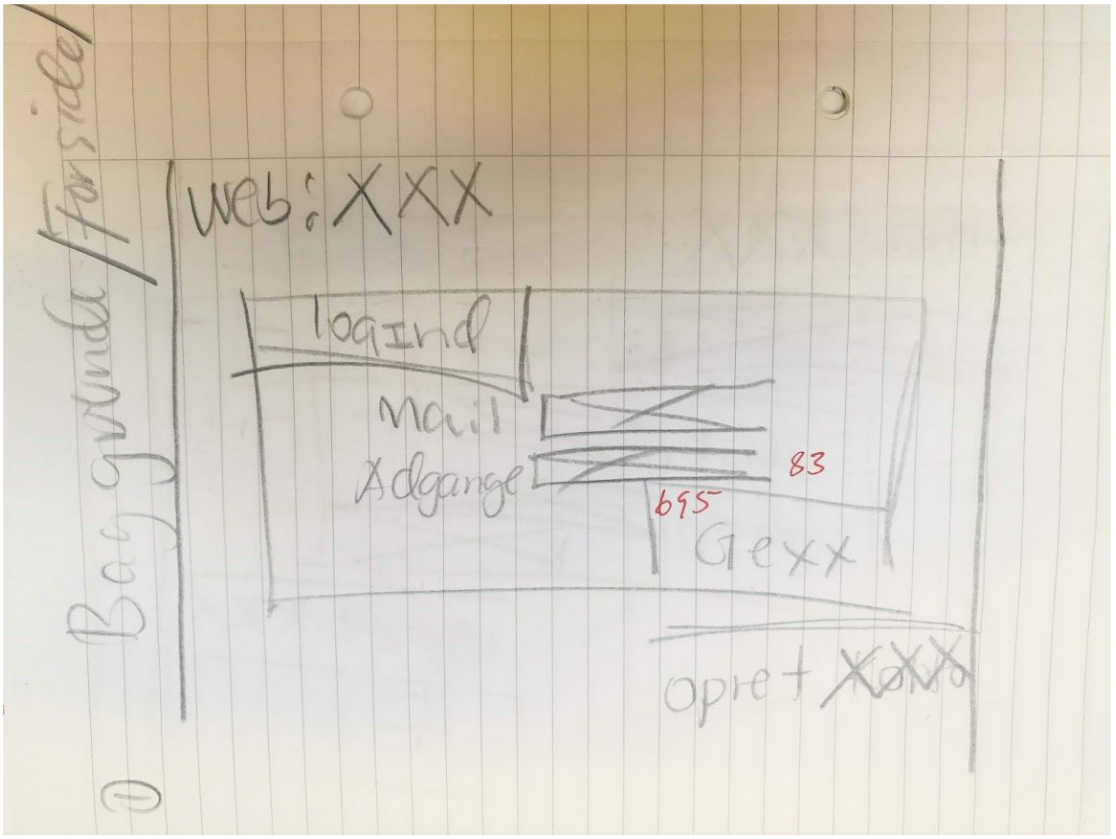
3.8


For side
navigation
1 project
2 project
3 project
4 project
rolle up
Client

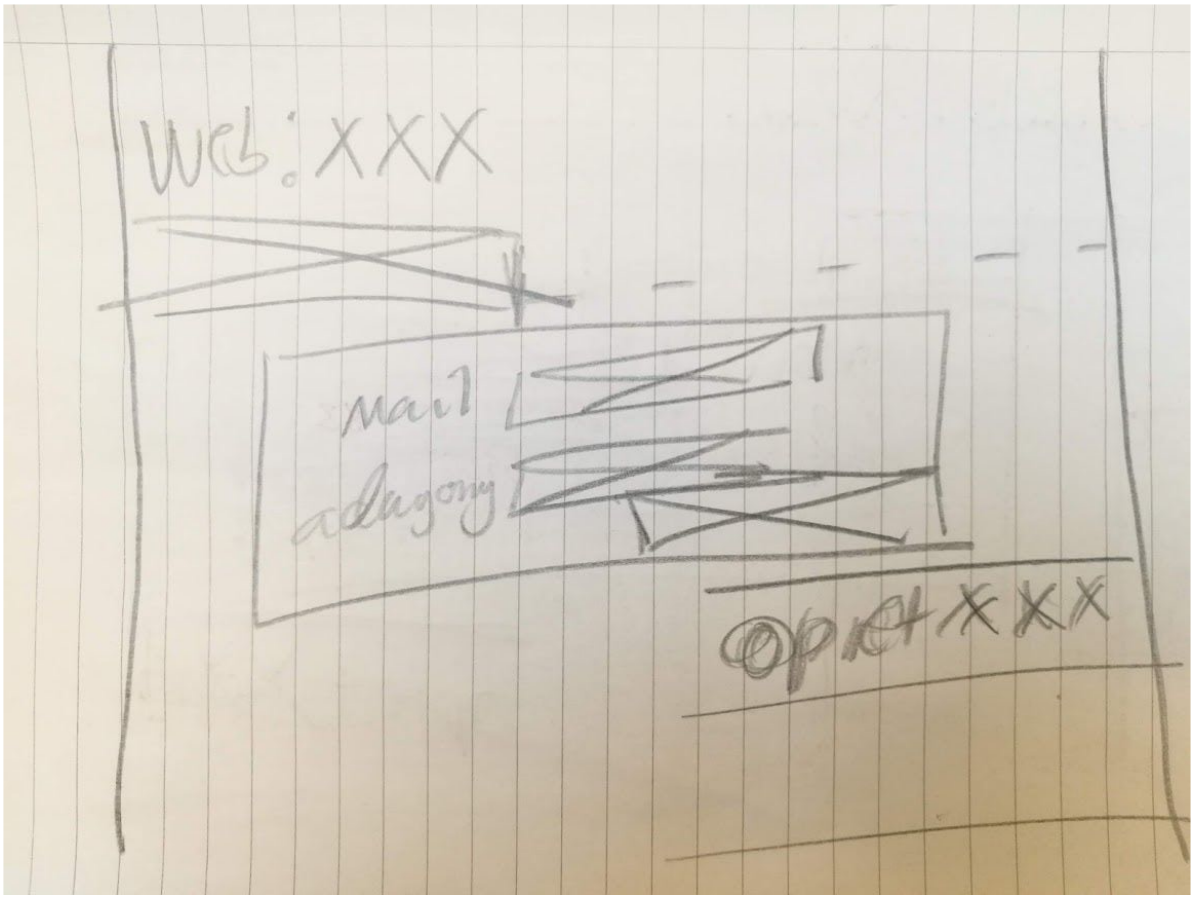
3.9


Web: XXX
141
248
124
1442
584
289
83
83
239
20
20
107
248
59 - 79
59
96

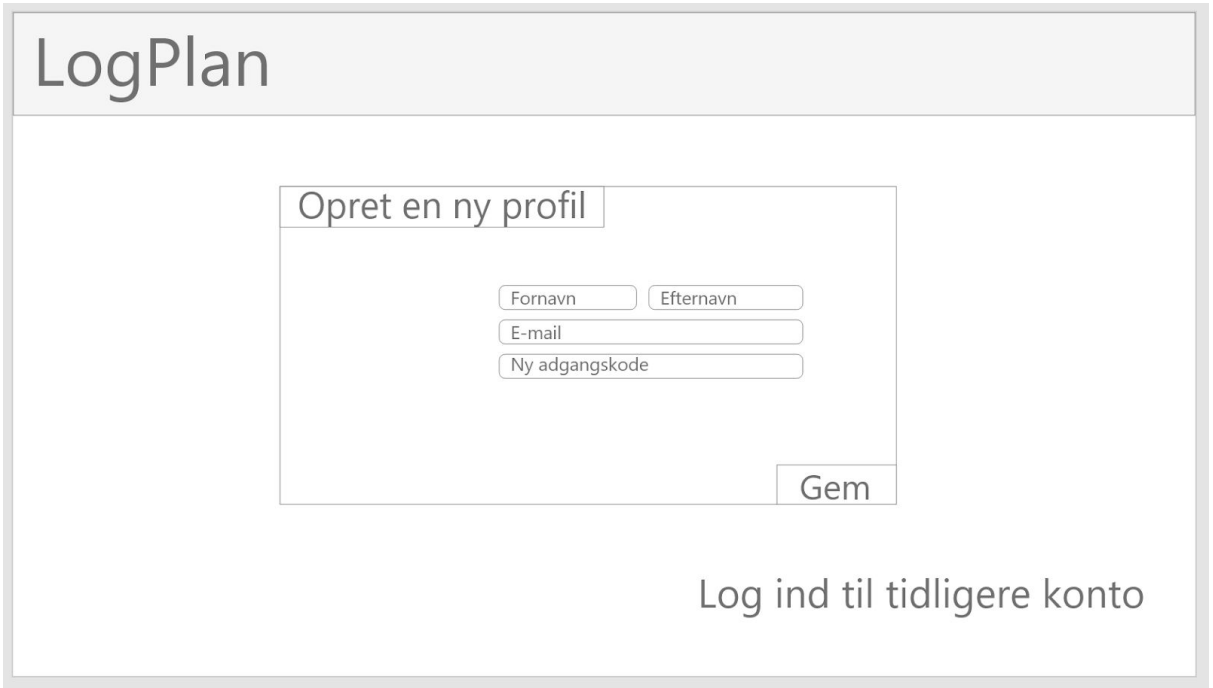
3.10


Baggrunds forside
Web: XXX
logInd
Mail
Adgange
83
695
Gexx
opret

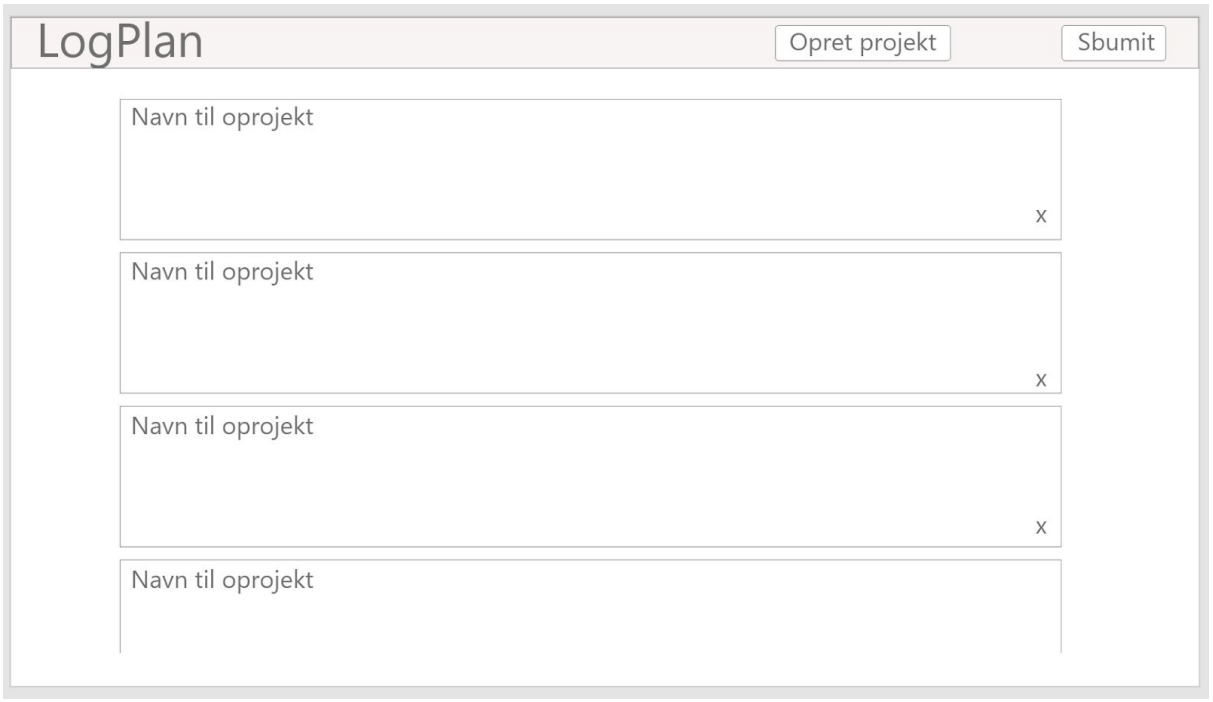
3.11


Web: XXX
mail
adgang
opret XXX

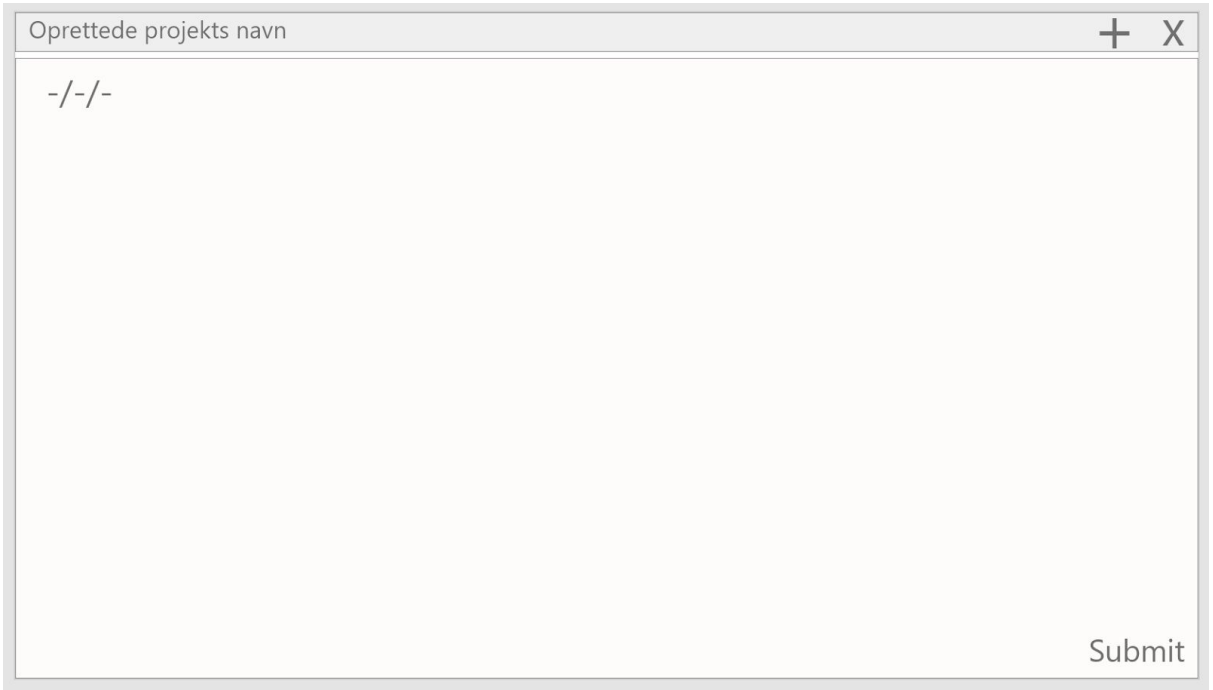
3.12


LogPlan
Opret en ny profil
Fornavn
Efternavn
E-mail
Ny adgangskode
Gem
Log ind til tidligere konto

3.13


LogPlan
Opret projekt
Sbumit
Navn til oprojekt
x
Navn til oprojekt
x
Navn til oprojekt
x
Navn til oprojekt

3.14


Oprettede projekts navn
-/-/-
Submit

Bilag 4, config.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "logplan";
//The information needed to log into the database

$conn = new mysqli($servername, $username, $password, $dbname);
//Create a connection to the database
mysqli_set_charset($conn,"utf8");
//Set the characterset to utf-8
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
//In case of failed connection
?>
```

Bilag 5, index.php

```php
<?php
// To have access to the sessions variables
session_start();
// Contains our connection to our database
include('config.php');
session_unset();
//Checks if a submit button that is inside the form, has been pushed.
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  //Checks if it is the login button
  if(isset($_POST["login"])) {
  header('location:login.php');
  }
  //There are only 2 buttons so if it is not the first then it is the
second.
  else {
    header('location:register.php');  }
  }
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/main.css">
    <title>Document</title>
</head>

<body>

  <div class="header">
    <h1>LogPlan</h1>
  </div>

  <div class="inputBox_index">
    <h1>Login or register</h1>

    <form method="POST">
      <input type="submit" name="login" value="Login">
      <input type="submit" name="register" value="Register">
    </form>
  </div>

</body>
</html>
```

Bilag 6, login.php

```
<?php
//Here a session is started, so any login-details can be saved
session_start();
//Connect to our database
include('config.php');

//Variables for checking login are created
$email="";
$pw="";
$cpw="";
$cemail="";

//If the mail-input and the pw-input are send, the following happens:
if(!empty( $_POST['email'] ) && !empty( $_POST['pw'] )) {
  //The email and password from the input are saved in the variables $email
and $pw
  $email = $_POST['email'];
  $pw = $_POST['pw'];
```

```
  //Here we check our database for a user with the email given as input
  $sql = "SELECT password FROM user WHERE email='$email';";
  $result = $conn->query($sql);
  $fetch = $result;
  $row = mysqli_fetch_assoc($fetch);

  //Here the hashed password of the user from the database, is saved in
$cpw, and cmail is set to the mail given as input
  $cpw = $row['password'];
  $cemail = $email;
}
  //Check if the mail is correct, and if the written password matches the
hashed password from the database
  if($email == $cemail && password_verify($pw, $cpw) == true) {
    //The mail is saved as a session-variable
    $_SESSION['email'] = $email;
    //Redirects to main.php
    header('location:main.php');
  }

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/main.css">
    <title>Document</title>
</head>
<body>

  <div class="header">
    <h1>LogPlan</h1>
  </div>



  <div class="inputBox_regIn">
    <h1>Login</h1>
    <form method="POST">
      <table class="table_regIn">

        <tr>
          <th>Email: <br/></th>
          <th><input type="text" name="email" placeholder="Enter email"
required/></th>
        </tr>
```

```
        <tr>
          <th>Password: <br/></th>
          <th><input type="password" name="pw" placeholder="Password"
required  /></th>
        </tr>

        <tr>
          <th colspan="2"><input type="submit" value="Login" /> </th>
        </tr>

      </table>
    </form>
  </div>

  <div>
    <button class="switch_regIn" type="button"
onclick="window.location.href='register.php'"
name="btnCancel">Register</button>
  </div>

</body>
</html>
```

Bilag 7, register.php

```
<?php
//We start a session here, which can be used to save login-details. We also
connect to our database with config.php
  session_start();
  include('config.php');

  if ($_SERVER["REQUEST_METHOD"] == "POST"){
    //If an input is send in a form with the method POST, if the input's
name is 'btnCreateUser', the following happens:
    if(isset($_POST['createUser'])){
      //The database is checked for users with the same email as what was
given as an input
      $email = mysqli_real_escape_string($conn, $_POST['email']);

      $sql = "SELECT email FROM user WHERE email = '$email'";
      $result = $conn->query($sql);

      $row = mysqli_fetch_assoc($result);
```

```
      $count = mysqli_num_rows($result);

      if($count == 1) {
        //If there is such a user, a failure-message is send to the webpage
        echo "<span class='errorText'>Unable to create user. The email is
already being used</span>";
      }else {
        //Otherwise, the info from the inputs are saved, the password is
hashed, and everything gets saved as a new user in our database, in the
table 'user'
        $password = mysqli_real_escape_string($conn,
password_hash($_POST['password'], PASSWORD_DEFAULT));

        $firstname = mysqli_real_escape_string($conn, $_POST['firstname']);

        $surname = mysqli_real_escape_string($conn, $_POST['surname']);

        $sql = "INSERT INTO user (email, password, firstname, surname)
VALUES ('$email', '$password', '$firstname', '$surname')";

        $conn ->query($sql);

        //At last the user gets redirected to login.php
        header("location:login.php");
      }
    }
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="css/main.css">
</head>
<body>

  <div class="header">
    <h1>LogPlan</h1>
  </div>

  <div class="inputBox_regIn">
    <form method="POST">
      <h1>Register</h1>
```

```
      <table class="table_regIn">

        <tr>
          <th>Email: <br></th>
          <th><input type="email" name="email" placeholder="Enter email"
required></th>
        </tr>

        <tr>
          <th>Firstname: <br></th>
          <th><input type="text" name="firstname" placeholder="Firstname"
required></th>
        </tr>

        <tr>
          <th>Surname: <br></th>
          <th><input type="text" name="surname" placeholder="Surname"
required></th>
        </tr>

        <tr>
          <th>Password: <br></th>
          <th><input type="password" name="password" placeholder="Password"
required></th>
        </tr>

        <tr>
          <th colspan="2"><input type="submit" value="Create"
name="createUser" /> </th>
        </tr>

      </table>
    </form>
  </div>

  <div>
    <button class="switch_regIn" type="button"
onclick="window.location.href='login.php'" name="btnCancel">Login</button>
  </div>

</body>

</html>
```

Bilag 8, main.php

```php
<?php
session_start();
//This file has our connection to our database
include('config.php');

//Check if we have a session called email. This way we block users from
changing the url and trying to skip login.
if(!isset($_SESSION['email'])){
  header('location: index.php');
}

function userID($email, $conn){
  $sql = "SELECT id FROM user WHERE email='$email';";
  $result = $conn->query($sql);
  $fetch = $result;
  $row = mysqli_fetch_assoc($fetch);
  return $row['id'];
}

if ($_SERVER["REQUEST_METHOD"] == "POST")  {
  //Checks if we have pushed the button named conversation
  if(isset($_POST['newProject'])){
    $user = userID($_SESSION['email'], $conn);
    $name = $_POST['projectName'];

    $sql = "INSERT INTO project (name, user_id) VALUES ('$name', '$user');";
    $conn ->query($sql);

    $sql = "SELECT MAX(id) FROM project WHERE user_id=$user;";
    $result = $conn->query($sql);
    $fetch = $result;
    $row = mysqli_fetch_assoc($fetch);
    $project = $row['MAX(id)'];

    $sql = "INSERT INTO user_project (user_id, project_id) VALUES ('$user',
'$project');";
    $conn ->query($sql);


    }else if(isset($_POST['open'])){

      $hentid = $_POST['openid'];
      $_SESSION['project'] = $hentid;
      header('location:project.php');
```

```
    }else if(isset($_POST['dlt'])){
      $commid = $_POST['dltid'];
      $sql = "DELETE FROM project WHERE id=$commid;";
      $conn->query($sql);
    }
  }

?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="css/main.css">
  <title>Document</title>
</head>

<body>

  <div class="header">

    <h1>LogPlan</h1>

    <form method="POST" class="header_Form">

      <input type="text" name="projectName">
      <input type="submit" name="newProject" value="Create Project">

    </form>

    <a href="index.php">
      <div class="logout">
        <p>Logout</p>
      </div>
    </a>

  </div>

  <?php
  $userid = userID($_SESSION['email'], $conn);
  $sql = "SELECT * FROM project INNER JOIN user_project ON project.id =
user_project.project_id WHERE user_project.user_id = $userid;";
  $result = $conn->query($sql);

  if($result->num_rows > 0){
  ?>
```

```
  <div class="projectTable">
    <p>Projects</p>
  <?php
  // løb alle rækker igennem
  while($row = $result->fetch_assoc()) {
  ?>
    <div>
      <div class="project">

      <?php
        $name = $row['name'];
        $id = $row['id'];
        $sql = "SELECT user_id FROM project WHERE id=$id;";
        $result2 = $conn->query($sql);
        $row2 = mysqli_fetch_assoc($result2);
        $creator = $row2['user_id'];
        echo "<h1>$name</h1> <form method='POST'> <input type='submit'
name='open' value='Open' /><input type='hidden' value='$id'
name='openid'/></form></div>";
        if($creator == $userid){
          echo "<div class='deletThis'><form method='POST'><input
type='submit' name='dlt' value='Delete'><input type='hidden' name='dltid'
value='$id'></form></div>";
      ?>
    </div>
  <?php
  }
  }
  ?>

  </div>

  <?php
  }
  ?>

<body>
</html>
```

Bilag 9, project.php

```
<?php
session_start();
//This file has our connection to our database
```

```
include('config.php');

//Check if we have a session called email. This way we block users from
changing the url and trying to skip login.
if(!isset($_SESSION['email'])){
  header('location: index.php');
}

function projectName($id, $conn){
  $sql = "SELECT name FROM project WHERE id='$id';";
  $result = $conn->query($sql);
  $fetch = $result;
  $row = mysqli_fetch_assoc($fetch);
  return $row['name'];
}

function userID($email, $conn){
  $sql = "SELECT id FROM user WHERE email='$email';";
  $result = $conn->query($sql);
  $fetch = $result;
  $row = mysqli_fetch_assoc($fetch);
  return $row['id'];
}

if ($_SERVER["REQUEST_METHOD"] == "POST")  {
  if(isset($_POST['add'])){
    $input = $_POST['mail'];
    $inputid = userID($input, $conn);

    if($inputid == null){
      echo "<div class='error'>user does not exist</div>";
    } else{
      $proid = $_SESSION['project'];
      $sql = "SELECT * FROM user_project WHERE user_id='$inputid' AND
project_id='$proid';";
      $result = $conn->query($sql);
      if($result->num_rows > 0){
        echo "allerede del af projekt";
      } else{
        $sql = "INSERT INTO user_project (user_id, project_id) VALUES
('$inputid', '$proid');";
        $conn ->query($sql);
      }
    }

  } else if(isset($_POST['time-element'])){

    $start = $_POST['start'];
```

```
    $end = $_POST['end'];
    $userid = userID($_SESSION['email'], $conn);
    $proid = $_SESSION['project'];
    $sql = "INSERT INTO time_element (user_id, project_id, end_time,
start_time) VALUES ('$userid', '$proid', '$end', '$start');";
    $conn ->query($sql);

  } else if(isset($_POST['descript'])){

    $hentid = $_POST['time_element_id'];
    $_SESSION['time-element'] = $hentid;
    header('location:time_element.php');
}else if(isset($_POST['dlt'])){
  $commid = $_POST['time_element_id'];
  $sql = "DELETE FROM time_element WHERE id=$commid;";
  $conn->query($sql);
}
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/main.css">
    <title>Document</title>
</head>

<body>

  <div class="header">

    <a href="main.php"><h1>LogPlan</h1></a>

    <a href="index.php">
      <div class="logout">
        <p>Logout</p>
      </div>
    </a>

  </div>
  <div class="projectView">
    <h1>
    <?php
    $prjname = projectName($_SESSION['project'], $conn);
    echo $prjname;
```

```
    ?>
    </h1>
    <div class="projectInput">
      <div>
        <form method='POST'>
          <input type="text" name="mail">
          <input type="submit" name="add" value="Add to project">
        </form>
      </div>

      <div>
        <form method='POST'>
          <input type='date' name='start'>
          <input type='date' name='end'>
          <input type='submit' name='time-element' value='Create
time-element'>
        </form>
      </div>
    </div>

      <?php
    $proid = $_SESSION['project'];
    $sql = "SELECT * FROM time_element WHERE project_id = $proid;";
    $result = $conn->query($sql);

    if($result->num_rows > 0){
      ?>
    <center><div class="projectComment">
      <table>
        <tr>
          <th colspan="2">Time-elements</th>
        </tr>
      <?php
      // løb alle rækker igennem
      while($row = $result->fetch_assoc()) {
      ?>
        <tr>
      <?php
      $time = $row['start_time'];
      $id = $row['id'];
      echo "<td>$time</td><form method='POST'><td><input type='submit'
class='open' name='descript' value='Open' /><input type='submit'
class='delete' name='dlt' value='Delete' /><input type='hidden' value='$id'
name='time_element_id'/></td></form>";
      ?>
        </tr>

    <?php
```

```
    }
    ?>

      </table>
    </div></center>
<?php
}
?>
  </div>
</body>
</html>
```

Bilag 10, time_element.php

```
<?php
session_start();
//This file has our connection to our database
include('config.php');

//Check if we have a session called email. This way we block users from
changing the url and trying to skip login.
if(!isset($_SESSION['email'])){
  header('location: index.php');
}

function userID($email, $conn){
    $sql = "SELECT id FROM user WHERE email='$email';";
    $result = $conn->query($sql);
    $fetch = $result;
    $row = mysqli_fetch_assoc($fetch);
    return $row['id'];
  }

$timeid = $_SESSION['time-element'];

if ($_SERVER["REQUEST_METHOD"] == "POST")  {
    if(isset($_POST['back'])){
        header('location:project.php');
    } else if(isset($_POST['adddes'])){

        $des = $_POST['dscrpt'];
        $sql = "UPDATE time_element SET description = '$des' WHERE id =
'$timeid';";
        $conn->query($sql);
    }else if(isset($_POST['postComment'])){
```

```
        $commenttext = $_POST['commentText'];
        $userid = userID($_SESSION['email'], $conn);
        $sql = "INSERT INTO comment (user_id, time_element_id, text) VALUES
('$userid', '$timeid', '$commenttext');";
        $conn->query($sql);
    }else if(isset($_POST['dlt'])){
      $commid = $_POST['dltid'];
      $sql = "DELETE FROM comment WHERE id=$commid;";
      $conn->query($sql);
    }
}
$sql = "SELECT * FROM time_element WHERE id='$timeid';";
$result = $conn->query($sql);
$row = $result->fetch_assoc();
$dscrpt = $row['description'];
$start = $row['start_time'];
$end = $row['end_time'];

$sql ="SELECT * FROM comment WHERE time_element_id='$timeid';";
$result = $conn->query($sql);


?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/main.css">
    <title>Document</title>
</head>
<body>

  <div class="header">

    <a href="main.php"><h1>LogPlan</h1></a>

    <a href="index.php">
      <div class="logout">
        <p>Logout</p>
      </div>
    </a>

    <div class="back">
      <form method='POST'>
      <input type='submit' name='back' value='Back'>
```

```
    </form>
  </div>

</div>
<h1 class="dateName">
  <?php

  echo $start;
  ?>
  <br>
  <?php
  echo $end;
  ?>
  <br>
</h1>

<div class="description">
  <p>Description</p>
  <form method='POST'>
  <br>
  <?php
  echo "<textarea name='dscrpt'>$dscrpt</textarea>";
  ?>
  <input type='submit' name='adddes' value='Save description'>
  </form>
</div>

<div class="writeComment">
  <p>LogComments</p>
  <form method='POST'>
    <textarea name='commentText'>Write your LogComment here...</textarea>
    <input type='submit' name='postComment' value='Post comment'>
  </form>
</div>

<table class="comments">

<?php
if($result->num_rows > 0){
// løb alle rækker igennem
while($row = $result->fetch_assoc()) {
?>

  <tr>

<?php
$poster = $row['user_id'];
$id = $row['id'];
```

```
  $sql = "SELECT firstname, surname FROM user WHERE id='$poster';";
  $result2 = $conn->query($sql);
  $row2 = $result2->fetch_assoc();
  $fname = $row2['firstname'];
  $lname = $row2['surname'];
  $userid = userID($_SESSION['email'], $conn);

  $comment = $row['text'];
  echo "<td>$fname $lname</td><td style='width:100%;'>$comment</td>";
  if($poster == $userid){
    echo "<td><form method='POST'><input type='submit' name='dlt'
value='Delete'><input type='hidden' name='dltid' value='$id'></form></td>";
  }
  ?>

    </tr>

  <?php
}

?>
 </table>
<?php

}else{
    echo "<div class='empty'>There are no LogComments on this
time-element</div>";
}
?>

</body>
</html>
```

Bilag 11, main.css

```
* {
margin: 0;
padding: 0;
text-decoration: none;
box-sizing: border-box;
}

.error {
    background-color: #57a773;
    padding-left: 4px;
```

```
}

.header {
    background-color: #EE6055;
    max-height: 92.8px;
}

.header h1 {
    font-size: 80px;
    color: #ff9b85;
    display: inline-block;
}

.header_Form {
    display: inline-block;
    position: absolute;
    left: 40%;
    top: 35px;
    vertical-align: super;
}

.header_Form input[type=text] {
    height: 36px;
    width: 300px;
    margin-right: 12px;
    border: 1px solid black;
}

.header_Form input[type=submit] {
    height: 36px;
    padding: 4px 6px;
    font-size: large;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
}

.logout {
    margin: 30px 30px;
```

```
    padding: 3px 40px;
    float: right;
    font-size: x-large;
    background-color: #ff9b85;
    border-radius: 4px;
    color: white;
}

.back {
    margin: 25px 30px;
    padding: 3px 30px;
    float: right;
    font-size: x-large;
    background-color: #ff9b85;
    border-radius: 4px;
}

.back input[type=submit] {
    height: 36px;
    padding: 4px 4px;
    font-size: large;
    border: none;
    cursor: pointer;
    background-color: #ff9b85;
    color: white;
}

body, html {
    background-color: #AAf683
}

.inputBox_index {
    background-color: #ffd97d;
    margin: 100px 30%;
    width: 40%;
    justify-content: center;
    text-align: center;
}

.inputBox_index h1 {
```

```
    color: white;
    font-size: 30px;
}

.inputBox_index input[type=submit] {
    font-size: x-large;
    background-color: #57a773;
    border: none;
    color: white;
    min-width: 44%;
    min-height: 170px;
    margin: auto;
    margin-top: 10px;
    margin-bottom: 22px;
    cursor: pointer;
}

.inputBox_regIn {
    background-color: #ffd97d;
    margin: 100px 40%;
    width: 20%;
    justify-content: center;
    text-align: center;
}

.inputBox_regIn h1 {
    color: white;
    font-size: 30px;
}

.table_regIn {
    margin: auto;
    margin-top: 10px;
    text-align: right;
}

.table_regIn input[type=text] {
    padding: 2px 4px;
    width: 100%;
}
```

```
.table_regIn input[type=password] {
    padding: 2px 4px;
    width: 100%;
}

.table_regIn input[type=email] {
    padding: 2px 4px;
    width: 100%;
}

.table_regIn th {
    padding-bottom: 10px;
}

.inputBox_regIn input[type=submit] {
    font-size: x-large;
    background-color: #57a773;
    border: none;
    color: white;
    width: 100%;
    margin: auto;
    margin-top: 4px;
    margin-bottom: 8px;
    padding: 10px;
    cursor: pointer;
}

.switch_regIn {
    position: fixed;
    bottom: 10px;
    right: 10px;
    font-size: 120px;
    background-color: transparent;
    border: none;
    color: white;
    cursor: pointer;
    text-shadow: 3px 2px #57a773;
}
```

```
.projectTable {
    margin: 100px 18%;
}

.projectTable p{
    font-size: 70px;
    color: white;
    text-shadow: 3px 2px #57a773;
}

.project {
    padding-left: 4px;
    background-color: #ffd97d;
    margin-bottom: 18px;
    border-radius: 14px;
    border-style: groove;
    display: inline-block;
    width: 80%;
}

.project h1 {
    display: inline-block;
    color: white;
}

.project form {
    display: inline-block;
}

.project input[type=submit] {
    font-size: x-large;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
    float: right;
    padding: 3px 40px;
    border-radius: 4px;
}
```

```
.project form {
    float: right;
    margin: 4px 4px;
}

.deletThis {
    display: inline-block;
}

.deletThis input[type=submit] {
    font-size: x-large;
    background-color: #EE6055;
    border: none;
    color: white;
    cursor: pointer;
    padding: 3px 3px;
    border-radius: 4px;
    margin-left: 4px;
}

.projectView {
    margin: 100px 18%;
    text-align: center;
}

.projectView h1 {
    font-size: 70px;
    color: white;
    text-shadow: 3px 2px #57a773;
}

.projectInput {
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
    margin: 0px 25%;
}

.projectInput input[type=text] {
```

```
    padding: 2px 4px;
}

.projectInput input[type=submit] {
    padding: 2px 4px;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
}

.projectInput input[type=date] {
    padding: 2px 4px;
}

.projectComment {
    margin: 18px 25%;
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
}

.projectComment th {
    color: gray;
}

.open {
    padding: 2px 4px;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
    margin: 0px 13px;
}

.delete {
    padding: 2px 4px;
    background-color: #EE6055;
    border: none;
```

```
    color: white;
    cursor: pointer;
}

.dateName {
    margin-top: 80px;
    margin-left: 40%;
    color: white;
}

.description {
    margin: 23px 33%;
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
}

.description p {
    color: gray;
    font-size: large;
    margin-bottom: -15px;
}

.description textarea {
    height: 69px;
    width: 100%;
}

.description input[type=submit] {
    padding: 2px 4px;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
    font-size: large;
    border-radius: 3px;
    margin-top: 3px;
}
```

```
.writeComment {
    margin: 23px 35%;
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
}

.writeComment p {
    color: gray;
    font-size: large;
}

.writeComment textarea {
    width: 100%;
}

.writeComment input[type=submit] {
    padding: 2px 4px;
    background-color: #57a773;
    border: none;
    color: white;
    cursor: pointer;
    font-size: large;
    border-radius: 3px;
    margin-top: 3px;
}

.comments {
    margin: 0px 25%;
}

.comments td {
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
}

.comments input[type=submit] {
```

```
    padding: 2px 4px;
    background-color: #EE6055;
    border: none;
    color: white;
    cursor: pointer;
    border-radius: 3px;
}

.empty {
    margin: 0px 35%;
    padding: 6px;
    background-color: #ffd97d;
    border-radius: 14px;
    border-style: groove;
}
```

Bilag 12, logplan.sql

```
-- phpMyAdmin SQL Dump
-- version 4.9.2
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Feb 07, 2020 at 08:44 PM
-- Server version: 10.4.11-MariaDB
-- PHP Version: 7.2.26

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `logplan`
--

-- --------------------------------------------------------
```

```
--
-- Table structure for table `comment`
--

CREATE TABLE `comment` (
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `time_element_id` int(11) NOT NULL,
  `text` varchar(2056) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `comment`
--

INSERT INTO `comment` (`id`, `user_id`, `time_element_id`, `text`) VALUES
(28, 2, 1, 'Write your LogComment here...'),
(29, 2, 1, 'Write your LogComment here...'),
(30, 2, 1, 'Write your LogComment here...'),
(31, 2, 1, 'Write your LogComment here...'),
(43, 1, 2, 'Write your LogComment here...'),
(44, 1, 2, 'Write your LogComment here...'),
(45, 1, 2, 'Write your LogComment here...'),
(46, 1, 2, 'Write your LogComment here...'),
(47, 1, 2, 'Write your LogComment here...'),
(48, 1, 13, 'William er gennemsnitlig'),
(49, 1, 13, 'Jeg har færdiggjort iteration 5 i dag'),
(50, 1, 13, 'Write your LogComment here...');

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

--
-- Table structure for table `project`
--

CREATE TABLE `project` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `project`
--

INSERT INTO `project` (`id`, `name`, `user_id`) VALUES
(13, '', 3),
```

```
(14, '1', 3),
(22, 'halløj', 2),
(23, 'jeg hader william', 1);

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

--
-- Table structure for table `time_element`
--

CREATE TABLE `time_element` (
  `id` int(11) NOT NULL,
  `project_id` int(11) NOT NULL,
  `description` varchar(2056) NOT NULL,
  `user_id` int(11) NOT NULL,
  `start_time` datetime NOT NULL,
  `end_time` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `time_element`
--

INSERT INTO `time_element` (`id`, `project_id`, `description`, `user_id`,
`start_time`, `end_time`) VALUES
(2, 12, 'Vi skal bage kage', 1, '2020-12-31 23:58:00', '2020-01-01
00:01:00'),
(3, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(4, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(5, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(6, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(7, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(8, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(9, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(10, 12, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(11, 22, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(12, 23, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(13, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(14, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(15, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(16, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(17, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(18, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(19, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(20, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(21, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(22, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(23, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
```

```
(24, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(25, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00'),
(26, 14, '', 1, '0000-00-00 00:00:00', '0000-00-00 00:00:00');

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

--
-- Table structure for table `user`
--

CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `email` varchar(128) NOT NULL,
  `firstname` text NOT NULL,
  `surname` text NOT NULL,
  `password` varchar(512) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`id`, `email`, `firstname`, `surname`, `password`)
VALUES
(1, 'erikbuur@hotmail.com', 'Erik', 'Christensen',
'$2y$10$0ZK22PIVpdktbNg3BUQZwu.r6hlG.wOhwN6lTkcNa1zUN02n99pY6'),
(2, 'julemand@jul.dk', 'Julemand', 'Julendal',
'$2y$10$wk5g6wHVDYaSMCqTy.BJUuS8lNYsrCCeeP/9HFbjG2bZJ/DgW9Cx2'),
(3, 'olivergeneser1@gmail.com', 'Oliver', 'Geneser',
'$2y$10$lVjElyri2LKWkpxN2mdVUutHNGLTwjir2Cnc9rQZWYhKQ.BWpbXQO');

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

--
-- Table structure for table `user_project`
--

CREATE TABLE `user_project` (
  `user_id` int(11) NOT NULL,
  `project_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `user_project`
--

INSERT INTO `user_project` (`user_id`, `project_id`) VALUES
(1, 10),
```

```
(1, 11),
(1, 12),
(1, 14),
(1, 15),
(1, 16),
(1, 17),
(1, 18),
(1, 19),
(1, 20),
(1, 21),
(1, 22),
(1, 23),
(2, 10),
(2, 11),
(2, 12),
(2, 14),
(2, 22),
(3, 13),
(3, 14);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `comment`
--
ALTER TABLE `comment`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `project`
--
ALTER TABLE `project`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `time_element`
--
ALTER TABLE `time_element`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `email` (`email`);
```

```
--
-- Indexes for table `user_project`
--
ALTER TABLE `user_project`
  ADD PRIMARY KEY (`user_id`,`project_id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `comment`
--
ALTER TABLE `comment`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=51;

--
-- AUTO_INCREMENT for table `project`
--
ALTER TABLE `project`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=24;

--
-- AUTO_INCREMENT for table `time_element`
--
ALTER TABLE `time_element`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=27;

--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```