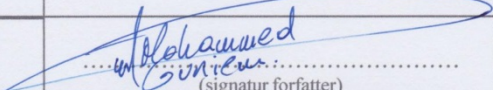




Universitetet  
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

## BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i data, ingeniørfag	Vår.....semesteret, 2018..  Åpen / Konfidensiell
Forfatter: Mohammed Guniem	 ..... (signatur forfatter)
Fagansvarlig: Erlend Tøssebro Veileder(e): Erlend Tøssebro	
Tittel på bacheloroppgaven: CityShare - web applikasjon for kartlegging, forskning og analyse av geografiske områder Engelsk tittel: CityShare - web application for mapping, research and analysis of geographical areas	
Studiepoeng: 20 Poeng	
Emneord: Web application - Geographic digital mapping - HTML - CSS - JavaScript - jQuery - Plugins - Ajax - Bootstrap - Python - MySQL - Google Maps APIs	Sidetall: ..... 113 sider CityShare_Flask + vedlegg/annet: City Data iOS CityDataAndroid CityShare_Demo Example_CSV_Figurs_fil.csv Stavanger, 15-05-2018..... dato/år

Forside for bacheloroppgaven  
Det teknisk-naturvitenskapelige fakultet

<b>Forord.....</b>	<b>4</b>
<b>Kapittel 1: Innledning.....</b>	<b>5</b>
1.1 Motivasjon.....	5
1.2 Bakgrunn .....	7
<b>Kapittel 2: Beskrivelse av prosjektet.....</b>	<b>7</b>
<b>Kapittel 3: Utviklingsverktøy, valg av arbeidsmetoder og plattform.....</b>	<b>10</b>
3.1 Python, Flask rammeverket og «render_template» motoren/metoden.....	11
3.2 HTML og CSS.....	13
3.3 JavaScript, JQuery og Ajax forespørsler .....	13
3.4 Bootstrap .....	14
3.5 Plugins .....	15
3.6 Google Maps APIs .....	17
3.7 MySQL.....	22
3.8 Plattform og dataprogrammer brukt i prosjektet. ....	23
<b>Kapittel 4: Database.....</b>	<b>25</b>
4.1 Relasjonsmodell .....	25
4.2 Normalisering.....	26
4.3 Indekser .....	27
4.4 Triggere .....	28
<b>Kapittel 5: Autoritet- og tilgang håndtering og Sikring av nettstedet.....</b>	<b>33</b>
5.1 Innlogging, utlogging og identifikasjon av klienter. ....	33
5.2 Autoritet- og tilgang håndtering .....	34
5.3 Kryptering og sikring av brukernavn og passord i databasen .....	36
<b>Kapittel 6: Feilhåndtering .....</b>	<b>38</b>
<b>Kapittel 7: Applikasjonens struktur og innhold.....</b>	<b>41</b>
7.1 Filsystemet i prosjektet .....	41
7.2 Server Program – «Bak-Ende».....	43
7.3 Grafisk brukergrensesnitt.....	48
7.4 hjemmesiden «home.html», registrering av nye brukere i «signup.html», og oppdatering av kontoopplysninger i «update_account.html».....	50
7.5 Opprettelse av nytt kart i siden «new_map.html» .....	51
7.6 Visning av alle kart som brukeren er medlem av i siden «view_user_maps.html».....	54
7.6.1 Søkning blant kart i siden «view_user_maps.html».....	55
7.7 Registrering av nye respondenter på kartet gjennom siden «open_map.html».....	56
7.7.1 Kartet fra «Google Maps API» .....	58

7.7.2 Kontrollpanel og hjelpeverktøy .....	61
7.8 Analysering av kart og data tatt fra respondenter i siden «analyze_map.html».....	64
7.8.1 Filtrering av data i kartet .....	67
7.8.2 Konstruksjon av en «CSV» fil som inneholder informasjon om figurene i kartet. ....	69
7.8.3 Kontrollpanel og hjelpeverktøy .....	72
7.9 Visning av svar på spørsmål fra respondenter i siden «respondents_answers.html».....	73
7.10 Tilbakemeldinger og samtale arena i siden «feedback.html» .....	74
7.11 Innføring av nye kategorier, spørsmål, administratorer og intervjuere i kartet etter opprettelse gjennom siden «update_map_details.html». ....	76
7.12 Feilmeldingssiden «error.html» .....	77
<b>Kapittel 8: Tilpasningsdyktig Skjerm design «Screen Responsive Design».....</b>	<b>78</b>
<b>Kapittel 9: Mobilapplikasjon kompatibel med operativsystemene, Android og iOS.....</b>	<b>82</b>
9.1 Om Web ramme «engelsek. Webframe» applikasjoner. ....	82
9.2 Fordeler og ulemper med en slik type applikasjon. ....	82
9.3 Refleksjon og avgjørelse angående en slik applikasjon.....	84
9.4 Implementering av web ramme applikasjon i operativsystemet «iOS» .....	84
9.5 Implementering av web ramme applikasjon i operativsystemet «Android».....	86
<b>Kapittel 10: Tilbakemeldinger på applikasjonen og Refleksjon.....</b>	<b>89</b>
10.1 Testing og Tilbakemeldinger på applikasjonen. ....	89
10.2 Refleksjon og konklusjon.....	91
<b>Kapittel 11: Vedlegg.....</b>	<b>92</b>
11.1 Framgangsmåte for publisering av nettsiden ved UNIX anlegget i Universitet i Stavanger. ....	92
11.2 Database.....	94
11.2.1 Skjema for opprettelse av databasen.....	94
11.2.2 De brukte MySQL Spørringene i applikasjonen.....	99
11.3 ArcGis Python Skript som konverterer til kategoriene i 3 separate filer, skrevet av veilederen Erlend Tøssebro .....	105
11.4 Tilbakemeldinger .....	108
11.4.1 Tilbakemelding fra Doktorgradskandidaten i Institutt for industriell økonomi, risikostyring og planlegging ved TN-fakultetet, ved Universitet i Stavanger, Ana Llopis Alvarez.....	108
11.4.2 Tilbakemelding fra Doktorgradskandidaten i Institutt for data- og elektroteknologi (IDE) ved TN-fakultetet, ved Universitet i Stavanger, Dario Garigliotti. ....	109
11.5 Spesifisert arbeidsmengde og timeliste .....	110
<b>Referanseliste .....</b>	<b>112</b>

## Forord

Denne applikasjonen er publisert under linken, [www.cityshare.ux.uis.no](http://www.cityshare.ux.uis.no). På denne nettsiden finnes det en videodemonstrasjon som skal kort og enkelt vise funksjonaliteten av applikasjonen. Videoen er også tilgjengelig med linken <https://www.youtube.com/embed/iHII4bKcK8>.

Denne rapporten tar leseren gradvis gjennom utviklingen av applikasjonen. Til å begynne med, forklares arbeidsmetodene og verktøyene som er brukt i dette prosjektet. For deretter å gi leseren konkret informasjon om databasen til applikasjonen, feil- og autoritetshåndtering og til slutt er det en gjennomgang av applikasjonens struktur og virkemåte.

Min bacheloroppgave har vært en god mulighet for meg til å utvikle mine kunnskaper og ferdigheter i mange fagfelt. Gjennom vårsemesteret har jeg prøvd å gjøre mitt beste for å gjøre ideen bak dette prosjektet til virkelighet. Vedlagt i slutten av denne rapporten en spesifisert liste over min arbeidsmengde i dette prosjektet.

Jeg vil herved takke alle som har vært med og bidratt til å gjøre ideen bak dette prosjektet til virkelighet. Jeg sitter stor pris på innsatsen.

## Kapittel 1: Innledning

### 1.1 Motivasjon

Denne applikasjonen skal gjøre det mulig for brukeren å gjennomføre spørsmålsundersøkelser og deretter kunne binde svar på disse spørsmålene til ulike posisjoner, sti og områder på et allerede valgt og avgrenset geografisk kart.

Geografiske data har gjennom tiden vært mye brukt i alle mulige felter til å danne et bilde om problemstillinger og observasjoner som oppstår på et avgrenset område.

Det er ofte stort behov til å knytte informasjoner og nylige observasjoner om ulike temaer til den geografiske posisjonen hvor de ulike hendelsene har bestått eller blitt observert.

Slike temaer kan for eksempel handle om blant annet forurensing, migrasjon, globaloppvarming, sykdomsspredning og ikke minst markedsføring.

Analytikere forsøker å tegne slike kart for å kunne forstå hvordan sine data og observasjoner fordeler seg på tvers av det interessante geografiske området. En slik geografisk tegning av objekter på et kart kan gi en bedre forståelse om observasjonen og tar analytikeren nærmere løsningen til mange problemstillinger og/eller skaffer oss bedre kunnskap om ulike temaer og problemstillinger.

Bildet nedenfor er et eksempel på en geografisk spørsmålsundersøkelse, den viser et helhetlig kart over Hillevåg området i Stavangerbyen i Norge med mange kommentarer om flere aktuelle geografiske former som kan være et bestemt sted, et område eller en sti i kartet. Til høyre i bildet nedenfor ser vi noen spørsmål som respondenten svarte på under intervjuet som f.eks. alder, i tillegg finner vi noen kommentarer og en kategoriforklaring.





## 1.2 Bakgrunn

Ideen bak dette prosjektet er foreslått av Førsteamanuensis ved Instituttet for industriell økonomi, risikostyring og planlegging ved Universitet i Stavanger, Daniela Müller-Eie.

Prosjektoppgaven er dermed formulert gjennom et tett samarbeid mellom Daniela Müller-Eie og doktorgradsstudenten Ana Llopis Alvarez i industriell økonomi, risikostyring og planlegging ved Universitet i Stavanger, og mellom utvikleren av dette prosjektet Bachelorgradsstudenten Mohammed Z. Gunim. Med veiledning av førsteamanuensis ved Instituttet for data- og elektroteknologi Erlend Tøssebro.

Før dette prosjektet brukte intervjuere ved Instituttet for industriell økonomi, risikostyring og planlegging ved Universitet i Stavanger å gjennomføre undersøkelser på papir, og nå foreligger det et ønske om å digitalisere den prosessen for å gjøre dette mer effektivt og produktivt.

## Kapittel 2: Beskrivelse av prosjektet

Instituttet for industriell økonomi, risikostyring og planlegging ved universitet i Stavanger ønsket seg en webapplikasjon(nettside), dette er fordi de ønsker at applikasjonen skal være tilgjengelig for alle mulige operativsystemer og at brukeren ikke skal være nødt til å bruke en spesifikk type digitalenhet som f.eks. kun nettbrett men også gjennom PC-er og andre enheter av alle mulige typer og operativsystemer.

Følgende funksjonalitet er forventet fra applikasjonen:

- En bruker av applikasjonen skal være registrert med brukernavn, passord, fornavn, etternavn, epostadresse og et telefonnummer. Disse opplysningene unntatt brukernavnet skal kunne endres av brukeren ved et senere tidspunkt.

- En bruker kan deretter opprette så mange kart han/hun ønsker, og vil ved dette tidspunktet bli den ene permanente eieren av dette kartet.
- Et kart skal ha en tittel, beskrivelse, opprettelsesdato, gyldighetsdato, geografiske grenser og forstørrelsesgrad(zoom/høyde over bakken).
- Med kartet bør oppretteren/eieren definere en eller flere av følgende geografiske kategorier(Punkt, Område, eller vei/sti). en slik kategori skal ha et navn og type, og skal lagre et unikt ikonbilde for hvert punkt og en unik farge for hver av område- og stikategoriene.
- Ved opprettelse bør også eieren av kartet definere et eller flere spørsmål som skal stilles til respondenter under undersøkelsene.
- Eieren av kartet er den eneste som kan dele ut tilgang til sitt kart til andre brukere av applikasjonen. Tabellen under viser hvordan en bruker får en rolle av et kart og hva muligheter gir denne rollen til brukeren som bærer den.

Rolle	Ansettelse	Muligheter
Eier	Når en person oppretter et kart vil den personen få eierrollen av dette kartet.	<p>En eier kan legger til kartet følgende ved enten opprettelse eller senere ved en oppdatering.</p> <ul style="list-style-type: none"> <li>➔ Nye administratorer og intervjuere</li> <li>➔ Nye kategorier</li> <li>➔ Nye spørsmål</li> </ul> <p>I tillegg til disse mulighetene vil eieren alltid være en administrator av sitt kart, og det vil si at eieren har alle muligheter som en administrator har. (se neste rad i denne tabellen)</p> <p>Med andre ord så er en Eier en</p>



		administrator, men en administrator er ikke nødvendigvis en eier av et kart.
Administrator	Eieren av kartet tildeler Administratorrollen til hvilken som helst av de registrerte personene.	Har tilgang til å se all data samlet opp av alle personer som har tilgang til kartet(administratorer og intervjuere)  Har mulighet til å se, slette og redigere innholdet av data(figurer på kartet) tatt opp av alle administratorer inkludert seg selv og intervjuere.  Har selv muligheten til å intervju nye respondenter i kart undersøkelsen.
Intervjuer	Eieren av kartet tildeler Intervjuerrollen til hvilken som helst av de registrerte personene.	Har kun tilgang til å se sine egne data.  Har kun muligheten til å se, slette og redigere innholdet av data(figurer på kartet) den han/hun har ført inn i kartet.  Har muligheten til å intervju nye respondenter i kart undersøkelsen.

- Eieren, administratorer og intervjuere kan intervju nye respondenter ved å stille de allerede definerte spørsmål av eieren og fylle inn et svar og eventuelt tegne inn nye geografiske figurer(Punkter, Stier eller områder) på kartet.

- Disse figurene som tegnes på kartet skal ha en tittel, beskrivelse og vurdering i følgende skala: (1) veldig dårlig, (2) dårlig, (3) ikke verst, (4) bra og (5) veldig bra.

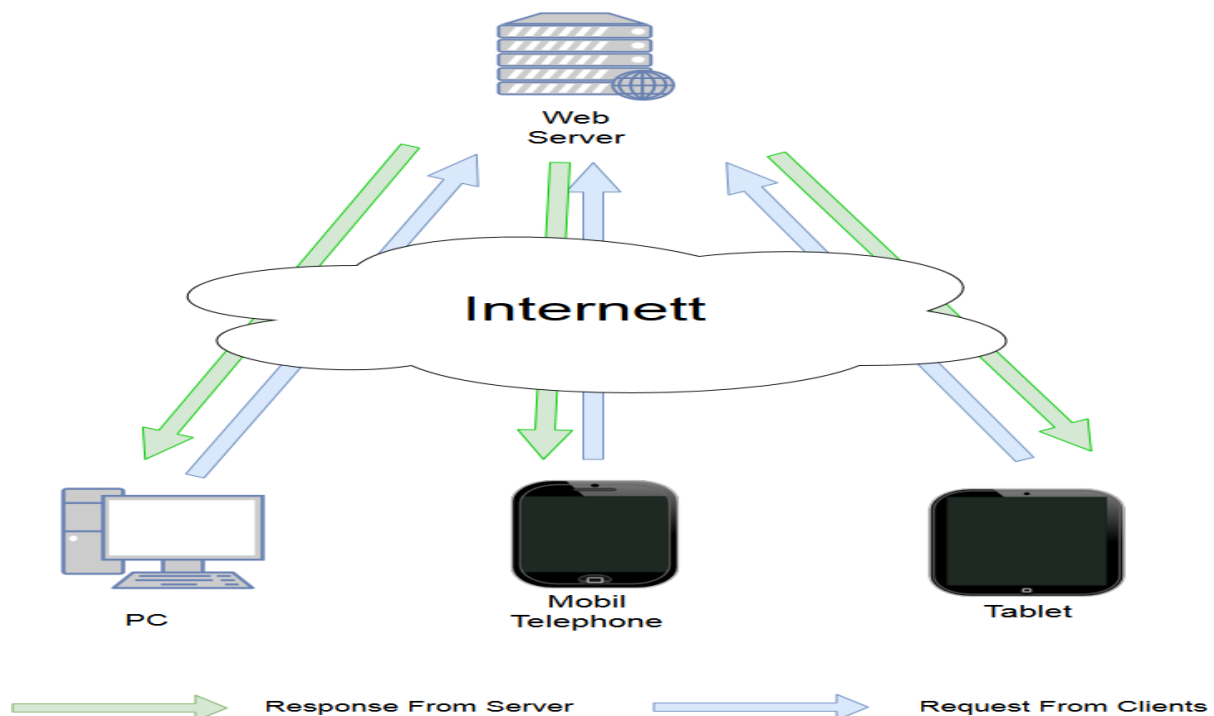
- Både svar fra respondenter og geografiske figurer skal lagres i applikasjonens database, for deretter å bli brukt til analyse av de involverte i arbeidet med undersøkelsen.

- All brukere av et kart har tilgang til en tilbakemeldingsarena der de kan legge til sine kommentarer og eventuelle spørsmål om arbeidet med dette kartet.
- En fil av typen «CSV» som inneholder informasjonen om figurene som er tegnet på kartet samt tilsvarende svar på spørsmål fra respondenter, og andre relevante detaljer skal kunne nedlastes av brukeren under analysing av kartet.

### Kapittel 3: Utviklingsverktøy, valg av arbeidsmetoder og plattform

Denne applikasjonen er av typen webapplikasjon, og som alle webapplikasjoner vil den bestå av en Server-side «Bak-Ende» og en Klient/bruker-side «Front-Ende».

Nettverksprotokollen HTTP blir brukt her; ved å bruke denne protokollen kan en klient som har en aktiv forbindelse til Internett nettet sende forespørsler til server-siden som lytter på innkommende forespørsler fra de mange klienter over Internett. Figuren under viser en forenklet beskrivelse av kommunikasjonen mellom klienter og server.



Figur 2, en forenklet modell av webapplikasjoner.

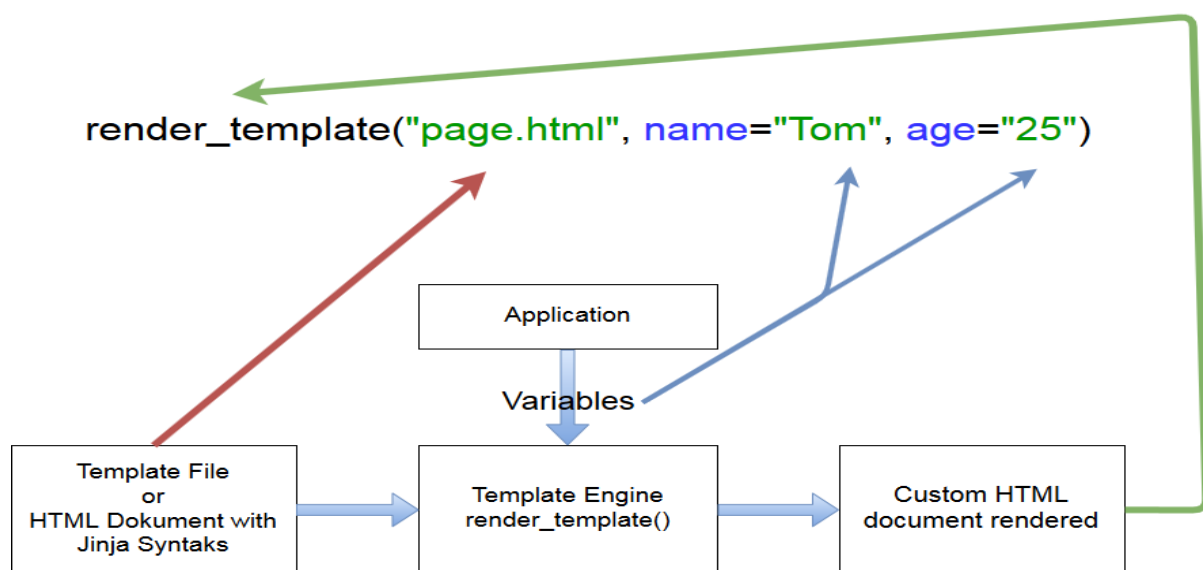
### 3.1 Python, Flask rammeverket og «render\_template» motoren/metoden.

Python og PHP er de to mest populære programmeringsspråkene i Bak-ende utviklingen av webapplikasjoner. Det eksisterer i dag mer programmeringsrammeverk i PHP enn Python men Python koden er mer leselig og lettere å forstå. Python har også bedre feilmeldinger, og kan bli brukt til andre programmeringsoppgaver enn bare webutvikling, mens PHP er dedikert til webutvikling. Basert på den sammenligningen og de kunnskapene jeg har fått gjennom min utdanning, har jeg valgt å bruke Python.

Python tilbyr hovedsakelig to hoved rammeverk, Flask og Django. Et rammeverk i programmeringssammenheng utgjør en rekke metoder og hjelpemidler som gir programmereren muligheten til å slippe å bygge alt fra bunnen av.

Flask rammeverket er en forenklet versjon og anses ofte som et forskningsbarn av Django rammeverket. Jeg har tatt avgjørelsen om å bruke Flask fordi jeg er mest kjent med dette rammeverket men også fordi filstrukturen i Django rammeverket kan virke litt komplisert, Flask i motsetning til Django bruker kun to filer; den ene heter «static» og inneholder alle dokumenter som ikke endrer seg dynamisk, mens på den andre filen som har navnet «templates» finner vi HTML-sidene som forandrer seg dynamisk for å tilpasse hver forespørsel for seg.

I denne applikasjon vil hver klient ofte motta en ulik HTML og JavaScript innhold enn andre klienter, alt etter hva klienten har rett og tilgang til. Derfor bruker Flask en såkalt «render\_template» motor som med likhet til den fysiske motoren tar inn et HTML-dokument «template» sammen med andre data verdier og setter disse verdiene inni dokumentet for å produsere et HTML-dokument med innhold tilpasset til den forespørselen fra klienten. For å sette disse data verdiene i HTML-dokumentet bruker Flask «Jinja» Syntaks som plasseres i den enkelte dokumentet/«template», «render\_template» motoren vil deretter gjenkjenne en slik syntaks fra vanlig HTML syntaks og behandler input data på den måten syntaksen er satt opp på. Figuren under viser hvordan «render\_template» motoren opererer i Flask.



**Figur 3, Virkemåten til metoden "render\_template".**

Et eksempel på hvordan «template» HTML-dokumentet vil se ut før forrige kall på «render\_template» funksjonen er slik. Etter kallet erstattes «{{ name }}» med «Tom» og «{{ age }}» med «25».

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Webpage</title>
  </head>
  <body>
    <div>My name is {{ name }} and I am {{ age }} years old</div>
  </body>
</html>

```

Mer informasjon om Jinja Syntaks er tilgjengelig på følgende linker:

<http://jinja.pocoo.org/docs/2.10/>

<http://flask.pocoo.org/docs/0.12/templating/>

## 3.2 HTML og CSS

HTML-dokumenter benyttes til formattering av nettsider og viser ønsket innhold av elementer som for eksempel tekstfelter, bokser og paneler.

For å gi nettsider ønsket design som kombinasjon av farger, plassering på skjermen og andre relevante alternativer brukes det CSS syntaks. CSS syntaks kan plasseres enten i selve hodet av html dokumentene mellom «<style></style>» elementet eller den kan importeres fra en ekstern fil med utvidelsen «CSS» ved å bruke koden i følgende eksempel i hodet av HTML-dokumentet. Merk at Jinja Syntaks og metoden «url\_for(...)» er her brukt til å formatere URL-stien til ønsket fil

```
<link rel="stylesheet" type="text/css" href="{ {  
url_for('static',filename='Stylesheets/filnavn.css') } }">
```

## 3.3 JavaScript, JQuery og Ajax forespørsler

JavaScript er et høynivå-programmeringsspråk som er en av grunnsteinene i moderne webutvikling. Alle tilgjengelige nettlesere kan kompilere JavaScript-programmer uten noe behov for utvidelser. I motsetning til språket Java som de fleste programmerere har vært borti er JavaScript et språk som er kjent for dynamiske datatyper som implisitt konverteres under kjøring, dette er en fordel for programmereren da programmering med JavaScript vil bli enklere og mer feiltolerant.

JavaScript koding kan ofte bli lang og komplisert til å forstå, for dette ble det utviklet et bibliotek som har navnet «JQuery» og som inneholder en rekke funksjoner og gjør JavaScript koden kortere og lett leselig. Det er derfor «JQuery» funksjoner er bedre foretrukket til bruk i det meste av JavaScript koden i denne applikasjonen.

Denne applikasjonen krever at nettsidene som viser kart skal kunne vært interaktive, dvs. at brukeren skal kunne kommunisere med serveren uten å ha noe behov for å laste hele siden på nytt ved å sende en ny «GET» forespørsel til serveren. For å løse dette tilbyr



JavaScript såkalte «AJAX» forespørsler som kan sende deler av informasjonene i siden til serveren og motta svar fra serveren, for deretter å oppdatere siden uten å påvirke andre data i den.

«AJAX» forespørsler kan sendes av typen «GET» eller «POST» men i denne applikasjonen er det kun brukt forespørsler av typen «POST». Det er mulig å implementere en «AJAX» forespørsel med enten ren JavaScript Syntaks eller ved bruk av «JQuery» biblioteket, JQuery tilbyr en kortere og konkret kode til denne forespørselen og derfor er dette brukt i denne applikasjonen, nedenfor har vi et eksempel til en slik forespørsel. I eksemplet nedenfor sender vi 2 parametere A og B med sine verdier til serveren og når kommunikasjonen er vellykket og en respons blir mottatt så skal «status» verdien bli lik 200, mens «data» representerer responsen fra serveren.

```
$.post("/server_url",  
{  
  parameter_A: verdi,  
  parameter_B: verdi,  
},  
function(data, status){  
  // status → forteller noe om en respons ble mottatt eller om en feil har skjedd osv.  
  // data er responsen som ble returnert fra serveren.  
  // her kan vi også gjøre noe med responsen vi mottar fra serveren  
});
```

### 3.4 Bootstrap

Bootstrap er ett front-ende rammeverk som er et åpenkilde bibliotek og er hovedsakelig skrevet i 3 språk; HTML, CSS, og JavaScript.

Basiske HTML elementer kan ofte virke gammaldags og ikke attraktive nok, derfor brukes det i denne applikasjonen «Bootstrap» for å gi HTML-elementene bedre design og funksjonalitet. I tillegg inneholder «Bootstrap» flere filer av typen «CSS» som har allerede definerte klasser som hjelper til å gi nettsiden bedre design.

«Bootstrap» har fordelen at den sparer programmereren tid fra å skrive selv «CSS» og «JavaScript» koden. Den også tilbyr flere enkle klasser som gjør nettsiden tilpasningsdyktig til størrelsen på skjermen, noe som blir også brukt her i kombinasjon med «JavaScript».

Ulempene ved «Bootstrap» er at klassenavn kan ta over egne definerte klasser og ødelegge funksjonen av egen «CSS» og «JavaScript» kodene. Så man er nødt til å være oppmerksom på hva slags klassenavn eksisterer i «Bootstrap» biblioteket for så å bruke andre navn enn de. Det er også hevdet at «Bootstrap» er mindre læringsrik fordi brukeren slipper å skrive selv denne koden, sammen med at «Bootstrap» gjør kanskje ofte at nettsiden ligner på alle andre nettsider som bruker samme bibliotek, noe som skape problemer hvis nettsiden representerer en forretning med andre konkurrenter, men denne applikasjonen har ikke som mål å tjene penger og derfor utgjør ikke dette en stor ulempe.

### 3.5 Plugins

I likhet med alle andre «CSS» og «JavaScript» bibliotek, er det også tilgjengelig flere allerede skrevet koder av programmerere rundt verden og som løser et problem eller forbedre designet. Noe som ofte blir kalt for «Plugins»

I denne applikasjonen er det brukt følgende «Plugins»:

A- Datovelger: for å gi brukeren en utvidet kalender når han/hun skal velge dato. Den er en del av «jQuery» biblioteket og brukes på følgende måte i JavaScript koden for HTML-filene «new\_map.html» og «update\_map\_details.html»:

```
// Datepicker - jQuery
var dateToday = new Date();
var dates = $("#date").datepicker({
    defaultDate: "+3m",
    changeMonth: true,
    numberOfMonths: 1,
    minDate: dateToday,
    onSelect: function(selectedDate) { ... };
```

B- «Font Awesome» ikoner og «bootstrap glyphicon»: disse utgjør en gruppe varierte ikoner som kan brukes i applikasjonen. «Bootstrap glyphicon» er inkludert i importeringen av «Bootstrap» og trengs ikke å bli importert mens «Font Awesome» kan bli importert slik som vist under.

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

C- Fargevelger: Brukes for å gi brukeren et bedre HTML-element til å velge ønsket farge. Denne «plugin» er lastet ned fra følgende linke:

<https://bgrins.github.io/spectrum/#methods-get>

og lagret under filen med navnet «Color\_Picker\_Plugin» for deretter å bli importert i hodet av html dokumentet og brukes ved hjelp av JavaScript på følgende måte:

```
$("#ID_name_of_html_select_list").spectrum({  
    color: "#3355cc"  
});
```

D- Punktikoner: Det foreligger et ønske om å ha flere små ikoner som skal representere geografiske punkter på kartet. Designing av slike ikoner krever mye tid, det er heldigvis noen som tilbyr gratis ikoner til nedlastning og bruk i nettsiden, ikonene er lastet fra følgende nettside <https://mapicons.mapsmarker.com/> og utgjør til sammen omtrent 697 ikoner til disposisjon for brukeren, alle ikoner er lagret under denne mappen «static/Icons» og Python skriptet «read\_icons.py» genererer «option» html elementene slik at de blir kopiert til «select» listen i «new\_map.html» og «update\_map\_details.html».

E- «Chosen Multi Select»: Det vanlige HTML-elementet «Select» gir brukeren mulighet til å velge flere verdier samtidig, men den er dessverre ikke så brukervennlig.

Derfor er «chosen Multi Select» brukt i html dokumentet «analyze\_map.html», «Select» listen skal inkludere alle navn fra administratorer og brukere for deretter å filtrere figurer i kartet basert på hvem som har lagt inn disse figurene.

Den er også brukt i «new\_map.html» for å gi brukeren anledningen til å se ikonbildene når «Select» menyen er åpen, noe vanlig det vanlige HTML-elementet «Select» ikke tilbyr.

Følgende link inkluderer informasjon om «Chosen Multi Select»

<https://harvesthq.github.io/chosen/>.

For «Select» menyen i «analyze\_map.html» blir dette satt opp i tilsvarende JavaScript fil på følgende måte.

```
$('#filter_interviewers').chosen({  
  width:'100%',  
  placeholder_text_multiple: "Alle Interviewere"  
});
```

Mens for «new\_map.html» er dette implementert rett etter «Select» listen som vist under.

```
<script>$('.my-select').chosen({ width:'100%' });</script>
```

### 3.6 Google Maps APIs

Det finnes i dag flere API tjenester som tilbyr digitale kart, men «Google Maps APIs» står per i dag som uslåelig av andre digitale karttjenester.

At Google Maps er mest brukt i verden er ikke overraskende ettersom Google gir valget mellom flere API tjenester som er tilpasset behovet en bruker har og er lett å implementere, bruke og kartlegge, Google gir også en svært detaljert API dokumentasjon med gode eksempler. I tillegg til dette driver Google med kontinuerlig forbedring og

utvikling av API egenskaper og tjenester som garanterer at Google Maps API vil bli brukt enda mer i framtiden.

Google Maps har allikevel noen ulemper som f.eks. at brukeren er nødt til å forstørre kartet ganske mye for å kunne se skisser av lokaler og eiendommer på kartet. Noe andre API er kanskje bedre på. Men dette utgjør faktisk ikke et stort problem i vår applikasjon.

En annen ulempe er at bruk av Google Maps APIs er ikke helt uten kostnad, men den tilbyr en grense på 25000 forespørsler i døgnet før nettsiden må begynne å betale 0,5 amerikansk dollar for hver 1000 forespørsler, noe som anses å være svært billig i forhold til kvaliteten på karttjenestene.

For å kunne bruke Google Maps API er programmereren nødt til å registrere seg og skaffe seg en nøkkel som skal implementeres i applikasjonen. For å generere en slik nøkkel kan følgende linke gi en rask framgangsmåte

<https://developers.google.com/maps/documentation/javascript/get-api-key>.

Denne applikasjonen bruker 4 tjenester fra Google Maps API, disse er «Google Maps JavaScript», «Google Maps Places» som brukes for adressegjenfinning, «Google Maps Static» som tilbyr statiske kartbilder av et bestemt geografisk område, og «Google Maps Street» som gir muligheten til å se på bilder fra en stor del av verdens gater og steder i 360 grader utsikt, noe som sparer mye plass i databasen, og utviklingstid fordi vi nå slipper å legge til en funksjon som tar opp bilder og lagrer disse i databasen.

Implementeringen av «Google Maps JavaScript» skjer som vist nedenfor i html dokumentet, som vi ser trenger URL 3 parametere, «key» nøkkel du genererte i forrige steg, «libraries» som aktivere bruk av «Google Maps Places» API og «callback» som skal ha navnet på funksjonen som setter opp og starter kartet. HTML-elementet «div» med ID-en «Map» skal her inneholde selve kartet.

```
<div id="Map" class="part"></div>
```

```
<script async defer
```



```
src="https://maps.googleapis.com/maps/api/js?key=DittNøkkelKommerHer&libraries=places&callback=initMap">
</script>
```

Nedenfor er det vist hvordan på en enkel måte og ved hjelp av JavaScript kan vi sette opp vårt kart ved opplasting av nettsiden, vi ser at «Google Maps Street» API er aktivert og posisjonert for kartet ved bruk av parameteren «**streetViewControlOptions**»

```
function initMap() {
    map = new google.maps.Map(document.getElementById('Map'), {
        zoom: 11,
        center: {lat: 58.969975, lng: 5.733107},
        mapTypeControlOptions: {
            position: google.maps.ControlPosition.RIGHT_BOTTOM
        },
        streetViewControlOptions: {
            position: google.maps.ControlPosition.RIGHT_CENTER
        },
        zoomControlOptions: {
            position: google.maps.ControlPosition.RIGHT_CENTER
        },
        fullscreenControl: false
    });
}
```

For å aktivere adressegjefinnding brukes det følgende kode i metoden «initMap()» vist ovenfor:

```
var card = document.getElementById('pac-card');
var input = document.getElementById('pac-input');
map.controls[google.maps.ControlPosition.TOP_CENTER].push(card);
```

```

var autocomplete = new google.maps.places.Autocomplete(input);

// Bind the map's bounds (viewport) property to the autocomplete object,
// so that the autocomplete requests use the current map bounds for the
// bounds option in the request.
autocomplete.bindTo('bounds', map);

autocomplete.addListener('place_changed', function() {

    var place = autocomplete.getPlace();
    if (!place.geometry) {
        window.alert("No details available for input: '" + place.name + "'");
        return;
    }

    // If the place has a geometry, then present it on a map.
    if (place.geometry.viewport) {
        map.fitBounds(place.geometry.viewport);
    } else {
        map.setCenter(place.geometry.location);
        map.setZoom(17); // Why 17? Because it looks good.
    }
});

```

I HTML-siden plasseres følgende element som skal være søkefeltet for adresser i kartet.

```

<div class="pac-card" id="pac-card">
  <div id="pac-container" class="input-group">
    <input id="pac-input" class="form-control" type="text">
  </div>
</div>

```

For å gi brukeren muligheten til å fastslå sin posisjon på kartet brukes koden nedenfor i metoden `initMap()`. Nettleseren vil stille brukeren spørsmålet om han/hun ønsker å vise sin posisjon etter opplasting av nettsiden og hvis valget er for dette så skal posisjonen finnes og en blå markør vil da vises på kartet sammen med en radius sirkel.

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    var pos = {
      lat: position.coords.latitude,
      lng: position.coords.longitude
    };
    map.setCenter(pos);
    map.setZoom(11);
    var location = new google.maps.Marker({
      clickable: false,
      icon: new google.maps.MarkerImage
        ('//maps.gstatic.com/mapfiles/mobile/mobileimgs2.png',
        new google.maps.Size(22,22),
        new google.maps.Point(0,18),
        new google.maps.Point(11,11)),
      shadow: null,
      zIndex: 999,
      map: map // your google.maps.Map object
    });
    location.setPosition(pos);
    // Add circle overlay and bind to marker
    var circle = new google.maps.Circle({
      map: map,
      radius: 10, // 10 miles in metres
      fillColor: '#00bcff',
      strokeColor: '#00bcff'
    });
    circle.bindTo('center', location, 'position');
```

```

}, function() {
    // User does not wish to share his/her position
});

} else {
    // Browser doesn't support Geolocation
    alert("Din browser støtter ikke geografisk gjenfinning");
}

```

«Google Maps static» blir her brukt i html dokumentet «view\_user\_maps.html» og siden «update\_map\_details.html» for å vise et overblikk statisk bilde av det kartet som er opprettet, koden nedenfor viser hvordan man importere et statisk kart fra Google ved bruk av HTML-elementet «img», merk at «Jinja» Syntaks er her brukt for å angi senter og høyde eller zoom over bakkenivå, og selvfølgelig sammen med nøkkelen til Google Maps APIs

```



```

### 3.7 MySQL

Denne applikasjonen er svært avhengige av relasjoner mellom ulike objekter som kart, brukere, kategorier, spørsmål og former/figurer. Derfor vil valget av en effektiv database være kritisk for denne applikasjonen.

MySQL er et velkjent databasesystem som kjører på mange operativsystemer, blant disse står Unix operativsystemet som anlegget ved Universitet i Stavanger kjører.

MySQL har også flere grensesnitt til programmeringsspråk, noe som gjør kommunikasjon mellom databasen og mange applikasjoner som bruker ulike språk lett mulig. Dette kan hjelpe dersom Python som serverspråk skal senere i framtiden endres til andre språk.

MySQL tilbyr en rekke funksjoner og datatyper som er egnet til bruk i denne applikasjonen, et eksempel er krypteringsfunksjonene «aes\_encrypt» og «aes\_decrypt» sammen med datatypene «Point», «Linestring» og «Polygon» og supplerende funksjoner som «INTERSECT» og «CONTAINS».

Det er også verdt å nevne at optimalisering av spørringer skjer på en automatisk måte i MySQL, noe som ofte gir raskere tilgang til data og dermed bedre kjøretid.

MySQL har vært tidligere kritisert for mangel av viktige funksjoner som andre databaser har hatt lenge, og mangel på transaksjoner og ACID-egenskaper, dessuten blir håndtering av NULL-verdier, standardverdier og verdier som ikke passer med kolonnetypene også kritisert av databaseeksperter.

Mye av denne kritikken har forsvunnet ettersom disse manglene er dekket og tilfredsstilt fra nye versjoner av MySQL inkludert MySQL versjon 5.7 som UNIX anlegget på UIS kjører per i dag.

### **3.8 Plattform og dataprogrammer brukt i prosjektet.**

For å utvikle denne applikasjonen er det brukt følgende plattformapplikasjoner.

- PyCharm med følgende lisens

PyCharm 2017.3.3 (Professional Edition)

Build #PY-173.4301.16, built on January 11, 2018

Licensed to Mohammed Guniem

Subscription is active until February 23, 2019

JRE: 1.8.0\_152-release-1024-b11 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 7 6.1



#### - MySQL Workbench 6.3

Denne plattformapplikasjonen er en åpenkilde og gratis applikasjon som ikke behøver noe lisens. Den kan nedlastes gjennom linken <https://dev.mysql.com/downloads/workbench/>.

#### - XCode for mac iOS

Denne plattformapplikasjonen er en åpenkilde og gratis applikasjon som ikke behøver noe lisens. Følgende link inneholder informasjon om bruk av «XCode»

<https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/>.

#### - Android Studio

Denne plattformapplikasjonen er en åpen kilde og gratis applikasjon som ikke behøver noe lisens. Følgende link inneholder informasjon om bruk av «Android Studio»

<https://developer.android.com/training/basics/firstapp/>.

## Kapittel 4: Database

### 4.1 Relasjonsmodell

# CityShare Database Relational Model



Figur 4, Entitetsrelasjonsmodell av applikasjonens database.

Relasjonsmodellen ovenfor representerer funksjonaliteten på denne applikasjonen og hvordan strukturen til databasen er bygget opp. Hver unik farge i modellen representerer en MySQL tabell, ovale former representerer kolonner i denne tabellen, mens diamanten viser relasjonen mellom to tabeller, firkanter inneholder navnet på MySQL tabellen. (N) står for mange og (1) står for en. Ovale former med strek under representerer en primærnøkkelkolonne i denne tabellen. Og bokstaven (o) viser at en person kan ha ulike roller i forskjellige kart.

Full oversikt over MySQL skjemaet fra oppsett av databasen kan finnes med vedlegg i kapittel 11.2.1

Gjennom testing som ble utført når applikasjonen var ferdig utviklet, har alle spørringene og MySQL setningene blitt tatt opp og dokumentert i tekstfilen med følgende sti «CityShare\_Flask/used\_queries.txt». alle disse spørringene er gjort tilgjengelige som vedlegg i kapittel 11.2.2.

Basert på «SELECT», «UPDATE» og «DELETE» setningene av disse spørringene er det gjort en vurdering av normaliseringsformer, og indekser.

## 4.2 Normalisering

Normalisering i sammenheng med relasjonsdatabaser beskriver måten dataattributtene er organisert mellom de forskjellige tabellene.

Høy normalform gir enklere innsetting og administrasjon siden samme data er lagret bare ett sted, den gir også flere tabeller og dermed større behov for å benytte «JOIN» betegnelse i spørringer, «JOIN» betegnelse har ofte høy kjøretid for spørringene fordi tabeller skal kombineres sammen. Men en høyere normalform sparer på mengden av data som lagres i databasen.

En lavere normalform vil derimot lagre mer data, fordi det kan kreves flere rader med attributtverdier som er like. Men en lav normal form har ofte bedre kjøretid for spørringene, fordi vi ikke behøver å slå sammen tabeller med «JOIN» betegnelse. Dersom målet er å ha så lav normalform som mulig så burde alt vært i en og samme tabell.

Målet her er å minimere mengden av data som lagres i databasen og samtidig minimere behovet for bruk av «JOIN» betegnelse i spørringer.

Som vi ser utfra MySQL skjemaet i 11.2.1, er de fleste tabeller i databasen normalisert til tredje normalform. Med noen få unntak som f.eks. typen til de geografiske figurkategoriene i tabellen «Maps\_Categories»; her velges det heller å inkludere typen som en «VARCHAR» attributt, dette er fordi alle former uansett type leses samtidig fra databasen. Men for å beskytte dette attributtet fra å ha andre verdier enn (Point, Area, Road) kan vi enten opprette en trigger eller sjekke typen statisk ved skrijving til databasen fra serveren. Det siste alternativet er valgt her. Koden under viser hvordan dette gjøres i serverprogrammet:

```
type = cat[1];  
if type == 'Point' or type == 'Area' or type == 'Road':  
    # Skriver til databasen
```

Antall ganger «JOIN» betegnelser som er brukt i «Select, Delete og Update» spørringer i applikasjonen er relativt liten i forhold til den totale antall spørringer.  $((7/27)*100 \approx 26\%)$ . Disse spørringene er tilgjengelige med vedlegg i kapittel 11.2.2.

Det er også viktig å ta i betraktning hvor ofte spørringene blir rettet til databasen, noe som er vanskelig å anslå før applikasjonen er publisert for fult og er brukt på en stund, derfor er frekvenselementet av spørringer ikke tatt med i denne vurderingen, men kan tas med i fremtiden når applikasjonen er blitt brukt i en periode.

## 4.3 Indekser

MySQL tabeller kan til å begynne med søkes uten noe problem helt til tabeller begynner å vokse til mer enn noen hundre rader, på dette tidspunktet vil tilgang til databasen begynner å bli tregere etterhvert nye rader legges til tabellen. Årsaken her er at MySQL databasen skal gjennomføre et søk ved å sjekke hver rad i tabellen for hver spørring.

Heldigvis kan vi i MySQL databaser oppnå en raskere spørring ved å sette opp en indeks på et eller flere attributter. Indekser kan bli lagt til ved opprettelsen av en ny tabell eller når som helst senere i framtiden.

Å velge hvordan en indeks skal plasseres i tabellen er ikke lett, da dette innebærer at vi kjenner til de mest brukte spørringene i databasen, og basert på disse spørringene kan vi bestemme hvilken kolonne i tabellen som krever en indeks.

Alle spørringer som har en «WHERE» betegnelse er dermed interessante å se nøye på. MySQL oppretter automatisk en indeks på både primærnøkkelkolonner og fremmednøkkelkolonner(dvs. kolonne som referer til en annen tabell).

Basert på slike spørringer tatt opp etter testing i tekstfilen «used\_queries.txt», ser vi at de fleste kolonner som brukes i «WHERE» betegnelsen har allerede en automatisk opprettet indeks på primær- og fremmednøkler, unntatt to kolonner i tabellen «Persons» som trenger 2 indekser; den ene er på «login\_pass» kolonnen og den andre er på «e\_post» kolonnen.

Dersom nye spørringer vil bli opprettet senere i framtiden, er det viktig å implementere nødvendige indekser for å gjøre søking raskest mulig.

#### 4.4 Triggere

En trigger i databasen utgjør en sekvens av kode som skal automatisk bli kjørt dersom en spesifikk definert hendelse skal oppstå på en spesifikk tabell i databasen.

Triggere er mest brukt for å opprettholde integriteten av informasjon og relasjoner i databasen, applikasjonens regler skal være kilden til hva en trigger bør gjøre og når. Beskrivelsen i kapittel 2 forklarer reglene til denne applikasjonen.

Triggere gir også fordelen til å skrive mindre kode i programmet, men dersom triggere ikke er godt dokumentert vil disse forårsake forvirring for databaseadministratoren, og programmereren som kommuniserer med databasen.

Triggere kan også i noen tilfeller utløse hverandre og forårsake et vranglås «dead lock» i databasen, derfor er det svært viktig at triggere skal studeres nøye før implementering og bruk.



I henhold til reglene til applikasjonen og håndtering av data i noen av tabellene, trengs det følgende trigger.

A- Hvis en Intervjuer blir senere lagt til som en administrator av et kart, skal den personen slettes som intervjuer av dette kartet, da administratorrollen gir personen alt en intervjuer har av muligheter i dette kartet. Dette er for å spare lagringsplass i databasen og unngå forvirring på hvem som er administrator og hvem som er intervjuer. Koden for en slik trigger er opprettet på denne måten:

delimiter |

```
CREATE TRIGGER Maps_Administrator_Interviewer_check_a AFTER INSERT ON  
Maps_Administrators
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DELETE
```

```
    FROM Maps_Interviewers
```

```
    WHERE username = NEW.username AND map_id = NEW.map_id;
```

```
END;
```

|

delimiter;

B- På samme måte som i trigger A ovenfor, kan en eier under oppdatering av tilgangsrettigheter til kartet i et senere tidspunkt ha et ønske om å ta administratorrollen fra et medlem av kartet og gi det medlemmet heller intervjuerrollen. Følgende trigger er opprettet i databasen for å fjerne personen fra tabellen «Maps\_Administrators» dersom brukeren blir lagt til i tabellen «Maps\_Interviewers». Her må det sjekkes om brukeren

ikke er eieren av kartet før den slettes fra administrator tabellen «Maps\_Administrators», dette er fordi en eier alltid vil være en administrator av sitt kart:

delimiter |

```
CREATE TRIGGER Maps_Administrator_Interviewer_check_b AFTER INSERT ON  
Maps_Interviewers
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF cast(aes_decrypt(NEW.username, 'enckey') AS CHAR(20)) NOT IN (
```

```
        select cast(aes_decrypt(M.map_creator, 'enckey') AS CHAR(20))
```

```
        From Maps M
```

```
        where (M.map_id = NEW.map_id)
```

```
    ) THEN
```

```
        DELETE
```

```
        FROM Maps_Administrators
```

```
        WHERE username = NEW.username AND map_id = NEW.map_id;
```

```
    END IF;
```

```
END;
```

|

delimiter ;

C- Etter opprettelse av et kart, skal eieren/opprettetoren automatisk bli lagt til som en administrator av dette kartet. Koden for en slik trigger er opprettet på denne måten:

delimiter |

```
CREATE TRIGGER Add_Map_Creater_As_Administrator_check AFTER INSERT ON  
Maps
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO Maps_Administrators(username, map_id)
```

```
    VALUES(NEW.map_creater,NEW.map_id);
```

```
END;
```

|

delimiter ;

D- Det skal ikke være mulig å slette en eier av et kart som administrator av dette kartet, da dette vil forårsake at en eier av et kart miste tilgang til sitt eget kart. Derfor er det opprettet følgende trigger, som sørger for å legge eieren inn igjen etter sletting:

delimiter |

```
CREATE TRIGGER Do_Not_Delete_Map_Creater_As_Admin AFTER DELETE ON  
Maps_Administrators
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF cast(aes_decrypt(OLD.username, 'enckey') AS CHAR(20)) IN (
```

```
        select cast(aes_decrypt(M.map_creater, 'enckey') AS CHAR(20))
```

```

        From Maps M

        where (M.map_id = OLD.map_id)

    ) THEN

        INSERT INTO Maps_Administrators(username, map_id)

        VALUES(aes_encrypt(OLD.username,'enckey'),OLD.map_id);

    END IF;

END;

|

delimiter ;

```

E- En person som legger inn en tilbakemelding på et kart, må ha tilgang til dette kartet som eier, administrator eller intervjuer. Dette blir implementert direkte i serverprogrammet.

F- En person som legger inn nye kategorier, spørsmål, administratorer og/eller intervjuere til et kart, må være en eier av dette kartet. Dette blir implementert direkte i serverprogrammet.

G- En person som skal ligge inn nye svar og geografiske former fra respondenter i kartet, må være medlem av dette kartet (eier, administrator eller intervjuer). Dette blir også implementert i serverprogrammet.

H- Nye geografiske figurer som punkter, stier/veier og områder skal være innenfor kartets allerede definerte grenser fra eieren av kartet. Dette blir implementert direkte i JavaScript programmet.

I- «category\_type» i tabellen «Maps\_Categories» skal kun ha en av følgende verdier (Point, Road, Area). Dette blir implementert i JavaScript programmet.

For å unngå forvirring og vranglås i databasen ved oppdatering av applikasjonens regler på ett senere tidspunkt, er valget tatt om å implementere triggere i punktene E, F, G, H, og I, direkte i Python serverprogrammet i bak-enden eller ved hjelp av JavaScript.

Det er fullt mulig at det foreligger flere restriksjoner som krever triggere og som jeg ikke har dekket her, derfor er det viktig å ta en helhetlig vurdering av databasen i sammenheng med reglene til applikasjonen før den tas i bruk.

## Kapittel 5: Autoritet- og tilgang håndtering og Sikring av nettstedet

### 5.1 Innlogging, utlogging og identifikasjon av klienter.

For å håndtere forespørsler fra klienter, må denne applikasjonen vite hvem av klientene som kommuniserer med serveren. Derfor er innloggingsmekanismen implementert i denne applikasjonen.

Innloggingsmekanismen er basert på bruk av «session» biblioteket, den lagrer informasjoner på klientens enhet som serveren kan etterspør. Dette ligner mye på måten den populære «cookies» teknologien fungerer på, bare at her er den kun brukt for å lagre brukernavnet til klienten.

Etter innlogging oppretter serveren en «session» slikt.

```
session["Logged_in"] = "Username"
```

Når brukeren logger seg ut, slettes «session» objektet, ved bruk av «pop()» metoden.

```
session.pop('Logged_in', None)
```

Når serveren mottar en forespørsel fra en klient som krever identifisering av klienten, kan brukernavnet leses fra «session» som vist i følgende eksempel, Metoden «session.get()» vil her returnerer verdien «None» dersom den ikke finner en innlogget bruker.

```
username = session.get("Logged_in", None)
```

«session» objektene blir kryptert og dermed sikret ved hjelp av en sikker nøkkel som administratoren av nettstedet bestemmer selv, koden nede er deklarerert i toppen av «CityShare.py» skriptet og viser at nøkkelen er satt til å være «any random string» noe som den ansvarlige av nettstedet bør forandre til annen nøkkel.

```
from flask import session
app = Flask(__name__)
app.secret_key = "any random string"
```

## 5.2 Autoritet- og tilgang håndtering

Ut fra beskrivelsen til denne applikasjonen ser vi at tilgang av kart, innholdet av kartet og tjenester som f.eks. tilbakemeldingsarena skal kun være tilgjengelige til medlemmer av dette kartet, hver etter sin rolle i kartet. Du kan se på Kapittel 2 for å vite mer om de ulike rollene til medlemmene av et kart.

For å løse dette gjennomføres det en rekke sjekk i bakenden av applikasjonen ved de forespørslene som krever en autoritetserklæring. Dette gjøres for å garantere at klienten ikke skal få tilgang til kart og data som han/hun ikke har autorisert tilgang til.

I Python skriptet «CityShare\_Flask/Classes.py» har vi klassen «Authority» som inneholder 4 metoder, disse metodene brukes i «CityShare.py» for å sjekke hvem har tilgang til hva. Ei liste av disse metodene er listet nedfor med tilsvarende returverdi og mulige signalverdier for forsøk på uautorisert tilgang.

Signal verdiene er følgende

- 0: Ukjent Feil, inkluderer alle andre feil som er ikke listet nedenfor, og fil bli skrevet i en loggfilen «CityShare\_Flask/Error\_log.txt», se kapittel 6 for mer om dette.
- 1: Feil ved kommunikasjon med MySQL databasen eller under spørringer.
- 2: Personen som er innlogget har ikke tilgang til data
- 3: Dette kartet er gått ut på dato og kan ikke åpnes lenger i siden «open\_map.html», med mindre eieren utvider gyldighetsdatoen ved å oppdatere kartet i siden «update\_map\_details.html».

- 4: Ingen person er logget inn, dvs. verdien fra «session.get("Logged\_in", None)» er null.

Etter at en forespørsel blir mottatt fra klienten vil en sjekk av autoritet gjennomføres av serveren, og dersom den brukeren ikke har tilgang til det han forsøker å få tilgang til, vil siden «error.html» bli returnert til brukeren for å informere om hva som har skjedd basert på signalverdiene nevnt ovenfor.

Hvis metoden i serveren er satt opp til å motta «Ajax POST» forespørsler brukes det heller JavaScript «alert» meldinger for dette, dette er fordi «Ajax POST» forespørsler er ment å kommunisere med serveren uten å oppdatere den nåværende nettsiden.

Metode	Return verdi ved vellykket tilgang	Signalverdier ved ikke vellykket tilgang
<b>def</b> get_all_map_users( <b>self</b> , map_id):	Brukernavn og e-post for intervjuere av oppgitt kart Og Brukernavn og e-post for administratorer av oppgitt kart	(0, 0) (1, 1)
<b>def</b> is_logged_in( <b>self</b> ):	True hvis en bruker er registrert i «session» False hvis ikke	Ingen, signalverdien 4 blir heller sendt fra CityShare.py etter en sjekk der.
<b>def</b> get_map_users( <b>self</b> , map_id):	Liste med brukernavn til både administratorer og intervjuere av dette kartet	0 1 2
<b>def</b> is_map_valid( <b>self</b> , map_id):	True hvis kartet har fortsatt en gyldig dato  Dette brukes kun i «open_map.html», etter	0 1 3

	gyldighetsdatoen skal ikke noen kunne legge inn nye figurer og svar fra respondenter i kartet, dvs. at siden «open_map.html» ikke skal kunne åpnes med mindre gyldighetsdatoen utvides av eieren.	
--	--	--

### 5.3 Kryptering og sikring av brukernavn og passord i databasen

I databasen utgjør brukernavnet en ID for hver bruker av applikasjonen, derfor er den unik for hver bruker, mens passordet som brukeren benytter for å logge seg inn er også viktig fordi den i kombinasjon med brukernavnet identifiserer brukeren ved innlogging.

For å beskytte brukernavnet og passordet i databasen. krypteres attributtene «username» og «login\_pass» i alle tabeller.

Dette skjer ved innsetting/oppretting av en ny bruker og med hjelp av MySQL-funksjonene «aes\_encrypt(value, encryption\_key)» som krypterer verdien av disse feltene ved innsetting og «aes\_decrypt(attributt, encryption\_key)» som dekrypterer ved lesing.

Et eksempel på en slik kryptering ved innsetting er følgende MySQL setning er vist under, i dette eksemplet og i vår applikasjon brukes «enckey» som er nøkkelen for krypteringen, dette bør forandres til en annen hemmelig nøkkel som den sikkerhetsansvarlige for databasen og nettsiden bestemmer:

```
INSERT INTO Persons(username, login_pass, first_name, last_name, e_post, telephone)
VALUES(aes_encrypt('Oli1992','enckey'),aes_encrypt('Olipassword','enckey'),'Ole','Nord
mann', 'example@yahoo.com', 004700000000);
```



Dette vil sikre at dersom noen andre får tilgang til databasen så vil innholdet av disse to attributtene være kryptert og kan ikke ses av noen andre som ikke har krypteringsnøkkelen «enckey», eksemplet nedenfor viser hvordan feltene blir vist ved en vanlig «SELECT» setning fra MySQL databasen etter disse er blitt kryptert.

username	login_pass	first_name	last_name	e_post	telephone
lUJNof: 'dU> SŠ	âDu6CcÁ«ulo^fibĐ	Ole	Nordmann	example@yahoo.com	4700000000

Encrypted values of  
username and  
password

**Figur 5, Brukernavnet og passordet er nå kryptert i databasen.**

For å lese av disse verdiene, må den nøkkelen som er brukt for å kryptere disse verdiene inkluderes i MySQL «SELECT» setningen. Et eksempel er vist nedenfor

```
SELECT cast(aes_decrypt(username, 'enckey') AS CHAR(20)) FROM Persons WHERE  
username = aes_encrypt('Oli1992','enckey');
```

Den forrige MySQL setningen vil da returnere verdien «Oli1992» som er brukernavnet for denne brukeren.

Kryptering av brukernavn vil også skje i alle andre tabeller der et attributt refererer til denne personens primærnøkkel «username», et eksempel på dette er «Shapes» tabellen der for hver figur skal det registreres hvem av kartmedlemmene som har lagt denne figuren til kartet.

## Kapittel 6: Feilhåndtering

Feil som oppstår mens serverprogrammet håndterer forespørsler er delt i to hovedtyper.

Den ene type feil kan oppstå ved kommunikasjon med databasen der programmet reiser et unntak av typen «mysql.connector.Error». En slik feil kan forårsakes av f.eks. feil i MySQL syntaksen, eller mangel på verdier. MySQL databasen returnerer en feilmelding som forteller noe om hvorfor feilen har skjedd

Alle andre typer feil må forekomme fra Python programmet som kjører og skal inneholde en feilmelding, sammen med metoden og sekvens linjenummeret der feilen har oppstått.

For å gjøre informasjon om feil lett tilgjengelige for administratoren av nettsiden, logges alle feil som oppstår i en tekstfil med følgende sti og navn i prosjektmappen:

«CityShare\_Flask/Error\_log.txt». Hver rad i denne filen representerer en feil som har oppstått ved denne applikasjonen.

En logget feil i denne filen skal inneholde tidspunktet(dato og klokke) i servertid, brukernavnet til brukeren som har opplevd feilen, feilmeldingen, navnet til metoden i serverprogrammet der feilen oppsto, og hvilken linje i programmet forårsaket denne feilen. Et eksempel på hvordan den filen kan se ut er gitt under:

```
TIMESTAMP, LOGGED_USER, ERROR_MESSAGE, METHOD, LINENUMBER
2018-05-06 12:13:51.875184, ErikKonge, too many values to unpack (expected 2),
view_resondent_answers, 1226
2018-05-06 12:13:58.988797, Ole1992, too many values to unpack (expected 2),
view_resondent_answers, 1226
2018-05-06 12:19:37.167191, OlavS, too many values to unpack (expected 2),
view_resondent_answers, 1226
```

Feil fanges opp i programmet ved hjelp av «try» og «except» kontrollstrukturen og en feilmelding blir skrevet i filen ved å bruke metoden «log\_error» i klassen «Error\_log» som er inkludert i filen «CityShare\_Flask/Classes.py». Følgende kode viser et eksempel

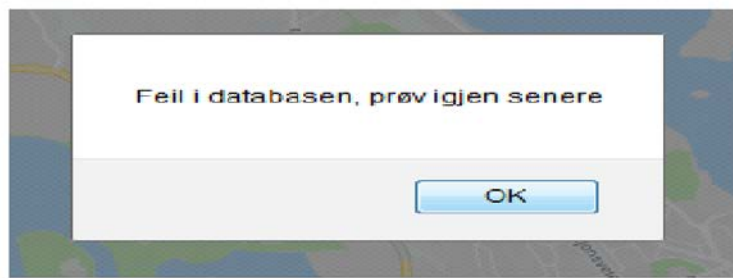
på hvordan man bruker «Error\_log» klassen for å logge inn nye feil i filen «Error\_log.txt»:

```
try:
    ...
except mysql.connector.Error as err:
    Error_logging = Classes.Error_log(session.get('Logged_in', None), str(err.msg))
    ...
except Exception as err:
    Error_logging = Classes.Error_log(session.get('Logged_in', None), err)
    ...
```

I klassen «Error\_log» fanges informasjon om feilen ved å bruke Python bibliotekene «sys, os, traceback» på denne måten.

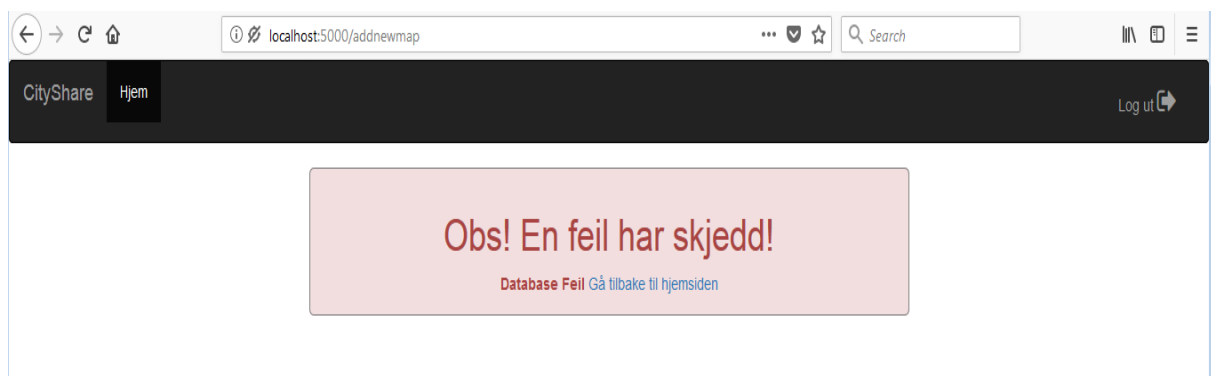
```
def __init__(self, username, error_msg):
    self.username = username # instance variable unique to each instance
    exc_type, exc_obj, exc_tb = sys.exc_info()
    tb = traceback.extract_tb(exc_tb)[-1]
    self.log_error(error_msg, tb[2], tb[1])
```

Brukeren skal også få varsel om feilen som har oppstått, dersom forespørselen er av typen JavaScript «AJAX» så skal en «alert» melding som det ser ut på bildet nedfor bli vist på skjermen.



**Figur 6, En varsel boks som informerer brukeren om feilen som har oppstått som et resultat av en "Ajax" forespørsel.**

Hvis forespørselen forventer ett HTML-dokument som respons, vil brukeren motta siden med navnet «error.html» som gir en melding om feilen og inneholde en linke som sender brukeren til hjemmesiden igjen. Et eksempel av denne siden er vist nedenfor.



**Figur 7, siden "Error.html" informerer brukeren om feil og tilgangsrrettigheter**

## Kapittel 7: Applikasjonens struktur og innhold

Dette kapittelet inneholder informasjon om filsystemet i delkapittel 7.1.

I delkapittelet 7.2 blir Bak-ende implementasjonen redegjort.

Deretter forklares Front-enden som inkluderer det grafiske grensesnittet til brukeren av applikasjonen sammen med de viktigste bitene av JavaScript koden som er brukt i hver nettside.

### 7.1 Filsystemet i prosjektet

Dette prosjektet er delt inn i 3 deler, en mappe for hver del:

- CityShare\_Flask: Som er Flask prosjektet og er selve applikasjonen.
- CityDataAndroid: Som utgjør en Web ramme applikasjonen til nettsiden i CityShare\_Flask for operativsystemet Android.
- City Data iOS: Som utgjør en Web ramme applikasjonen til nettsiden i CityShare\_Flask for operativsystemet «Apple iOS».

A- «CityShare\_Flask» er Flask prosjektet som utgjør selve applikasjonen og er ansvarlig for mottakelse og respons av HTTP forespørsler fra og til klienter. Hele Flask prosjektet eksisterer i CityShare\_Flask mappen, inni den mappen finner vi følgende

- «static»: Her ligger blant annet importerte ikoner, plugins, database dokumentasjon, bilder, JavaScript filer og CSS filer for styling av HTML dokumentene.
- «templates»: Her finnes de forskjellige HTML-dokumentene som sendes som respons til klienter gjennom «render\_template» motoren.
- «CityShare.py»: Ved å kjøre dette skriptet vil nettsiden bli aktivert, her ligger handlingene av de forskjellige HTTP forespørslene med en unik URL for hver forespørsel.
- «Classes.py»: Inneholder klasser og metoder som hjelper med autoritet-, tilgang-, og feilhåndtering.
- «Error\_log.txt»: Alle feil som oppstår i prosjektet vil bli logget til denne filen, en ny rad for hver ny feil.

- «read\_icons.py»: Er en Python skript som skal lese alle ikoner for punktkategorier og som blir lagt i filen «static/icons» for deretter å kopiere HTML «Option» elementene og lime disse i «Select» listene for punkt kategorier i «new\_map.html» og «update\_map\_details.html».
- «used\_queries.txt»: den inneholder Alle SQL Setninger brukt i dette prosjektet, spørringene i denne filen er brukt under analysing av databasen for å danne et grunnlag om valg av effektive indekser og normaliseringsformer i de forskjellige tabellene i databasen

Ofte er det en tilsvarende JavaScript og CSS fil til hver av html malene «templates», men i noen tilfeller trengs ikke dette, da siden ikke har en JavaScript kode, og i noen andre tilfeller kan to HTML-dokumenter dele samme kode som f.eks. CSS koden for dokumentene «open\_map.html» og «new\_map.html». filen vil ha følgende navn «open\_and\_analyze\_map\_stylesheet\_version\_0.css».

De fleste nettlesere lagrer JavaScript og CSS filer lokalt i enheten til klienten for å ikke behøve å sende en ny forespørsel for filen dersom nettsiden blir besøkt på nytt. Derfor er versjonstallet tatt med i filnavnet til JavaScript og CSS filene slik som vist under.

static/Scripts/analyze\_map\_JavaScript\_version\_0.js

static/Stylesheets/open\_and\_analyze\_map\_stylesheet\_version\_0.css

Dersom en av JavaScript eller CSS filene oppdateres, bør versjonstallet i navnet til filen inkrementeres med 1 og oppdateres i importeringssetningen i tilsvarende HTML-dokument. Slik vil nettleseren til klienten sende en ny forespørsel til den filen selv om den gamle filen er lagret i loggen til klientens nettleser.

B- «CityDataAndroid» er en applikasjon skrevet i programmeringsspråket Java og implementerer nettsiden i en web ramme. Denne applikasjonen fungerer kun på operativsystemet «Android».

C- «City Data iOS» er en applikasjon skrevet i programmeringsspråket Swift, og skal også implementere nettsiden i en web ramme for operativsystemet «apple iOS».

Temaet om Web ramme applikasjonene og hvordan disse blir implementert vil bli diskutert senere i kapittel 9.

## 7.2 Server Program – «Bak-Ende»

Serversiden av applikasjonen mottar og behandler forespørsler fra klienter av applikasjonen, All forespørsler som sendes til serveren fra klienten skal ha en global IP adresse, et portnummer, og en URL. De to første parameterne trenger ikke en bruker å oppgi direkte, da et såkalt «DNS» innslag i en «DNS» server blir ofte opprettet. For denne applikasjonen er dette «www.cityshare.ux.uis.no» som er lik «152.94.1.67:5000/URL».

En flask applikasjon definerer slike parametere slik som det vises under. Vi ser at server adressen er av typen IPv4 og den er for vår applikasjon «152.94.1.67» sammen med at port nummeret er «5000», eksemplet viser også hvordan man deklarerer metode «**def** signup():» som behandler forespørsler fra klienter, URL-en her er «**/signup**», deklart med metoden «**@app.route('/signup', methods=['POST', 'GET'])**».

```
@app.route('/signup', methods=['POST', 'GET'])
def signup():
    if request.method == 'GET':
        ## Eventuell kode som serveren skal utføre kommer her
        return render_template("signup.html")

if __name__ == '__main__':
    app.run(host='152.94.1.67', port=5000)
```

Som vi ser fra eksemplet støtter denne metoden 2 typer forespørsler, både «POST» og «GET». «GET» forespørselen brukes ofte til og etterspør data fra en server og i denne

applikasjon er data et html-dokument, «GET» forespørsler kan se slik «'/signup?name1=value1&name2=value2'» der «name1» og «name2» utgjør noen parametere som sendes til serveren sammen med noen verdier.

«POST» forespørsler derimot brukes ofte til å sende inn data til serveren og klienten kan også forvente en respons av serveren.

Merk at «GET» forespørsler kan sendes gjennom URL-feltet i nettleseren og dermed kan verdiene lett bli manipulert av klienten ved for eksempel å erstatte «value1» med «value2», derfor er «POST» forespørsler mer foretrukket enn «GET» for å sende data fra klienten til serveren. Men selv om «POST» forespørsler er sett på som mer sikre så kan disse forespørlene sendes av kommandolinjen til operativsystemet. Derfor er det også viktig at sjekking av rettigheter til data skal gjennomføres før en respons blir returnert til klienten. Se kapittel 5 for mer informasjon om autoritet og rettigheter.

Tabellen under oppsummerer alle URL linkene som er brukt i prosjektet og som er implementert i hovedfilen i Flask applikasjonen «CityShare.py»

URL	HTTP metode	Hensikt, og vellykket returverdi
/	GET	Åpner hovedsiden  Returnerer «home.html»
	POST	Registrerer en ny bruker ved å motta data fra formen i «signup.html»  Returnerer «home.html»
/signout	GET	Logger ut brukeren ved å fjerne «session».  Returnerer «home.html»
/login	POST	Mottar data fra innloggingsformen i «home.html», og opprettet en «session» ved vellykket innlogging.
/Update_Account	GET	Returnerer «update_account.html»



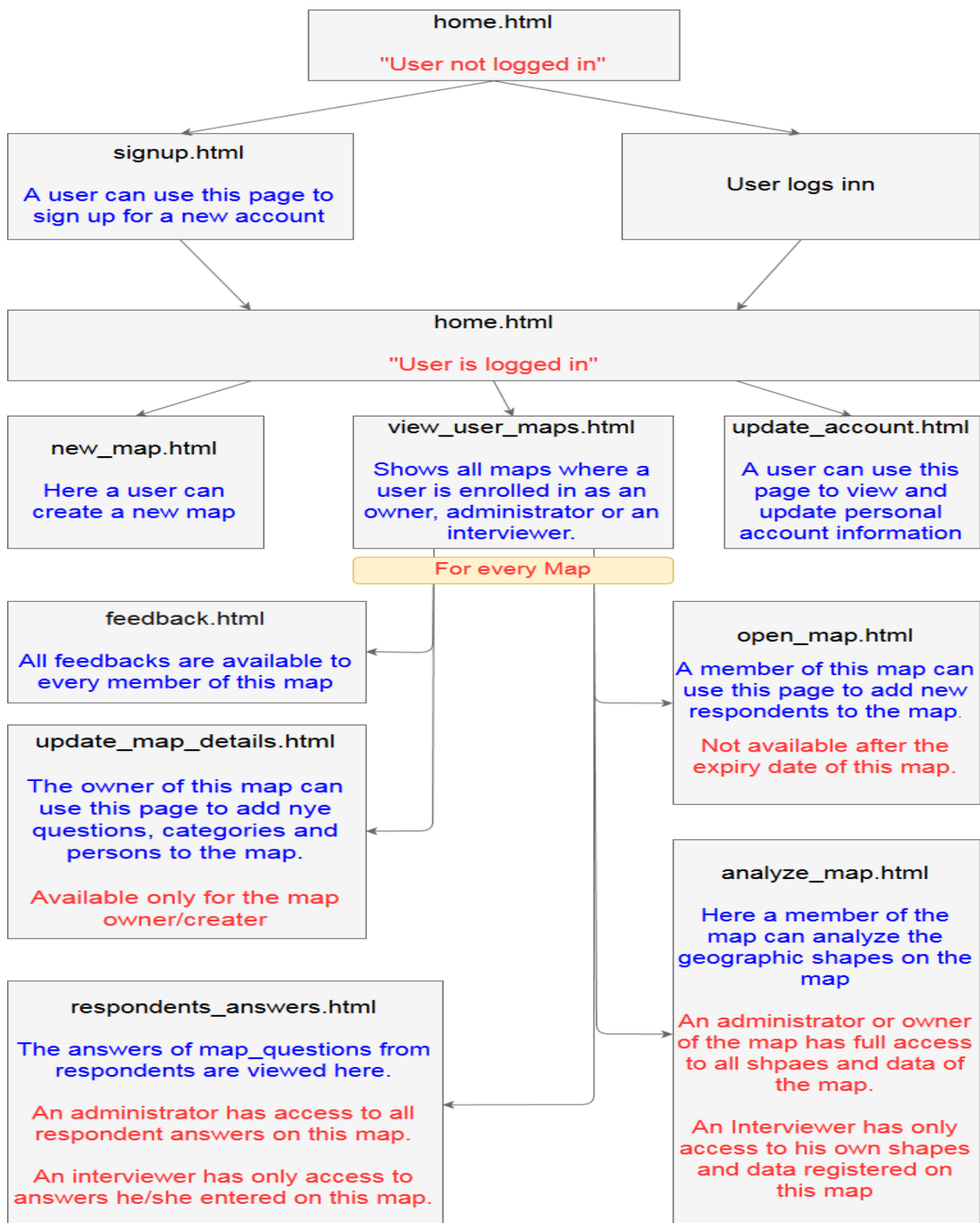
	POST	Oppdaterer kontoen til brukeren ved å motta data fra formen i «update_account.html»  Returnerer «home.html»
/searchUsers	POST	Søker etter en bruker med brukernavn eller e-post adresse ved å motta data fra «new_map.html»  Returnerer den samme verdien den mottar hvis den finner brukeren i «Persons» tabellen i databasen.
/addnewmap	GET	Returnerer «new_map.html»
	POST	Mottar all registrerte data om ett nytt kart fra input elementer og skjemaet «form» i new_map.html, for deretter å lage et nytt kart.  Returnerer «home.html»
/showmaps	GET	Leser alle kart brukeren er medlem av.  Returnerer «view_user_maps.html»
/EditMap	GET	Returnerer «open_map.html», der brukeren kan legge inn nye respondenter og lar de svare på spørsmål og tegne geografiske figurer
/RegisterRespondoent	POST	Mottar data fra metoden «function SaveRespondoent()» som er en JQuery Ajax metode implementert i html siden «open_map.html». og lagrer data om den nye respondenteren i databasen.

		Returnerer nøkkelordet "Success" hvis vellykket lagring blir utført
/ViewMapResult	GET	Returnerer analysesiden «analyze_map.html», som er den siden brukeren analyserer ett kart i ved å vise alle figurer som kartet inneholder fra alle intervjuere for en administrator og fra kun selve brukeren for en intervjuer.
/Edit_Shape	POST	Redigerer en figur fra kartet (punkt, sti eller område) ved å endre en eller flere av følgende: tittel, beskrivelse, eller vurdering. Eller sletter figuren fra kartet.  Returnerer nøkkelordet "Success" hvis vellykket lagring blir utført
/DownloadMapAsCSVFile	POST	Returnerer innholdet av en «CSV» fil som inneholder informasjon om respondenter og tilsvarende figurer mottatt av metoden «function Download_Map_AS_CSV_File()» i «analyze_map.html». Responsen sendes til klienten som en «String» og JavaScript metoden «function download(filename, text)» tar seg av nedlastningen som en fil med «CSV» formatet ned til klientens datamaskin.
/UpdateMapDetails	GET	Returnerer «update_map_details.html» som er den siden eieren av kartet kan legge inn nye intervjuere, administratorer, figurkategorier og spørsmål til kartet i et senere tidspunkt etter opprettelse av kartet.
	POST	Mottar data som nye detaljer, kategorier,

		<p>spørsmål, administratorer og intervjuere fra «update_map_details.html» som eieren av kartet ønsker å legge til kartet.</p> <p>Returnerer «view_user_maps.html»</p>
/FeedbackArena	GET	<p>Leser tilbakemeldinger av dette kartet og returnerer «feedback.html» som er en tilbakemeldingsarena hvor medlemmer av kartet kan legge sin tilbakemelding på</p>
	POST	<p>Mottar og registrerer en tilbakemelding fra «feedback.html» når klienten trykker på send knappen.</p> <p>Returnerer det registrerte fornavnet og etternavnet til denne brukeren som er innlogget hvis lagring av tilbakemeldingen er vellykket.</p>
/ViewRespondentAnswers	GET	<p>Returner «respondents_answers.html» som inneholder alle registrerte svar fra respondenter på de spørsmål som kartet inneholder.</p> <p>Eier og administratorer av kartet skal se alle svar, mens intervjuere kun skal se svar som fra respondenter som de selv har intervjuet.</p>

I metodene nevnt i tabellen ovenfor kan feil av ulike årsaker oppstå, eller kan rettigheter til data forårsake en signalverdi av en feil ved sjekking av rettigheter, derfor er det viktig å legge merke til at andre returverdier som inkluderer en signalverdi av en feil eller en omdirigering av trafikk til en annen URL som f.eks. ved bruk av «redirect()» metoden er aktuell for hver URL. Mer om feilhåndtering i applikasjonen er klargjort i kapitel 6.

### 7.3 Grafisk brukergrensesnitt



Figur 8, Et skjema som viser hvordan brukeren navigerer fram til de ulike sidene i applikasjonen. teksten i blå forklarer hva siden er ment til å vise, mens den røde teksten forteller kort om tilgangsrettigheter til siden.

I tillegg til disse sidene er det en «error.html» siden som skal sendes til brukeren dersom en feil oppstår i forespørslene.

All disse nettsidene utvider «index.html» som inneholder en navigasjonsbar på toppen sammen med felles HTML-struktur for alle disse sidene. Hovedstrukturen til koden i «index.html» er vist nedenfor, her ser vi at det er tre blokker opprettet. En blokk for «CSS» kode, en for «JavaScript» kode og en for selve det grafiske grensesnittet som er det HTML-innholdet i siden som utvider «index.html».

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CityShare</title>
  <!-- Common imports between all templates comes here -->
  {% block stylingblock %}

  {% endblock %}

  {% block scriptblock %}

  {% endblock %}
</head>
<body>
  <!-- Common navigationbar for all templates comes here -->
  {% block content %}

  {% endblock %}
</body>
</html>
```

Et eksempel på hvordan en ny side skal utvide «index.html» er gitt under, mellom «block» og «endblock» kan vi programmere inn ønsket innhold for denne nettsiden.

```
{% extends 'index.html' %}
```

```
{% block stylingblock %}
```

```
<!-- CSS kode og/eller import setninger -->
```

```
{% endblock %}
```

```
{% block scriptblock %}
```

```
<!-- JavaScript/JQuery kode og/eller import setninger -->
```

```
{% endblock %}
```

```
{% block content %}
```

```
<!-- GUI HTML Elementer kommer her -->
```

```
{% endblock %}
```

## 7.4 hjemmesiden «home.html», registrering av nye brukere i «signup.html», og oppdatering av kontoopplysninger i «update\_account.html».

Hjemmesiden «home.html» inneholder to seksjoner den ene vises dersom brukeren ikke er innlogget, dvs. at et brukernavn ikke eksisterer i en «session» fra den klienten og den andre seksjonen vises dersom brukeren er innlogget.

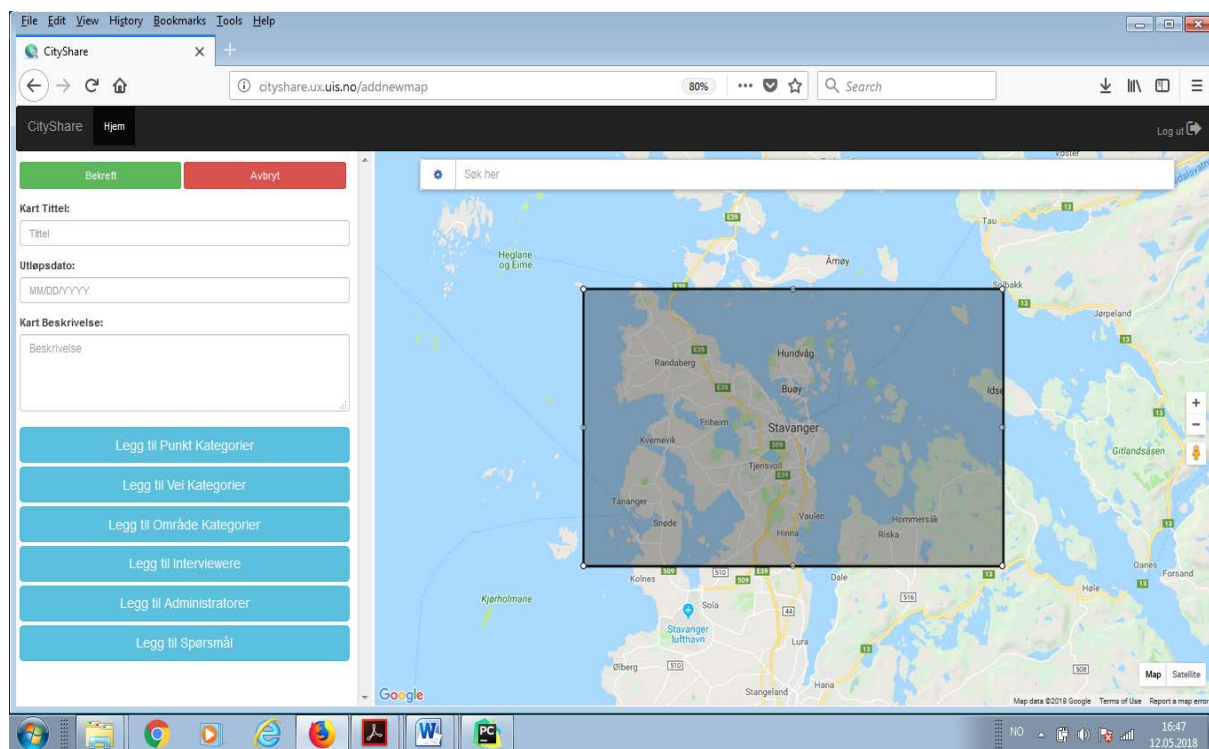
- Ved ikke innlogget bruker skal det til venstre vises en Demo video som forklarer hvordan en klient kan bruke applikasjonen, mens på høyre side skal det stå en skriftlig forklaring av applikasjonen
- Når brukeren logger inn, får han/hun tilgang til en hjemmeside som inkluderer tre mulige bokser som det går an å trykke på; en for opprettelse av et nytt kart, en for visning av alle registrerte kart som brukeren er medlem av og en for visning/oppdatering av personopplysninger til brukeren.

Registrering av nye brukere kan skje ved å klikke på en knapp til høyre i navigasjonsbaren som sender brukeren til siden «signup.html», den siden inneholder en form som skal fylles ut av brukeren og sendes til serveren.

Hvis brukeren ønsker å vise eller oppdaterer sine personlige opplysninger som fornavn eller telefonnummer kan han/hun etter å ha logget seg inn trykke på «min konto» boksen som sender brukeren til siden «update\_account.html» der gjeldende opplysninger vises og kan endres, med unntakelse for brukernavnet som ikke kan endres da den kan være knyttet til allerede eksisterende kart og data i databasen.

Dersom en feil oppstår sender serveren siden «error.html» som inneholder en link som brukeren kan trykke på for å gå tilbake til hjemmesiden.

## 7.5 Opprettelse av nytt kart i siden «new\_map.html»



Figur 9 HTML-siden «new\_map.html» der en bruker kan opprette et nytt kart.

Det grafiske brukergrensesnittet av denne siden inneholder:

- Et kart importert fra «Google Maps API».
- Et valgpanel med de forskjellige input verdiene til kartet.

Kartet skal alltid inneholde en firkantet, gjennomsiktig og lett farget boks som er plassert i midten av kartet og er ca. halvparten så stor som kartet. Denne boksen representerer det geografiske området som det nye kartet skal være i dvs. grensene til det geografiske området som er aktuell for undersøkelsen. Boksen tilpasser seg kartet dersom brukeren forandrer lokasjonen i kartet, og skriver både høyde over bakken(zoom) og grensene til et skjult HTML-element av typen «input» med forholdsvis ID-ene `#geo_boundary` og `#geo_zoom`.

Valgpanelet inneholder et skjema som skal fylles ut av brukeren med følgende verdier:

- Kartets tittel, beskrivelse, og dato.
- Punkt-, vei/sti- og område kategorier til det nye kartet
- Spørsmål som skal stilles til respondenter i undersøkelsen.
- Administratorer og Intervjuere av dette kartet.

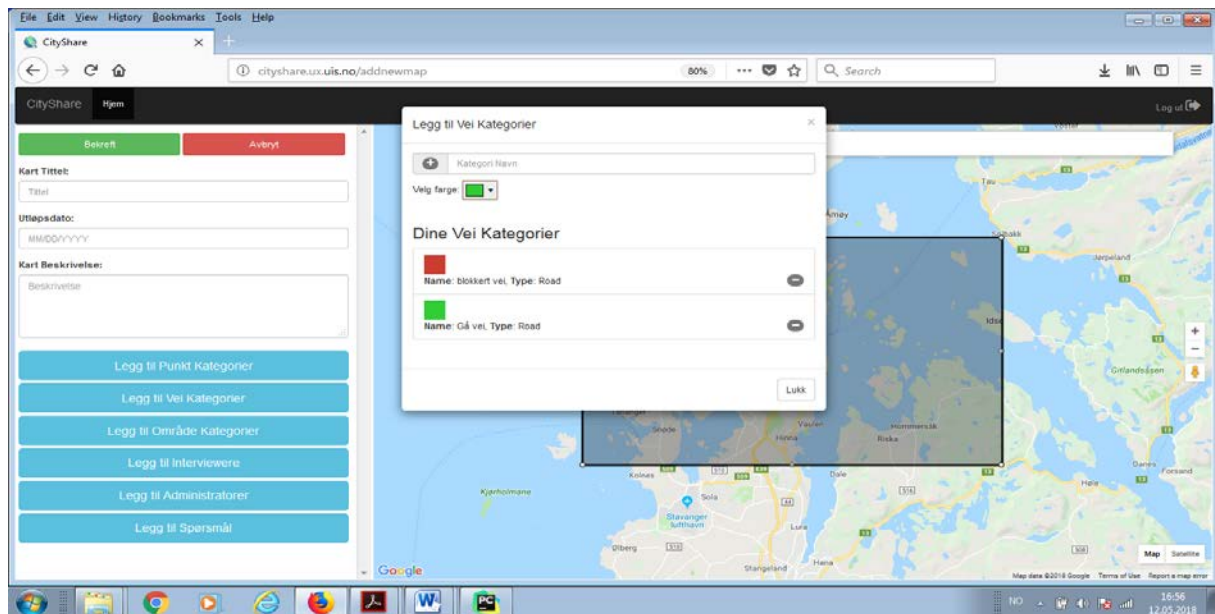
Attributtet «required» blir brukt i feltene for tittel, beskrivelse og dato for å stoppe utsendelse av skjemaet dersom disse feltene ikke er fylt ut.

For å sørge at brukeren legger inn minst en administrator, en intervjuer, et spørsmål og en av de tre kategoriene, er det opprettet følgende lytter på utsendelsesknappen

«`$("#form").submit(function (e) { ... })`», denne skal kontrollere at kravene er oppfylt og den gir en beskjed til brukeren om mangler i formen.

For å legge til kategorier(punkt, vei/sti og områder), spørsmål, administratorer, og intervjuere er det inkludert en HTML-boks av typen «Bootstrap Modal» til hver av disse. Disse boksene inkluderer HTML-elementene som er nødvendig for hver enkelt inputtype sammen med ei liste over input som er blitt lagt til. Et eksempel på dette er ei liste over alle spørsmål som brukeren har lagt til skal vises under feltet for å slå inn nye spørsmål.





Figur 10, figuren viser «Bootstrap Modal» boksen og innholdet av den ved spesifisering av sti kategorier fra brukeren.

Lyttermetodene nedenfor tar hånd om adderingsmekanismene i hver av de «Modal» boksene

```

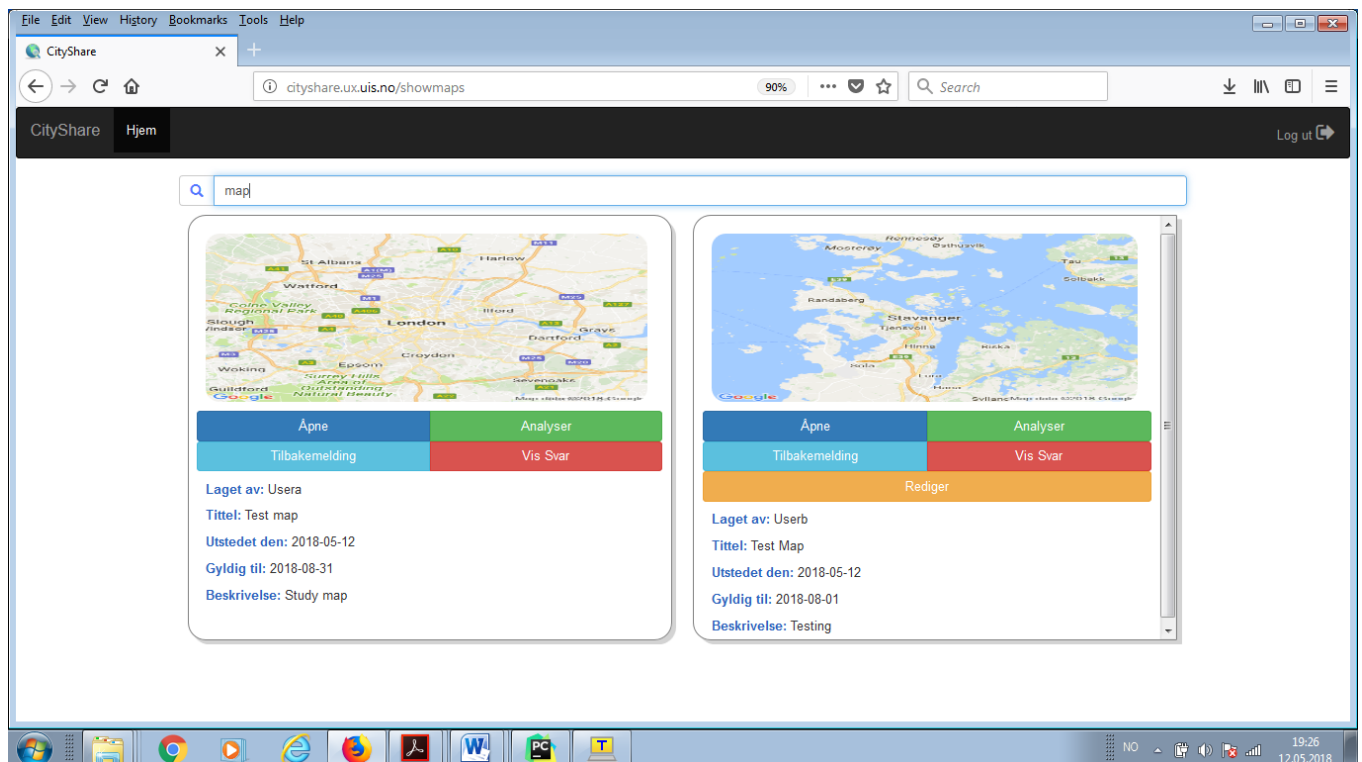
$("#add_point_category").on('click', ...);
$("#add_road_category").on('click', ...);
$("#add_area_category").on('click', ...);
$('#SearchInterviewersButton').on('click', ...);
$('#SearchAdministratorsButton').on('click', ...);
$('#AddQuestionButton').on('click', ...);

```

Det er kanskje mulig å slå inn disse lytterfunksjonene i en eneste funksjon fordi det er små forskjeller i mellom disse, men for å holde funksjonene mest mulig atskilte har jeg valgt å ha hver lytter funksjon for seg selv.

JavaScript koden for denne filen er skrevet i filen med følgende sti og navn: «static/Scripts/new\_map\_JavaScript\_version\_0.js», her finner vi kode for «plugins» som er redegjort i kapittel 3.5 og skjermtilpasning som er redegjøres i kapittel 8.

## 7.6 Visning av alle kart som brukeren er medlem av i siden «view\_user\_maps.html»



Figur 11, siden "view\_user\_maps.html" viser alle kart som en bruker er medlem av, kart med den Orange redigeringsknappen er eid av brukeren som er innlogget, mens andre kart har brukeren som enten administrator eller intervjuer, og er dermed eid av andre brukere.

I denne siden skal alle kart som den innloggede brukeren er medlem av enten som administrator eller intervjuer stå. Det grafiske grensesnittet er bygget opp slik at hvert kart er inkludert i en «div» boks med klassen «.map\_box», og hvert kart har en ID lik «#Map\_{{ map.mapid }}» der «{{ map.mapid }}» er kart ID-en fra databasen.

Disse kartene blir lest fra databasen i serveren etter en «GET» forespørsel til følgende URL: «/showmaps» og deretter sendes disse i «render\_template» motoren og resultatet blir et HTML-dokument som inneholder alle kart som den innloggede brukeren i «session» er medlem av.

### 7.6.1 Søking blant kart i siden «view\_user\_maps.html»

Når brukeren er medlem av mange kart samtidig kan det å komme fram til riktig kart bli litt utfordrende, derfor er det lagt på toppen av siden en tekstfelt som skal fungere som et filter.

Filtreringsmekanismen er implementert ved bruk av JavaScript/JQuery. Denne koden bruker Jinja Syntaks fra «render\_template» for å generere tilpasset JavaScript kode og bør stå i blokket «**scriptblock**» i «view\_user\_maps.html».

JavaScript Metoden «**function** read\_maps()» bruker Jinja Syntaksen for å lese inn ID-ene til alle kart og returnerer en matrise som inneholder alle kart ID-ene i siden.

«\$("#filter").on("keyup keydown", **function**() { ... })» er en JQuery lytter på inntasting i filtertekstfeltet.

Funksjonen til lytteren er å søke i gjennom de forskjellige kartene som er listet opp basert på tittel, beskrivelse, eier, opprettelsesdato, og utløpsdato samtidig.

Dersom det brukeren har skrevet i feltet passer med noen av kartene vil disse kartene bli vist til brukeren ved hjelp av følgende JQuery CSS funksjon

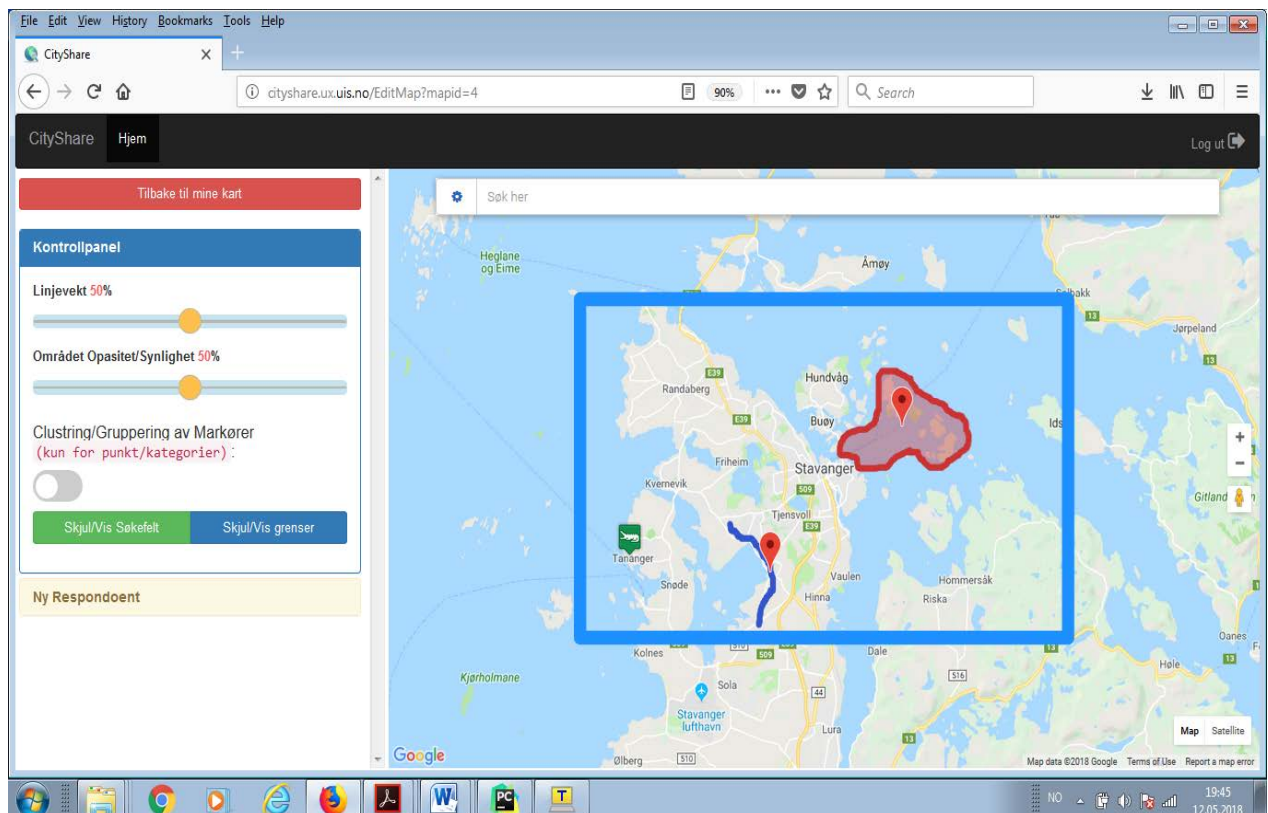
«\$("#Map\_" + key).css("display", "inline-block");» der «key» er en variabel som er lik ID-en på dette kartet og slik vil dette passe med tilsvarende «div» boks til kartet. Samme setning blir brukt for å skjule de kartene som ikke passer til søket men i stedet for «**display**", "inline-block"» brukes det «**display**", "none"» for å skjule tilsvarende «div» boks til kartet.

Dersom søket ikke passer med noen av kartene vil brukeren få en «Bootstrap» varsel av klassen «alert-warning» som gir beskjed til brukeren at ingen resultater er funnet.

Filteret bruker JavaScript for å søke blant kart, noe som sparer på datamengden brukt for å kommunisere med serveren. men dersom brukeren blir lagt til nye kart av andre brukere underveis mens han filtrerer vil ikke de nye kartene bli vist før brukeren laster ned siden på nytt, dette er ikke kritisk for denne applikasjonen. Det som er nyttig her er at brukeren slipper å bruke datamengde fra Internett-leverandøren «ISP» for å filtrere kart og at

klientens PC kan ta seg av en slik prosess. På den måten vil filtreringen vanligvis gå raskere enn ved å filtrere med forespørsler fra serveren.

## 7.7 Registrering av nye respondenter på kartet gjennom siden «open\_map.html»



**Figur 12, i siden "open\_map.html" kan brukeren intervju nye respondenter og registrere data i kartet.**

Som nevnt tidligere i applikasjons beskrivelse i kapittel 2, innsetting av data vil skje gjennom en samtale mellom intervjueren og respondenten. Respondenten skal stilles de spørsmålene som er linket til dette spesifikke kartet og svar kan fylles på av brukeren av kartet, det er også mulig å la være å registrere et svar på noen eller alle de spørsmålene.

Sammen med svarene har intervjueren anledningen til å tegne på kartet en eller flere av punkter, veier/stier, og områder. Denne siden skal sørge for denne funksjonaliteten.

Jinja Syntaksen brukes her av serversiden for å generere den aktuelle JavaScript Koden, noe som er nødvendig for å kunne ha ett HTML-dokument som har de definerte kartdetaljene, figurkategoriene og spørsmålene.

Problemet her er at Jinja Syntaksen kan dessverre kun brukes for filer i folderen «templates». For denne grunnen er plasseringen av JavaScript koden delt mellom HTML-dokumentet «open\_map.html» og filen «open\_map\_JavaScript\_version\_0.js» som importeres av denne nettsiden, altså «open\_map.html».

«open\_map.html» vil inneholde både et kart og et valgpanel med samme størrelse som i «new\_map.html». På den høyre side er det et kart fra «Google Maps API» og til venstre er det valgpanelet.

Skjermtilpasning er også implementert her, koden her er identisk med «new\_maps.html». se kapittel 8 for mer detaljer om skjermtilpasning. Nedenfor forklares JavaScript koden bak denne nettsiden.

Følgende variabler og metoder er definert for å håndtere data i siden:

I. var map = Kart Objektet

II. var Listeners = en matrise av en gruppe hendelseslyttere som blir lagt til kartet, dette er for å kunne fjerne gamle hendelseslyttere når brukeren bytter fra en type kategori til en annen til å tegne på kartet.

III. var open\_info\_window = holder styr på hvilket informasjonsvindu er åpent for øyeblikket, den har verdien null dersom ingen vinduer står åpne, merk også at det er kun mulig å vise et vindu om gangen. Denne variabelen hjelper også oss med å løse dette.

IV. var shapes = en matrise av alle figurer som intervjueren tegner på kartet på vegne av respondenten.

V. var current\_shape\_id = holder figur ID-en som gjelder akkurat nå slik at den og bare den kan slettes av «shapes» matrisen hvis brukeren ønsker dette.

### 7.7.1 Kartet fra «Google Maps API»

Kartet skal være justert til det geografiske området som ble definert under opprettelsen av kartet. Dvs. at brukeren skal ikke kunne gå ut av disse grensene.

For å gjøre dette mulig leses grensene på kartet fra databasen og ved hjelp av «Jinja»

Syntaksen vil grensene til kartet bli definert slik:

```
northeastcorner = new google.maps.LatLng({ { map.northeastcorner } });
```

```
southwestcorner = new google.maps.LatLng({ { map.southwestcorner } });
```

Følgende lyttere vil deretter sørge for at brukeren ikke får mulighet til å gå utover grensene til kartet.

```
google.maps.event.addListener(map, 'drag', function () { .. });
```

```
google.maps.event.addListener(map, 'bounds_changed', function () { ... });
```

Noen verktøy som zoom knapper, knapp for «Street View» som gir brukeren anledning til å se bilder på bakkenivå i et sted på kartet, samt en knapp for å variere typen på kartet (Vanlig, terreng eller Satellitt), kan settes opp ved oppsetting av kartet slik som vist under:

```
mapTypeControlOptions: {  
  position: google.maps.ControlPosition.RIGHT_BOTTOM,  
},  
streetViewControlOptions: {  
  position: google.maps.ControlPosition.RIGHT_CENTER  
},  
zoomControlOptions: {  
  position: google.maps.ControlPosition.RIGHT_CENTER  
},
```

Intervjueren kan tegne på kartet hvilken som helst av de kartkategoriene (punkter, stier og områder) som ble definert av eieren ved opprettelse av kartet, og disse vil være knyttet til svarene på spørsmål fra respondenten.

En funksjon står og lytter på hendelser fra «Radio Button Group» HTML-elementet i panelet for valg av kategorier og implementerer denne tegningsfunksjonaliteten, funksjonen er deklarerert som vist under.

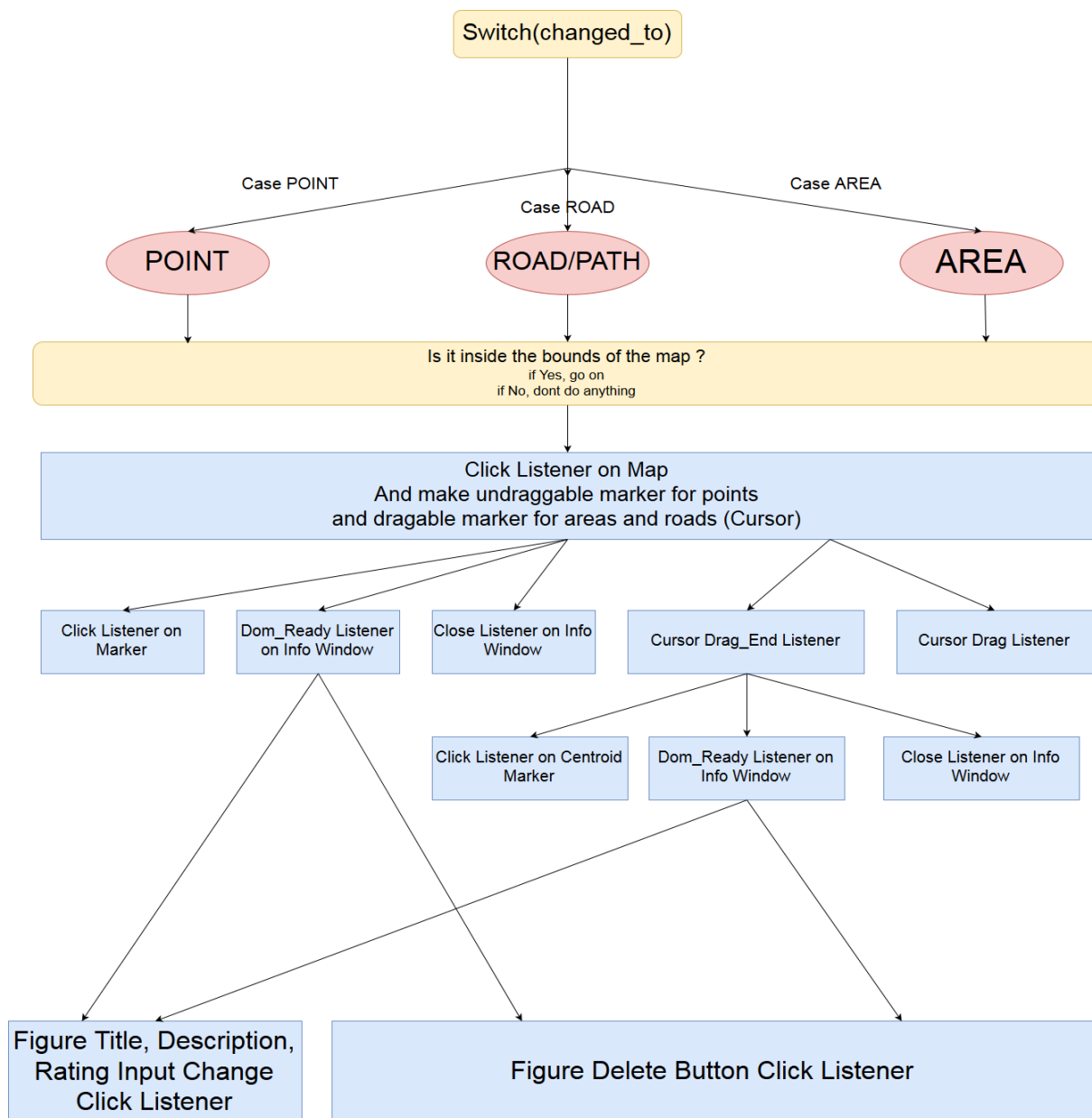
```
«$("input[name~='New_Marker']").change(function (e) { ... });»
```

Deretter fanger den hva slags ikonbilde (for punkter) eller farge (for områder og stier) denne valgte figurkategorien har. Etterpå vil den gå i et bytte kontrollstruktur «switch» som generer de aktuelle hendelseslytterne til denne figurtypen. Hendelseslyttere vil herved være nestet, dvs. at en hendelseslytter kan godt opprette en eller flere andre hendelseslyttere.

Blokkskjemaet nedenfor viser hvordan dette systemet fungerer. Pseudokoden for denne funksjonen ser slik ut:

- Bestem kategoritypen til den nye figuren
- Sjekk om brukeren har klikket innenfor grensene til kartet, hvis ikke avslutt
- Sett et tilsvarende ikon for punkter, og en flyttbar penn-ikon for veier og områder
- Dersom typen er et punkt: opprett punktet og sitt informasjonsvindu(som inneholder input HTML-elementene for tittel, beskrivelse og vurdering) og deretter opprettes det 3 lyttere; den ene lytter om markøren er klikket på, den andre lytter på stenging av informasjonsvinduet, og den tredje setter opp lyttere for å oppdatere forandring i tittel, beskrivelse eller vurdering av den geografiske figuren sammen med en knapp til å slette denne figuren.
- Dersom kategoritypen er område eller sti; så skal det settes opp to lyttere til den flyttbare penn-markøren, den ene mens brukeren holder og dra på pennen for å tegne og den andre når brukeren slipper opp fingeren eller museknappen.
- Mens brukeren holder og drar på museknappen vil det bli hele veien tatt nye punkter og lagret i en midlertidig lokal matrise, og når brukeren slipper opp fingeren eller museknappen så vil et senterpunkt regnes for denne figuren ved hjelp av metoden «**function** computeCentroid(pointsarray)», figuren

tegnes deretter på kartet og informasjonsvinduet åpnes på samme måte som for punktkategorier.



**Figur 13, figuren viser den generelle strukturen til lytteren som gjør det mulig å legge nye figurer i kartet. Merk at «Cursor Drag» og «Cursor Drag End» lytterne opprettes kun når den valgte kategorien er av typen sti- eller område.**



### 7.7.2 Kontrollpanel og hjelpeverktøy

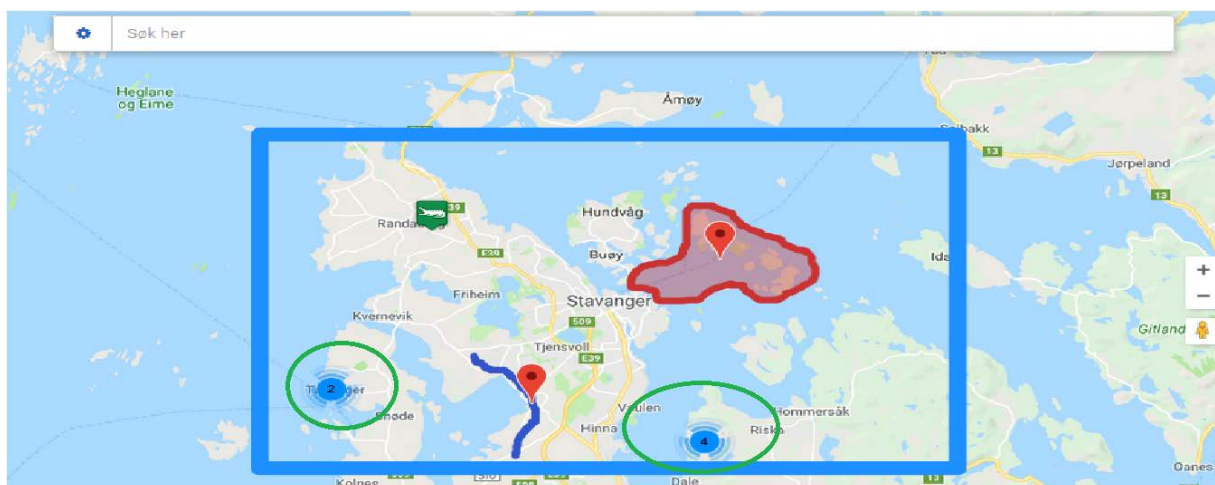
- Variasjon av tjukkelsen til stier: For at brukeren skal kunne ha mer fleksibilitet i applikasjonen er det opprettet en JQuery Funksjon som lytter på en glidebryter og varierer tjukkelsen på linjene som er tegnet på kartet. Lytteren

«`$("input[name~='Opacity']").change(function (e) { ... }));`» vil sørge for å gjøre dette.

- Variasjon av fargeopasitet til områder: For enda mer fleksibilitet, er det også mulig å variere hvor mørk eller lys farge på de ulike områdene ønskes å være ved bruk av en annen glidebryter i kontrollpanelet i det grafiske grensesnittet. «JQuery» funksjonen som tar seg av en slik oppgave er følgende:

«`$("input[name~='Weight']").change(function (e) { ... }));`»

- Gruppering av punkt kategorier: Dersom det er mange punktkategorier som ligger på kartet kan brukeren benytte grupperingsbryteren på kontrollpanelet for å slå av og på grupperingseffekten. Dette vil automatisk regner ut hvor mange punkter er det som ligger nære hverandre og bytter alle de ikonene med en felles ikonbilde som inneholder antall grupperte punkter.



**Figur 14, ikonene for punktkategoriene i de 2 grønne sirklene er gruppert sammen fordi de er plassert tett til hverandre.**

jQuery funksjonen «`$("#clustering_switch").change(function () { ... });`» tar seg av denne funksjonaliteten ved å bruke «Clustering» metoden fra «Google Maps» nemlig «MarkerClusterer». Denne funksjonen bruker kun punktkategoriene som er registrert i «shapes» matrisen.

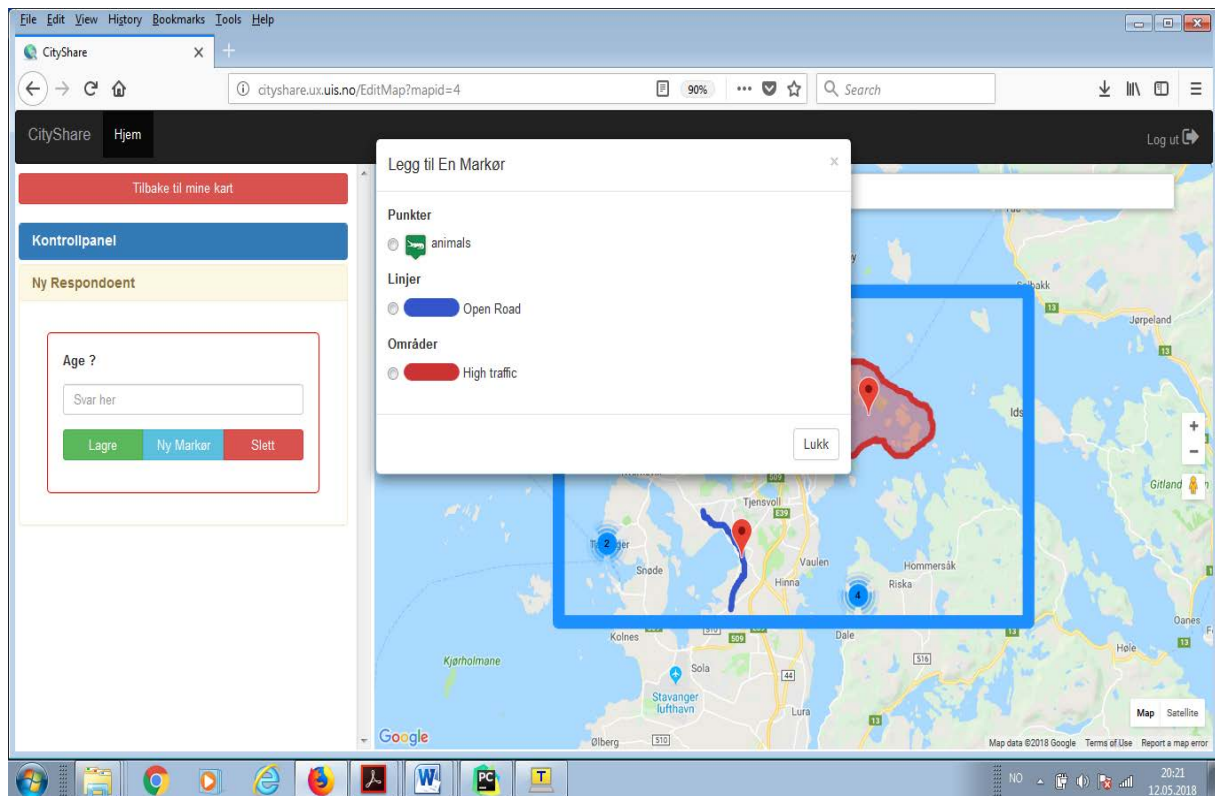
```
var labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';  
markers = [];
```

```
for (var key in shapes) {  
    if (shapes[key]["type"] == "POINT") {  
        markers.push(shapes[key]["marker"]);  
    }  
}
```

*// Add a marker clusterer to manage the markers.*

```
markerCluster = new MarkerClusterer(map, markers,  
    { imagePath:  
    'https://developers.google.com/maps/documentation/javascript/examples/markerclus  
terer/m'  
    });
```

- Svar fra respondenten: Nedenfor kontrollpanelet kan brukeren klikke for å legge til svar på kartets spørsmål til den nye respondenten. Her finner vi et tekstfelt til hvert spørsmål som blir dynamisk generert av «Jinja» og «render\_template» motoren. Sammen med en sletteknapp som sletter alle figurer og svar fra gjeldende respondent derom en feil skjer, lagreknapp og til slutt knappen for å åpne «Bootstrap Modal» som inneholder de definerte figurkategoriene i kartet.

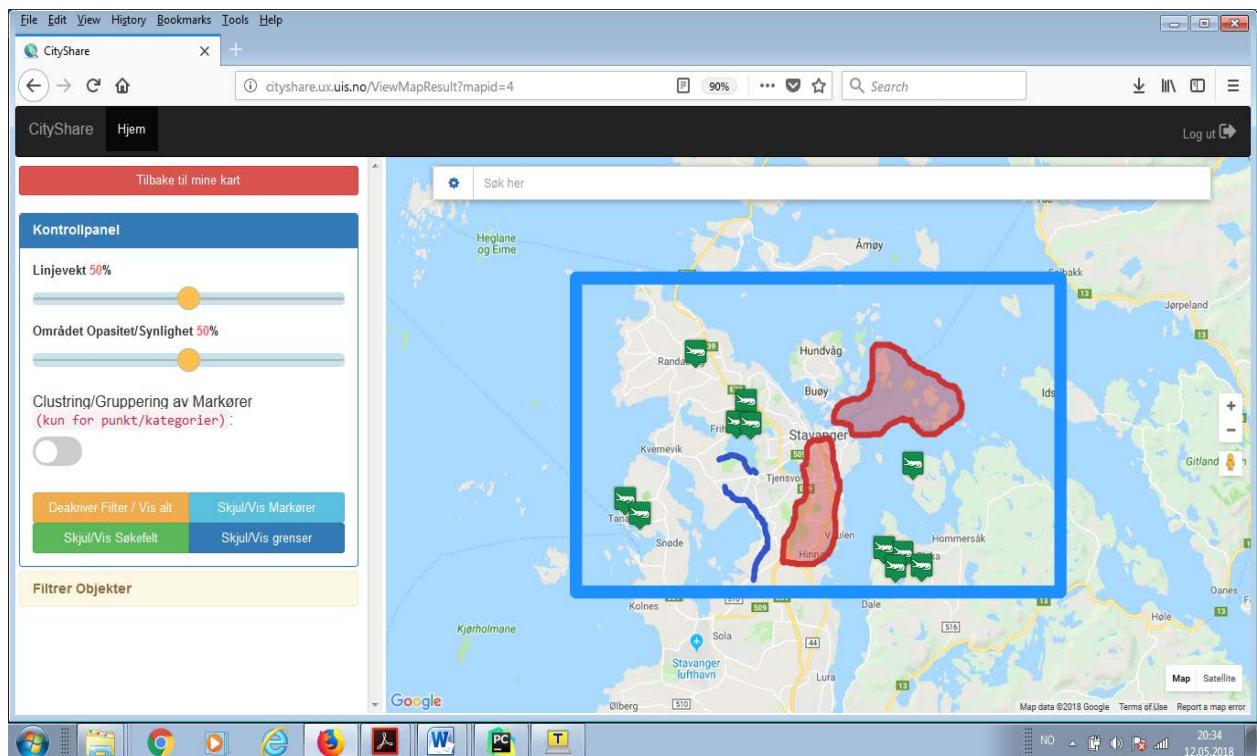


**Figur 15, viser hvordan en bruker kan intervju en ny respondent og velge noen av figurkategoriene for å tegne på kartet.**

Metoden «*ResetResponderentFieldsAndShapes()*» gjennomfører jobben med å slette uønsket data. Den sletter alle figurer i kartet, tømmer «shapes» matrisen og tilslutt tømmer svarfeltene til spørsmålene.

Metoden «*SaveResponderent()*» vil sende inn gjeldene data om respondenten til registrering i databasen. Ved å klikke på Lagre knappen skal en «JQuery Ajax POST» forespørsel bli sendt til følgende URL **"/RegisterResponderent"**. Forespørselen bærer med seg data om respondents- svar og figurer i JSON format. Den også bærer med seg ID-en på dette kartet. Dersom data blir lagt inn i databasen uten feil vil serveren returnerer ordet **"Success"**, hvis noe feil skjer vil responsen være en feil eller autoritet signalverdi som indikerer hva feilen var og en varsel boks skal varsle brukeren om feilen. Se kapittel 6 for mer informasjon om feilhåndteringsmekanismen. Og delkapittel 5.2 for mer om autoritetshåndtering .

## 7.8 Analysering av kart og data tatt fra respondenter i siden «analyze\_map.html»



Figur 16, viser et eksempel av analysesiden "analyze\_map.html".

Etter og mens undersøkelsen pågår skal medlemmer av kartet kunne analysere de figurene og svarene fra respondenter som er registrert på kartet. Denne siden gjør dette mulig.

Her er det viktig å merke at brukere med intervjuerrollen skal kun få se, og kunne påvirke sine egne figurer og brukere med administratorrollen kan se alle figurer lagt inn i kartet. Dette skjer i serveren ved håndteringen av forespørselen som er ansvarlig for utsendelse av denne siden.

Det grafiske grensesnittet til denne siden ligner stort sett på «open\_map.html» med 2 forskjeller.

- I stedet for registrering av nye respondenter i valgpanelet har vi et filter som filtrerer figurer etter ønske.
- I kontrollpanelet har vi en ekstra knapp som skal veksle mellom å skjule/vise markørene til sti- og områdefigurene.

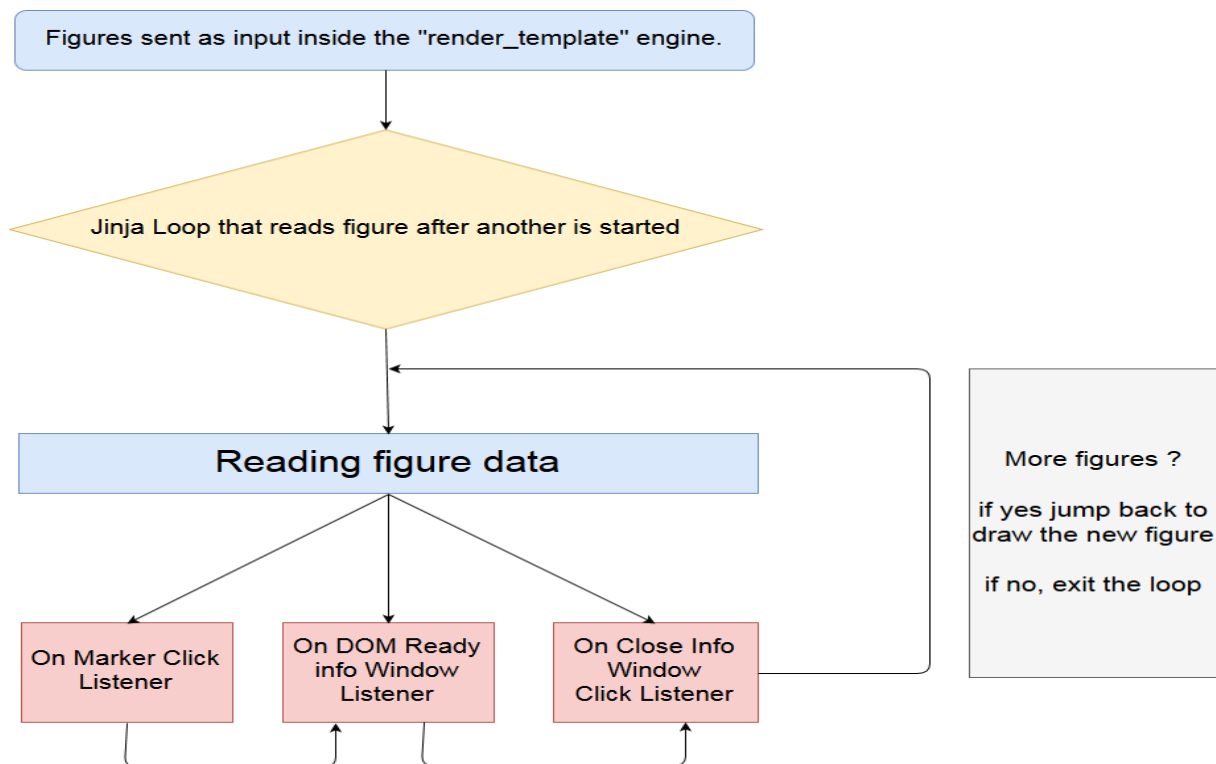
Men først og ved en forespørsel fra en klient til URL-en «[/ViewMapResult](#)» vil både registrerte figurer og svar fra respondenter leses fra databasen og sendes i «render\_template» motoren.

I JavaScript metoden «load\_shapes()» genererer vi ved hjelp av «Jinja» syntaksen en matrise «shapes» som skal inneholde alle figurer og deretter tegne alle figurer på kartet.

Data med «JSON» format kunne også blitt brukt her i stedet for «Jinja» syntaksen, men jeg personlig fikk den ideen mot slutten av prosjektet og hadde dessverre ikke mye tid på meg til å prøve dette.

Prosessen med å tegne og lagre de figurene i metoden «load\_shapes()» i JavaScript ligner stort sett på oppretting av figurer som i «open\_map.html» med et unntak fra alle lyttere som er nødvendige for tegning av figurer. Skissen under viser hvordan «render\_template» motoren håndterer dette.

Metoden «draw\_current\_shapes()» tegner figurene i «current\_shapes» på kartet. Derfor er det viktig å sette «shapes» lik «current\_shapes» ved opplasting av nettsiden.



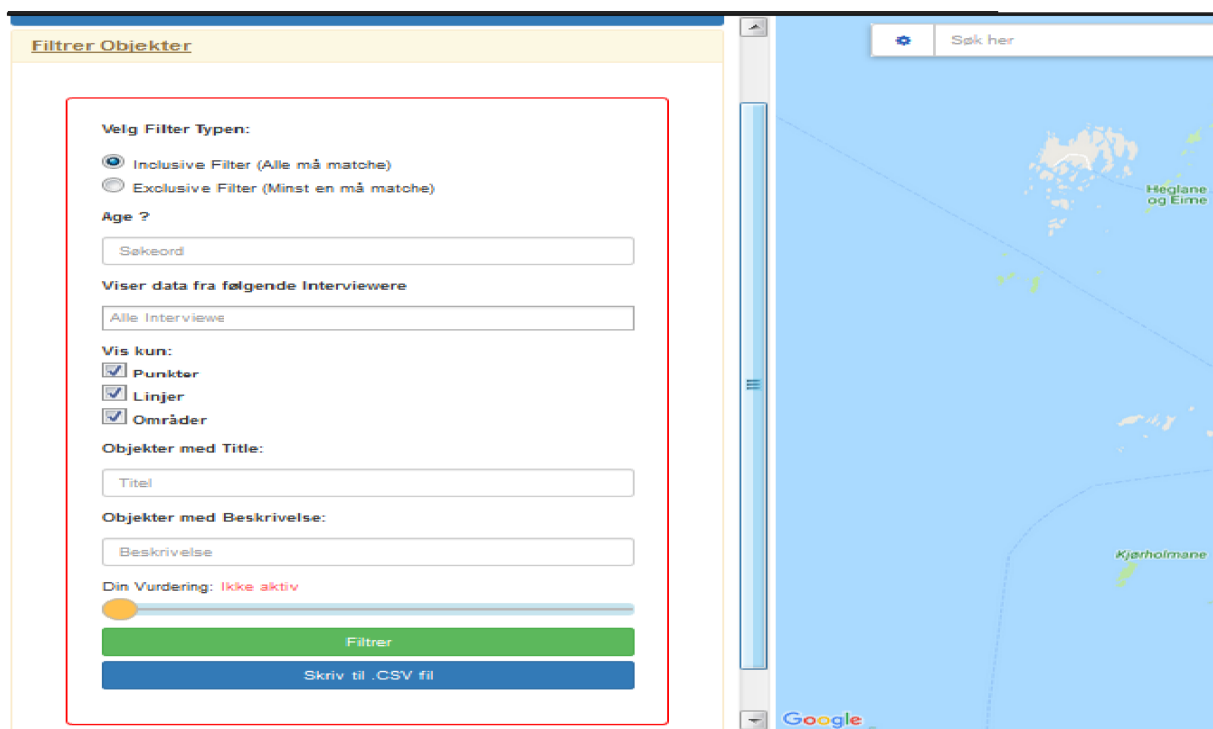
**Figur 17, en skisse som viser den overordnede strukturen til metoden "load\_shapes()" som skal tegne alle de opplastede figurene på kartet.**

Etter at figurene er tegnet kan innhold i tittel, beskrivelse og vurdering forandres ved hjelp av metoden «**function** *Update\_Shape()*» som sender «Ajax POST» forespørsel til følgende URL «**/Edit\_Shape**» for å oppdatere informasjonen i databasen. Samme URL brukes i forespørselen som sletter figuren fra databasen ved hjelp av metoden «**function** *Delete\_Shape()*»

### 7.8.1 Filtrering av data i kartet

Under kontrollpanelet ligger det filteret som inneholder følgende

- HTML «Toggle» Knapper for valg av filtreringsmetode.
- Spørsmålstekst og tekstfelt for svaret for hvert spørsmål kartet inneholder.
- «Select» liste som inneholder navn på alle administratorer og intervjuere.
- 3 «Check» bokser for typen av figuren.
- Tekstfelt for tittel og beskrivelse i figuren.
- «Slider» som skal være en vurdering velger
- En knapp for å starte filtrering
- En knapp for importering av figurer til en fil med «CSV» format.



**Figur 18, et eksempel som viser hvordan det grafiske grensesnittet for filteret kan se ut. «Age?» her er et eksempel på et spørsmål.**

For at en figur skal passe til filtreringskriteriene skal tekst verdier være like eller inkludere den teksten som det søkes for. For andre verdier som «Select» administratorer og intervjuere, typen til figuren eller vurderingen skal verdiene i figuren være akkurat like verdien i søke det søkes for. Da har vi en match.

En variabel matrise «current\_shapes» skal etter filtreringen inneholde kun de figurene som passe til søkekriteriene fra brukeren. Mens «shapes» matrisen alltid skal inneholde alle figurer. «current\_shapes» brukes av metoden «draw\_current\_shapes()» for å tegne figurene på kartet etter filtreringen.

Metoden «Filter\_Shapes();» blir utløst når knappen for filtreringen blir trykket på, metoden gjør følgende:

- Samler opp søkekriteriene fra Filteret i HTML-koden, dersom brukeren ikke skriver noe i noen av de tekstfeltene eller de andre HTML-elementene i filteret, vil ikke disse kriteriene bli tatt med i filtreringsprosessen. Dvs. at filtrering fungerer kun for input som er spesifisert av brukeren.
- Leser fra «Toggle» knappen for filtreringsmetoden om brukeren ønsker å filtrere inklusivt eller eksklusivt.
- Kaller den passende metoden for filtreringen, metodene er forklart etter dette avsnittet.
- Svar (True eller False) om figuren passer til søkekriteriene blir mottatt, og hvis figuren passer, legges den i matrisen «current\_shapes»
- Går videre til sammenligning av neste figur i «shape» matrisen, helt til det ikke er mer figurer igjen å sjekke.

Det er to metoder for filtrering, disse er:

1). Inklusiv Filtrering: Med dette må alle søkekriteriene som brukeren spesifiserer i de forskjellige feltene på kartet passer sammen med data fra figuren for at den figuren skal vises på kartet.

Dette er realisert ved å bruke en «AND» operator som utløser en «True» verdi dersom alle data om figuren stemmer med søket.

Metoden «**function** Compare\_Shape\_inclusive(...）」 tar seg av sammenligningen mellom figurer i matrisen «shapes» og søkekriteriene. Denne metoden returnerer en boolsk verdi



som indikerer om denne figuren er et treff eller ikke, hvis den er det skal denne figuren bli lagt til i matrisen «*current\_shapes*» for deretter å tegne denne figuren på kartet

2). Eksklusiv Filtrering: Med dette er det nok hvis en eller flere av søkekriteriene som brukeren skriver i de forskjellige feltene på kartet passer sammen med data fra figuren for at den figuren skal være en match og dermed vises på kartet.

Dette er realisert ved å bruke «OR» operatoren som utløser en «True» verdi dersom data om figuren stemmer med en eller flere av søkekriteriene i filteret.

Metoden «**function** Compare\_Shape\_exclusive(...)

» tar seg av sammenligningen mellom figurer i matrisen «shapes» og søkekriteriene. Denne metoden returnerer en boolsk verdi som indikerer om denne figuren er et treff eller ikke, hvis den er det så skal denne figuren bli lagt til i matrisen «*current\_shapes*» for deretter å tegne figuren på kartet.

### 7.8.2 Konstruksjon av en «CSV» fil som inneholder informasjon om figurene i kartet.

Som nevnt i oppgavebeskrivelsen i kapittel 2, skal en medlem av kartet kunne laste ned en fil som inneholder informasjon om de geografiske figurene som i det øyeblikket(dvs. eventuelt etter filtrering av figurer) er vist på kartet. Filen skal være av typen «CSV».

Denne filen er ment til å brukes for 3 hovedprinsipper:

- Den filen fungerer som en sikkerhets kopi av informasjon og data om figurer i kartet og de geografiske grensene for kartet. Dette er nyttig dersom databasen eller applikasjonen kommer i ustand eller blir utilgjengelig.
- Den filen kan bli laste opp igjen i andre eksterne programmer som f.eks. programmet «ArcGis». Dette gir analytikerene enda større fleksibilitet i bruk av verktøy og mer redundante lagringsalternativer.
- I framtiden er det også mulig å implementere en funksjon i applikasjonen som leser inn en slik fil og tegne figurene på kartet.

Denne filen inneholder dataattributter om figurene som er vist på kartet når brukeren klikker på nedlastingsknappen, dvs. de geografiske figurene(punkter, stier og områder) etter filtrering eller alle de geografiske figurer ved opplasting av siden

«analyze\_map.html». Dersom filteret ikke viser noe resultat og kartet er tomt for figurer skal brukeren få en JavaScript «alert» varsel som gir beskjeden om at ingen figurer er tilgjengelige til å skrives i filen og filen vil ikke bli nedlastet.

Dataattributtene om figurene er lest fra databasen og er forhåndsbestemt i dialog med veilederne Erlend Tøssebro og Daniela Müller-Eie, disse attributtene er forklart i tabellen under:

Attributt	Forklaring
Map_ID	Kart ID-en fra databasen.
Map_bounds	De geografiske grensene til kartet, i «Well Known Format, POLYGON». For at dette skal fungere på ArcGis programmet er breddegradene «latitude» og lengdegradene «longitude» byttet om i punktene av «WKF».
Area_or_path_Coordinates	De geografiske koordinatene til figuren i «WKF, LINESTRING». Merk: dette inneholder en doble duplikat av senteret for punktfigurer.
Center	Senteret av punkter, stier og områder i «WKF, POINT».
Interviewer	Brukernavnet til intervjueren eller administratoren som har lagt til denne figuren i kartet.
Title	Tittelen av figuren.
Description	Beskrivelsen av figuren.
Rating	Vurderingen av figuren, der 1 er verst og 5 er best.
Category_type	Typen til figuren, «Point, Road, or Area».
Category_ID	Kategori ID fra database, som kan være lik for en eller flere figurer.
Respondent_ID	ID til respondenten fra databasen, en eller flere figurer kan ha same respondent
Spørsmål a?,spørsmål b?...	Alle spørsmål i dette kartet tas med, sammen med

	svarene fra den respondenten som er knyttet til figuren i en rad. Hvis en respondent ikke har svart på dette spørsmålet skal ordet «None» indikerer mangel på svar.
--	---

Når brukeren klikker på knappen for nedlasting av «CSV» filen som er inkludert i Filterboksen i «analyze\_map.html», vil følgende skje:

- JavaScript metoden «**function** *Download\_Map\_AS\_CSV\_File()*» samler opp ID-ene til figurene og tilsvarende Respondenter fra matrisen «current\_shapes» for deretter å sende disse i JSON format til serveren ved bruk av en «Ajax POST» forespørsel til URL-en «[/DownloadMapAsCSVFile](#)».
- Serveren vil her gjennomføre lesing av data fra databasen, konstruere innholdet av filen og deretter sende dette innholdet til klienten som en respons på forespørselen.
- JavaScript metoden «**function** *download*(filename, text)» vil deretter nedlaste filen på klientens enhet (ofte nedlastingsmappen) etter samtykket.

Merk at brukeren ikke kan velge hvor han ønsker å lagre den filen, dette er på grunn av sikkerhetsårsaker, da tilgang til klientens filsystem ved nedlastning anses å være en sikkerhetstrussel til klienten av de fleste nettlesere.

Det er også mulig å lese disse informasjonene direkte fra matrisen «current\_shapes» uten å sende en forespørsel til serveren. Men i dette tilfellet vil man ikke kunne få med seg de oppdateringer av andre administratorer på de forskjellige figurene som gjennomføres samtidig som konstruering av innholdet til filen pågår. Derfor har jeg valgt å gjøre dette på den måten.

ArcGis programmet er det som er mest brukt ved universitet i Stavanger for å gjennomføre geografiske analyser. ArcGis krever at punktene i «WKF» attributtene skal

ha longitude til venstre og latitude til høyre, noe som gjøres på motsatt måte i MySQL. Derfor er disse koordinatene byttet om i «CSV» filen som nedlastes.

En annen ting med ArcGis er at den krever en enkel fil for hver av figurtypene, dvs. en til punkter, en til stier og en til områder. For å gjøre dette hadde veilederen i dette prosjektet, Erlend Tøssebro skrevet et Python skript som skal gjøre dette mulig. Dette skriptet er vedlagt i kapittel 11.3.

Jeg skulle gjerne gjort dette selv og implementert funksjonen i applikasjonen men jeg hadde dessverre ikke mye tid til overs før jeg måtte levere oppgaven.

Vedlagt med denne rapporten i programfiler.zip, finner leseren et eksempel på hvordan den filen ser ut filen har navnet «Example\_CSV\_Figurs\_fil.csv».

### 7.8.3 Kontrollpanel og hjelpeverktøy

Kontrollpanelet ligger i valgpanelet til venstre som vist i figur nummer 16. Den inneholder alle funksjoner som siden «open\_map.html» har, Se kapittel 7.7.2 for mer informasjon om valgpanelet.

Et tillegg her er muligheten til å skjule/vise markørene til områder og stier i kartet. Disse markørene er nødvendige for at informasjonsvinduet for denne figuren skal dukke opp, og blir plassert i senteret av sti- og områdefigurene. Dersom det er mange av disse ved siden av hverandre kan dette bli forvirrende under analysering, derfor er det her lagt en knapp for å skjule og vise de igjen etter behov. JavaScript metoden «[function toggle\\_markers\\_for\\_areas\\_and\\_roads\(\)](#)» tar seg av denne funksjonaliteten.

## 7.9 Visning av svar på spørsmål fra respondenter i siden «respondents\_answers.html»

Age ?	
Svar	Antall
22	4
18	2

Where do you go to school ?	
Svar	Antall
Svar ikke oppgitt	3
UiS	2
UiO	1

What is your nationality ?	
Svar	Antall
Svar ikke oppgitt	3
american	1
Swedish	1
Frenck	1

**Figur 19, viser hvordan svarene på spørsmål tatt fra de forskjellige respondentene i undersøkelsen er vist i siden "respondents\_answers.html".**

Medlemmer av kartet kan se de registrerte svarene på kartetsspørsmål fra respondenter. administratorer kan se alle svar lagt inn av alle administratorer og intervjuere, mens intervjuere kun kan se svar fra respondenter som de selv har intervjuet og sendt inn.

Det er ikke mulig å endre eller slette noen av svarene, da dette kan være truende for troverdigheten av undersøkelsen.

Ved forespørsel fra klienten leses inn relevante svar og spørsmål fra respondenter i dette kartet og data sendes inn i «render\_template» motoren for å generere de forskjellige tabellene for spørsmål og fylle inn svar og antall av dette svaret i tabellen.

Det ferdig genererte HTML-siden «respondents\_answers.html» inneholder en tabell for hvert spørsmål i kartet, denne tabellen inkluderer alle registrerte svar av dette spørsmålet,

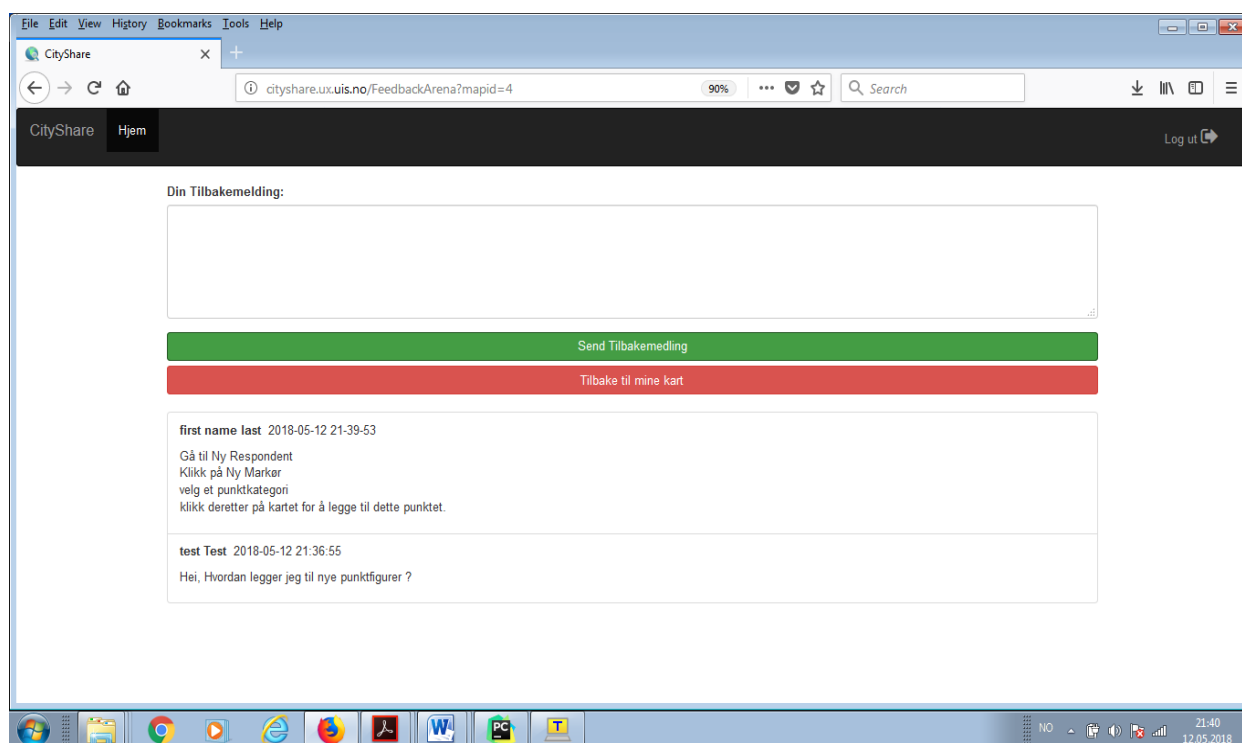
dersom et og samme svar er registrert flere ganger vil dette vises i antall svar til høyre i tabellen.

Hvem av administratorer og intervjuere som legger inn dette er ikke viktig i denne sammenhengen, da målet her er bare å vise alle mulig svar.

For at en intervjuer ikke skal kunne se hva andre brukere har lagt inn av svar fra respondenter og at en administrator skal kunne gjøre dette, gjennomføres en sjekk i databasen og en tilpasset MySQL setning blir sendt til databasen for å motta de aktuelle dataene som klienten har rett til å se.

HTML-Siden «respondents\_answers.html» behøver ikke bruk av JavaScript og tabellene er designet ved hjelp av «Bootstrap» biblioteket.

## 7.10 Tilbakemeldinger og samtale arena i siden «feedback.html»



**Figur 20, siden "feedback.html" gir medlemmer av kartet muligheten til å kommunisere med hverandre.**

HTML-siden «feedback.html» skal være en digital arena der både administratorer og intervjuere kan legge inn sine tilbakemeldinger og spørsmål angående kartet og arbeidet med intervjuingsprosessen. Alle innlagte tilbakemeldinger blir registrert i databasen og er tilgjengelige til å se for alle medlemmer av dette kartet.

Ideen her er å la medlemmer av et kart kommunisere med hverandre, dette vil forsterke arbeidet bak undersøkelsen og gjøre ting lettere for brukeren. Brukeren kan her gi en generell tilbakemelding eller be om hjelp fra andre medlemmer i kartet.

Etter en forespørsel fra klienten leses alle eksisterende meldinger for dette kartet og sendes i «render\_template» motoren for å generere ei «Bootstrap» HTML-liste over meldinger, og deretter sendes disse til klienten.

En sjekk om klienten som har sendt forespørslene er innlogget og registrert som medlem av dette kartet bør gjennomføres for å garantere at uautoriserte brukere skal ikke få tilgang til tilbakemeldinger av medlemmer i dette kartet.

Siden «feedback.html» inneholder på toppen HTML-elementet «textarea» sammen med en knapp for å sende inn meldingen som skrives i «textarea».

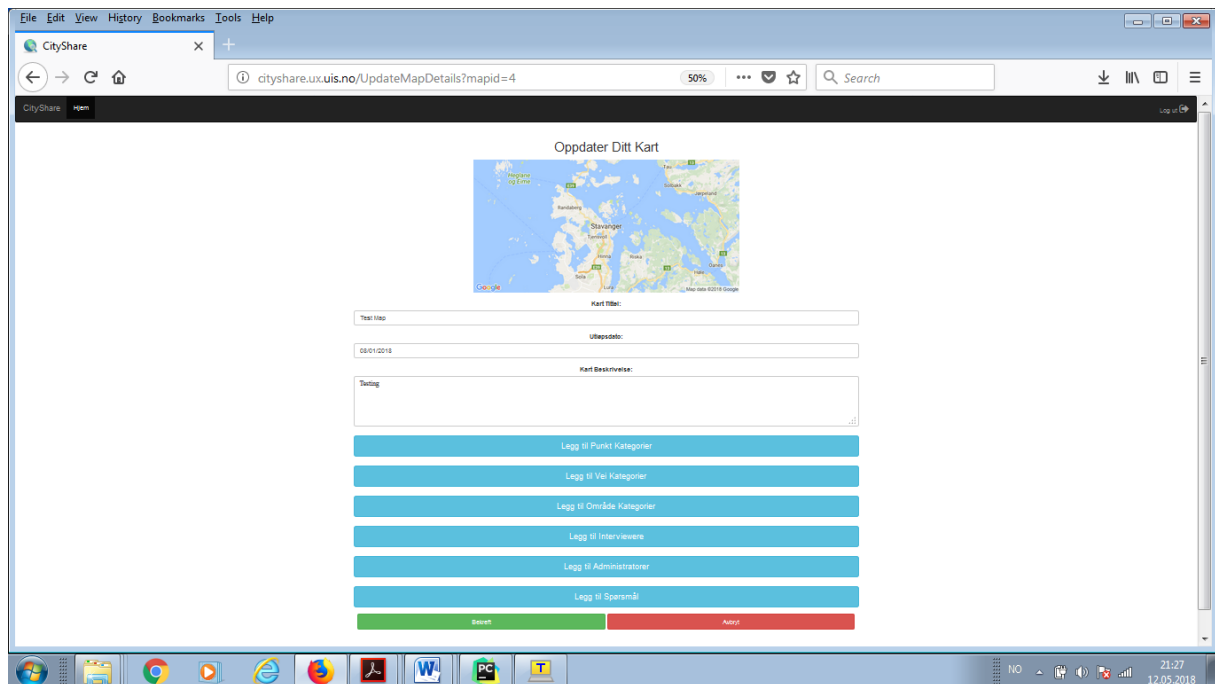
En lytter på denne sendeknappen er definert i «scriptblock» i «feedback.html», lytteren ser slik ut

```
$("#add_feedback_btn").click(function(){...});
```

Her finner vi JavaScript kode som sjekker om «textarea» feltet inneholder en melding, og gir varsel hvis ikke. Metoden vil deretter sende en Ajax «POST» forespørsel til følgende URL «/FeedbackArena» som skal legge inn den nye meldingen i databasen og returnerer fornavnet og etternavnet til brukeren.

Dersom en vellykket respons blir mottatt fra serveren skal meldingene legges inn med de andre meldinger i listen ved hjelp av JavaScript, dette sparer på brukt datamengde av «ISP» da brukeren ikke blir nødt til å nedlaste siden på nytt for å se sin melding i listen.

## 7.11 Innføring av nye kategorier, spørsmål, administratorer og intervjuere i kartet etter opprettelse gjennom siden «update\_map\_details.html».



**Figur 21, et eksempel av hvordan siden "update\_map\_details.html" ser ut. Denne siden og dens funksjonalitet er kun tilgjengelig for eieren av dette kartet.**

Eieren av kartet skal ved et senere tidspunkt etter opprettelsen ha anledningen til å legge til nye kategorier, spørsmål, administratorer og intervjuere til kartet, i tillegg skal eieren endre tittelen og beskrivelsen på kartet, og kanskje utvide eller forkorte gyldighetsdatoen til sitt kart.

De geografiske grensene skal ikke kunne endres, da dette anses å være identisk for kartet, sammen med at det kan forårsake integritetsfeil i databasen. Hvis brukeren ønsker et annet geografisk område for undersøkelsen, har brukeren alltid anledningen til å opprette en ny undersøkelse med et nytt kart. Det er heller ikke mulig å slette nye kategorier, spørsmål, administratorer og intervjuere for samme årsak.



For å gjøre dette mulig brukes siden «update\_map\_details.html» som inkluderer et statisk bilde av kartet levert fra «Google Maps static API», sammen med det skjemaet for oppretting av et nytt kart har av felter og «Modal» bokser.

Gamle kart opplysninger, spørsmål, kategorier, administratorer og intervjuere fylles inn ved hjelp av «Jinja» Syntaks og «render\_template» motoren i «update\_map\_details.html» før den sendes til brukeren.

Mekanismen for utfylling av data i skjemaet ligner stort sett den i filen «static/Scripts/new\_map\_JavaScript\_version\_0.js» for «new\_map.html», men for å holde de to kodene atskilt fra hverandre er den koden for denne funksjonaliteten lagt i følgende fil «static/Scripts/update\_map\_details\_JavaScript\_version\_0.js». Se kapittel 7.5 for mer informasjon om denne koden.

Ved å sende skjemaet inn skal nye kart opplysninger, kategorier og spørsmål legges til kartet i databasen på serversiden. Et varsel vil bli gitt dersom noe input mangler eller en feil oppstår.

## 7.12 Feilmeldingssiden «error.html»

Denne siden blir vist til brukeren dersom en feil skjer ved «GET» forespørslene.

Siden bruker «Bootstrap» varsal av typen «danger» og inneholder en feilmelding sammen med en link som sender brukeren til hjemmesiden. For mer informasjon om feil- og autoritetshåndtering kan du se i kapittel 5 og 6.

Kjernekode for denne siden er skrevet med «Jinja» Syntaks og sjekker signalverdiene for det som har gått galt på den måten:

```
{% block content %}
<div id="errdiv" class="alert alert-danger">

  <h1>Obs! En feil har skjedd!</h1>

  {% if Fail_Code == 1 %}
    <strong>Database Feil</strong>
```

```

{% elif Fail_Code == 2 % }
    <strong>Du har ikke rettighet til å gjøre dette</strong>
{% elif Fail_Code == 3 % }
    <strong>Dette Kartet er gått ut på dato</strong>
{% elif Fail_Code == 4 % }
    <strong>Du må være innlogget for å få tilgang</strong>
{% else % }
    <strong>Ukjent Feil</strong>
{% endif % }

<a href="/">Gå tilbake til hjemmesiden</a>

</div>

```

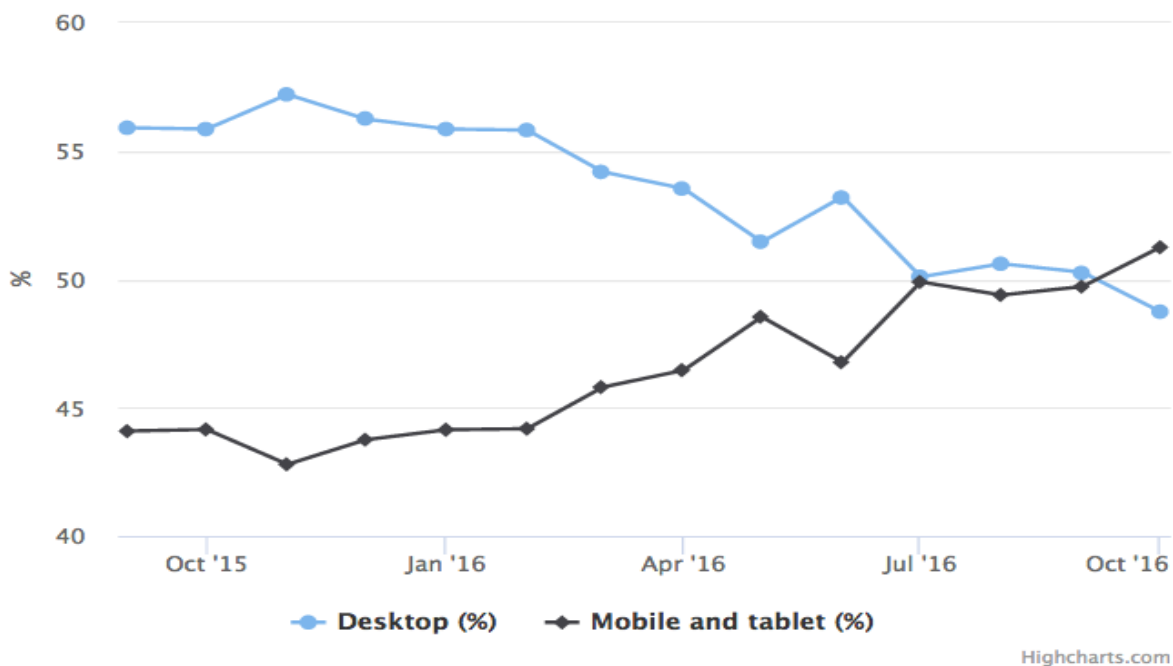
```

{% endblock % }

```

## Kapittel 8: Tilpasningsdyktig Skjerm design «Screen Responsive Design»

Tilgjengeligheten og bruken av mobiltelefoner og Nettbrettenheter opplever en sterk økning i den siste tiden, samtidig som bruken av stasjonære enheter avtar.



**Figur 22, En sammenligning mellom bruk av stasjonære datamaskiner og bruk av Mobile enheter. (Highcharts.com, 2017).**

For den årsaken kreves det at moderne nettsider skal kunne tilpasse så mange enheter som mulig av alle typer skjermdimensjoner og opplysninger.

En tilpasningsdyktig skjerm design justerer innholdet av websidene relativt til den tilgjengelige vindustørrelsen på brukerens enhet. En slik design vil forbedre måten innholdet av HTML-dokumentene blir vist på og gjør brukeropplevelsen av det grafiske grensesnittet mye lettere.

Etter testing med ulike skjermer og parametere har jeg kommet fram til at valgpanelet skal vises på fult skjerm (100% av bredden) i enheter som har mindre enn 800px i breddestørrelse og på omtrent 30% av denne størrelsen hvis breddestørrelsen er mer enn 800px.

For å implementere dette brukes det følgende JavaScript metode «**function** *auto\_adjust\_dimentions()*» som realiserer dette etter den tilgjengelige breddestørrelsen til vinduet.

En lytter til vinduet blir lagt til for å kalle inn metoden «**function** *auto\_adjust\_dimentions()*» dersom dimensjonene til vinduet endrer på seg. Denne lytteren er vist under.

```
auto_adjust_dimentions();
$(window).resize(function () {
    auto_adjust_dimentions();
});
```

Merk at lytterfunksjonen vil ikke bli utløst ved første opplasting av den siden, derfor er det viktig med et enkelt kalt på metoden «**function** *auto\_adjust\_dimentions()*» først.

En variable «**var** *choice\_panel\_toggled*» holder styr på om valgpanelet er skjult eller ikke, denne variabelen oppdateres jevnlig av den lytteren som er nevnt ovenfor, og av en lytter på knappen som veksler mellom skjult og vist valgpanel, denne lytteren har følgende deklarasjon:

```
$("#toggle_panel").click(function () { ... });
```

Og sørger for vise valgpanelet på fult skjerm dersom enheten er mindre enn 800px i bredestørrelse.

Implementeringskoden ovenfor er identisk i alle de 3 JavaScript filene og tilsvarende HTML-dokumenter. Men jeg velger uansett å ha hver i tilsvarende JavaScript fil, bare for ikke å miste oversikt dersom noe uforventet skjer. Kodene finnes i disse filene:

- ..\CityShare\_Flask\static\Scripts\new\_map\_JavaScript\_version\_0.js
- ..\CityShare\_Flask\static\Scripts\analyze\_map\_JavaScript\_version\_0.js
- ..\CityShare\_Flask\static\Scripts\open\_map\_JavaScript\_version\_0.js

For andre HTML-dokumenter holder dette med kun CSS kode eller ved bruk av «Bootstrap Grid System». Ettersom alle HTML-dokumenter arver fra dokumentet «index.html» kan vi plassere koden vist nedenfor i hodet av «index.html» for at alle sider skal kunne få dette med.

```
<meta name="viewport" content="width=device-width">
```

Denne vil sørge til å sette bredestørrelsen til scenen lik den for enheten. Dette er nødvendig for at CSS koden for tilpasning av skjermen skal fungere.

I dokumentet «home.html» ser vi f.eks. at de forskjellige boksene vil ha bredden lik 30% av skjermen hvis skjermen er større enn 749px, og her brukes det følgende CSS kode:

```
.box {  
    display: inline-block;  
    float: left;  
    border: 1px solid grey;  
    box-shadow: 5px 5px #d3d3d3;  
    padding: 5%;  
    margin: 5px;  
    width: 30%;  
    text-align: center;
```

```
vertical-align: middle;
border-radius: 50px 20px;
}
```

Hvis skjermen blir mindre enn 749px vil følgende kode dynamisk bytte breddestørrelsen til 90% av skjermen. Disse parameterne er også bestemt ved testing.

```
@media only screen and (max-width: 800px) {
  .box {
    width: 90%;
  }
}
```

I denne løsningen bruker jeg antall piksler for å løse dette problemet, men det finnes sikkert noen enheter med skjermtypen som Apple retina som er på 13,3 tommer størrelse med en oppløsning på 2560x1600 piksler og Samsung Galaxy S9 smarttelefon som har 5,8 tommer skjerm med oppløsning 2960x1440 piksler.

Denne løsningen kan forbedres i framtiden ved å bruke en annen teknikk enn direkte piksel tall dersom problemer oppstår som følge av uordnet grafisk grensesnitt. Men for nå og under testingen med flere enheter er det i hvert fall sikkert at dette vil virke.

Applikasjonen er forventet å brukes med nettbrett av typen Apple og Samsung samt stasjonære datamaskiner. Derfor er det gjennomført en rekke vellykkete tester på følgende enheter:

- Apple iPad 4 med 9.7-tommers Retina Skjerm.
- Apple iPad Pro med 12.9-tommers Retina Skjerm.
- Mac Pro laptop med 13-tommers skjerm.
- Lenovo Yoga 2 Pro laptop.
- HP EliteBook laptop.
- Apple iPhone 5C.
- Apple iPhone 6.
- Samsung Galaxy S II.

## Kapittel 9: Mobilapplikasjon kompatibel med operativsystemene, Android og iOS

### 9.1 Om Web ramme «engelsek. Webframe» applikasjoner.

En web ramme applikasjon er en helt vanlig applikasjon som lastes på brukerens enhet og viser en nettside gjennom applikasjonens vindu.

På den måten trenger ikke brukeren av nettsiden å være avhengige av en nettleser for å få tilgang til nettsiden, i stedet kan brukeren benytte applikasjonen på sin enhet for å få direkte tilgang til nettsiden. Brukeren trenger også ikke å skrive inn URL-adressen på nettsiden for å få tilgang til den. En slik applikasjon kan godt ses på som en nettleser applikasjon dedikert til kun en nettside, og i dette tilfellet er nettsiden «www.cityshare.ux.uis.no»

### 9.2 Fordeler og ulemper med en slik type applikasjon.

Denne type applikasjoner har følgende fordeler:

- Lett å implementere og teste:

Dette er fordi kjernen av applikasjonen er en importert nettside, mens applikasjonen i seg selv er ikke mer enn bare en nettleser. Derfor trengs det mye mindre koding for å lage applikasjonen enn ved å produsere en vanlig applikasjon.

- Mer sikker enn vanlig applikasjon:

Dette er fordi kommunikasjon med databasen gjennomføres på serversiden, og brukeren kommuniserer med serveren gjennom http eller https forespørsler og mottar responser.

Å kommunisere med databasen i en vanlig applikasjon krever at en forbindelse til databasen skal opprettes ved bruk av hemmelige parametere som adresse, passord og navn på databasen, med såkalt «reverse engineering» kan en programmerer se på koden til applikasjonen og dermed vil han/hun kunne få tak i de parameterne som gir tilgang til

databasen over internett. Dette er svært kritisk da data i databasen kan bli manipulert eller stjålet som følge av dette.

- Lett å oppdatere:

Dersom applikasjonen skal oppdateres til f.eks. bedre design eller andre funksjonaliteter, så holder det bare å oppdatere nettsidens kjerneapplikasjon (i dette tilfellet menes prosjektet «CityShare\_Flask»), og da vil alle klienter få samme innhold uansett om de bruker nettleser eller en web ramme applikasjon.

- Applikasjonens størrelse blir veldig liten i forhold til vanlige applikasjoner.

Dette er fordi det kreves lite lagringsplass på enheten til brukeren for stil ark «Stylesheets» og andre filer for koding av applikasjonens logikk.

Følgende ulemper er også verdt å nevne:

- web ramme applikasjon øker antall forespørsler til serveren, noe som kan være en belastning på serversiden for applikasjoner med allerede mange brukere i websiden, fordi serveren kan krasje ned som følge av alt for mange forespørsler sendt fra klienter.

En applikasjon som ikke bruker web rammer kan som nevnt kommunisere direkte med databasen i serveren, og behøver ikke bruk av http forespørsler.

En annen løsning kan være å ha to forskjellige servere, en til den vanlige applikasjonen og en til nettsiden. Begge serverne kan deretter kommunisere med databasen. Men dette krever nemlig mer ressurser.

- En annen ulempe er at de fleste mobile enheter i dag har forskjellige skjermdimensjoner og varierer mye i antall piksler. Derfor må innholdet av nettsiden tilpasse seg brukerens enhetsskjerm. Noe som krever en tilpasningsdyktig design i applikasjonen. For å løse dette er denne applikasjonen designet for å tilpasse seg skjermen som viser applikasjonen, mer om denne løsningen er redegjort i kapittel 8.

### 9.3 Refleksjon og avgjørelse angående en slik applikasjon.

Instituttet for industriell økonomi, risikostyring og planlegging ved universitet i Stavanger ønsket seg en nettsideapplikasjon som også skal være tilgjengelig for operativsystemene «iOS» og «Android», og ikke bare ved bruk av en nettleser.

Den beste løsningen til å få dette til vil være en kombinasjon av web rammer og vanlig programmering i tilsvarende plattform til operativsystemet. En slik applikasjon bruker HTTP eller HTTPS forespørsler for å få inn data fra serveren i f.eks. «JSON» format og deretter håndtere disse i applikasjonen. Slik vil applikasjonen kun være avhengige av serveren til nettsiden for å kommunisere med databasen, men uavhengige av nettsidens design og endringer.

Et annet alternativ er å bruke verktøyet «**Xamarin**» som gir programmere anledningen til å skrive kode for mange plattformer som «Android», «Windows» og «iOS» på en gang. Men her må ofte noe av koden allikevel bli skrevet i det morsmålsspråket til de forskjellige operativsystemene, dvs. «Swift» for «iOS» og «Java» for «Android»

Løsningene med vanlig applikasjon og «Xamarin» krever mer arbeid, da mer koding er krevd for designing av applikasjonen. For i denne applikasjonen forventes ikke antall klienter å være høy, og første prioritet er selve nettsiden. Da er det nok best på dette tidspunktet å lage en web ramme applikasjon på de to mest brukte operativsystemene «iOS» og Android. Implementering av denne koden er forklart i neste to delkapitler

### 9.4 Implementering av web ramme applikasjon i operativsystemet «iOS»

Prosjektmappen «City Data iOS» inneholder filene for denne web ramme applikasjonen, Du kan bruke verktøyprogrammet «XCode» for å åpne dette prosjektet.

Filene med følgende sti i prosjektet «\City Data iOS\City Data iOS\ViewController.swift» er der koden for web rammen skal stå, denne koden er også vist her.



```
import UIKit

class ViewController: UIViewController {
    // Refers to the web frame element from Main.storyboard
    @IBOutlet weak var webView: UIWebView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let url = URL(string: "http://cityshare.ux.uis.no/")
        let urlRequest = URLRequest(url: url!)
        webView.loadRequest(urlRequest)
    }
}
```

Her ser vi at URL for nettsiden vår er skrevet inn til variabelen «url» og en forespørsel er sendt til serveren.

«iOS» operativsystemet vil automatisk blokkere tilgang til ikke sikre nettsider som bruker nettverksprotokollen «HTTP», noe vår applikasjon gjør i utviklings- og testfasen. Derfor bør det plasseres et unntak av vår nettside i filen «City Data iOS\City Data iOS\Info.plist» på den måten.

```
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <false/>
    <key>NSExceptionDomains</key>
    <dict>
        <key>cityshare.ux.uis.no</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
        </dict>
    </dict>
</dict>
```

```

    <key>NSTemporaryExceptionAllowsInsecureHTTPLoads</key>
    <true/>
    <key>NSTemporaryExceptionMinimumTLSVersion</key>
    <string>TLSv1.1</string>
  </dict>
</dict>
</dict>

```

Denne applikasjonen er lastet ned lokalt ved bruk av programmet «XCode» og testet på 2 enhet av typen «iPhone 5C» og «iPhone 6», testen var vellykket. Men applikasjonen er ennå ikke publisert i «AppStore» da dette krever en betalt konto. Men den er i hvert fall klar til publisering når som helst.

## 9.5 Implementering av web ramme applikasjon i operativsystemet «Android»

Prosjektmappen «CityDataAndroid» inneholder filene for denne web ramme applikasjonen. Du kan bruke verktøyprogrammet «Android Studio» for å åpne dette prosjektet.

Først må applikasjonen få tillatelse til å bruke Internett gjennom enheten, følgende kode i filen «\CityDataAndroid\app\src\main\ AndroidManifest.xml» gjør dette mulig.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Deretter implementeres koden for web rammen i filen

«\CityDataAndroid\app\src\main\java\no\uis\citydataandroid\MainActivity.java» som vist under

```
package no.uis.citydataandroid;
```

```

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {

    private WebView Web_View;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Capturing the Web View element from the XML sheet file, activity_mail.xml.

        Web_View = (WebView) findViewById(R.id.WebView);

        // This line is to enable loading the website inside the app and not in the default
        // external browser of the users device."

        Web_View.setWebViewClient(new WebViewClient());

        // Specifies which website will be loaded in the web view element box.

        Web_View.loadUrl("http://cityshare.ux.uis.no/");

        // enables JavaScript

        WebSettings webSettings = Web_View.getSettings();
        webSettings.setJavaScriptEnabled(true);

```

```

}

// event that listens to the back button on the user android device
@Override
public void onBackPressed() {

    // if the history includes at least one page, then go to the last visited page

    if (Web_View.canGoBack()) {
        Web_View.goBack();

        // if not, then close/hide the App
    } else {
        super.onBackPressed();
    }
}
}

```

Her ser vi at URL-en for nettsiden vår er skrevet inn til «loadUrl()» metoden og en forespørsel er sendt til serveren.

Siden «Android» enheter har en knapp for å navigere tilbake en side, har vi lytteren «onBackPressed()» som sørger for at dette skal gå uten problemer.

Denne applikasjonen er lastet ned lokalt ved bruk av programmet «Android Studio» og testet på en enhet av typen Samsung Galaxy S II, testen var vellykket. Men det har oppstått et problem med valgpanelet; Når brukeren prøver å taste inn data, forsvinner disse og brukeren blir sendt tilbake til kartet. Jeg rakk dessverre ikke å implementere en løsning på dette. Da det viktigste var å få applikasjonen klar fra en nettleser og helst fra en web ramme applikasjon kompatibel med «Apple iOS» operativsystemet.

Applikasjonen er ennå ikke publisert i «GooglePlayStore» da dette kanskje krever en betalt konto.

## **Kapittel 10: Tilbakemeldinger på applikasjonen og Refleksjon**

### **10.1 Testing og Tilbakemeldinger på applikasjonen.**

Når utviklingen av applikasjonen var fullført ble den testet av veilederen Erlend Tøssebro og doktorgradsstudent Ana Llopis Alvarez. Tilbakemeldingene var i det meste positive og applikasjonen ble godkjent og sett på som meget tilfredsstillende til Institutt for industriell økonomi, risikostyring og planlegging.

For å dokumentere dette ba jeg Ana Llopis Alvarez om å skrive en tilbakemelding som jeg inkluderer her i kapittel 11.4.1.

I tillegg til dette har jeg fått tilbakemelding fra Dario Garigliotti som er doktorgradskandidat i Institutt for data- og elektroteknologi (IDE) ved TN-fakultetet, ved Universitet i Stavanger. Tilbakemeldingen fra Dario er tilgjengelig i kapittel 11.4.2

Dario mente at det grafiske grensesnittet var lett å forstå og bruke. Han ga meg flere råd om oppdateringer av applikasjonen, disse er:

- Gjør det mulig for eieren av et kart å legge til nye kategorier til kartet ved et senere tidspunkt.

Jeg tok rådet og fikk implementert dette i applikasjonen. Se kapittel 7.11 for mer informasjon om dette.

- Gi brukeren anledningene til å definere noen bestemte mulige svar på spørsmål fra kartet som kan aksepteres fra respondentene under undersøkelsene. Et eksempel på dette er en liste med land for spørsmålet «Hvor er du fra?» i stedet for å gi intervjueren muligheten til å skrive hva som helst som svar på spørsmål.

Jeg syns at dette vil gjør det vanskelig for intervjueren å registrere svar fra respondenter, da disse respondentene kan komme med andre svar enn det som er tilgjengelig for utfylling. Men dette er faktisk en god ide i noen tilfeller og bør vurderes som en oppdatering av nettsiden i framtiden.

- Gi brukeren anledning til å bestemme typen av HTML-feltene som er dedikert for å registrere svar fra respondenter om spørsmål fra kartet, et eksempel på dette er en radioknapp gruppe for spørsmålet om kjønn av respondenten, og hvis andre svar skal komme her da kan brukeren heller velge å skrive fri tekst i en vanlig HTML «input» tekstfelt.

Dette er en kjempe bra ide, men jeg rakk dessverre ikke å få dette fullført på grunn av begrenset tid fra jeg fikk tilbakemeldingene til prosjektet måtte leveres. Men ideen bør absolutt vurderes i framtiden.

- Vurderingsskala kan inneholde et ord som viser hva vurderingen er, et eksempel er «veldig dårlig» for vurdering med 1 poeng og «veldig bra» for vurdering med 5 poeng.

Jeg var enige med Dario, og implementerte dette i applikasjonen.

- Dario likte grupperingsfunksjonen av punkt kategorier, og lurte om dette kan bli brukt for områder og stier.

Det er fullt mulig å implementere gruppering av områder og stier, men jeg og veilederne ser ikke for oss at det vil bli brukt. Så jeg velger å la være å gjøre dette.

- Dario spurte om det går an å gruppere punkter i en sirkel med bestemt radiusavstand på kartet som brukeren spesifiserer.

Jeg syns dette er en kjempe god ide, men har dessverre ikke mye tid til å kunne implementere dette, derfor bør dette kanskje tas med i en eventuell oppdatering i framtiden.

## 10.2 Refleksjon og konklusjon

Dette prosjektet har en stor potensiell til videreutvikling i framtiden. De nevnte oppdateringene i kapittel 10.1 er noe som bør vurderes for å produsere en bedre applikasjon i framtiden.

Denne nettsiden bruker nettverksprotokollen «HTTP», denne protokollen tilbyr ikke kryptering av innholdet i forespørselen mellom serversiden og klientene. Når applikasjonen skal tas i bruk er det derfor viktig å bruke nettverksprotokollen «HTTPS» som tilbyr kryptering. IT-avdelingen ved Universitet i Stavanger kan støtte med dette.

Arbeidet med denne applikasjonen var både utfordrende og spennende. Gjennom prosjektet fikk jeg økt mine kunnskaper på mange områder og fagfelt. Min kommunikasjonsevne med andre fagfolk fra annet fagfelt er også forbedret og forsterket.

Jeg vil herved takke Erlend Tøssebro, Daniela Müller-Eie, Dario Garigliotti, og Ana Llopis Alvarez. Samarbeidet mellom oss har vært veldig avgjørende for å gjennomføre dette prosjektet.

Planen var å arbeide med dette prosjektet sammen med en annen student, men denne studenten hadde ikke tatt faget «Web programmering» som utgjør grunnlaget i utviklingene av denne applikasjonen. På grunn av dette endte jeg opp til å gjøre nesten hele prosjektet selv. Denne studenten valgte på et senere tidspunkt å levere sin egen rapport. Noe jeg ikke var enig i, da vi burde holde oss sammen som et lag og levere samme rapport uansett om hvem som har gjort hva. Men jeg håper at de ser innsatsen som vi begge har lagt inn i prosjektet og takker for oss.

## Kapittel 11: Vedlegg

### 11.1 Framgangsmåte for publisering av nettsiden ved UNIX anlegget i Universitet i Stavanger.

I dette delkapitlet skal jeg gå gjennom framgangsmåten for publisering av nettstedet ved å bruke anlegget ved Universitet i Stavanger. Og som kjører operativsystemet UNIX.

Vennligst følg tålmodig stegene under for en enkel publisering.

- Kontakt den IT-ansvarlige i din avdeling og be han/hun om følgende
  - o Opprette en UNIX brukernavn og passord for deg  
Example: brukernavn@badne7.ux.uis.no.
  - o Dersom du ønsker å endre database, be også om opprettelse av en ny MySQL database versjon 5.7.
  - o Opprette en «DNS» domene for ditt nettsted med ønsket navn, og åpne tilgang til den for hele det globale internett nettverket.
- Hvis du bruker Operativsystemet «Windows», last ned «TeraTerm» og åpne den.
- Hvis du er en Mac-bruker, åpne programmet «Terminal».
- Bruk scp kommandoen, for å kopiere prosjektmappen «CityShare\_Flask» til ditt UNIX konto. Merk at «scp» fungerer kun på Mac iOS. For «Windows» anbefales det å ta kontakt med din IT-avdeling, eller vurder bruk av GitHub.com og «git clone» kommandoen.

Eksempel med «scp» kommandoen:

```
scp -r ../CityShare_Flask dittbrukernavn@badne7.ux.uis.no:Bachelor_Oppgaven
```

- Etabler en «SSH» forbindelse til Unix anlegget ved UiS ved å bruke ditt Unix brukernavn og passord.

- Bruk deretter følgende kommandoer:

```
tmux
```



- Naviger til mappen CityShare\_Flask:

```
cd Bachelor_Oppgaven/CityShare_Flask/
```

- Opprett en virtuell miljø, aktiver den og last ned «flaskpakken» og «mysql-connector»

```
virtualenv env
```

```
. env/bin/activate
```

```
pip install flask
```

```
pip install mysql-connector
```

- Kjør nå serveren, ved å bruke:

```
python CityShare.py
```

- Tast inn **Ctrl + b + d**, for å gå ut av «tmux».

- Serveren er nå oppe. Test dette ved å gå inn i linken «CityShare.ux.uis.no» fra en nettleser.

- Avslutt «SSH» forbindelsen ved å bruke kommandoen (**exit**)

- Dersom du vil senere gå inn i «tmux» igjen. Gjør følgende

Bestem hvilken «tmux» du ønsker å gå inn i ved å bruke kommandoen (**tmux ls**)

Gå inn i «tmux» ved å bruke kommandoen (**tmux a -t 0**), der ( 0 ) er nummeret til ønsket «tmux»

- Du kan avslutte Python skriptet og dermed ta serveren ned ved å bruke (**Ctrl + c**)

## 11.2 Database

### 11.2.1 Skjema for opprettelse av databasen.

```
CREATE SCHEMA dbmguniem;
```

```
USE dbmguniem;
```

```
CREATE TABLE Persons (
```

```
username VARCHAR(20),
```

```
login_pass VARCHAR(20) NOT NULL,
```

```
first_name VARCHAR(20) NOT NULL,
```

```
last_name VARCHAR(20) NOT NULL,
```

```
e_post VARCHAR(30) NOT NULL,
```

```
telephone LONG NOT NULL,
```

```
PRIMARY KEY(username),
```

```
INDEX(login_pass),
```

```
INDEX(e_post)
```

```
);
```

```
CREATE TABLE Maps (
```

```
map_id INT auto_increment,
```

```
map_creator VARCHAR(20),
```

```
title VARCHAR(40) NOT NULL,
```

```
description TEXT NOT NULL,
```

```
start_date TIMESTAMP DEFAULT NOW(),  
  
end_date DATE NOT NULL,  
  
geo_boundery POLYGON NOT NULL,  
  
zoom INT NOT NULL,  
  
PRIMARY KEY(map_id),  
  
FOREIGN KEY (map_creator) REFERENCES Persons(username)  
  
);
```

```
CREATE TABLE Maps_Interviewers(  
  
username VARCHAR(20),  
  
map_id INT,  
  
PRIMARY KEY (username, map_id),  
  
FOREIGN KEY (username) REFERENCES Persons(username),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id)  
  
);
```

```
CREATE TABLE Maps_Administrators(  
  
username VARCHAR(20),  
  
map_id INT,  
  
PRIMARY KEY (username, map_id),  
  
FOREIGN KEY (username) REFERENCES Persons(username),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id)
```

);

```
CREATE TABLE Maps_Categories(  
  
category_ID INT AUTO_INCREMENT,  
  
category_name VARCHAR(30) NOT NULL,  
  
category_type VARCHAR(30) NOT NULL,  
  
category_image_or_color VARCHAR(30) NOT NULL,  
  
map_id INT,  
  
PRIMARY KEY (category_ID),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id)  
  
);
```

```
CREATE TABLE Maps_Questions(  
  
question_ID INT AUTO_INCREMENT,  
  
question TEXT NOT NULL,  
  
map_id INT,  
  
PRIMARY KEY (question_ID, map_id),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id)  
  
);
```

```
CREATE TABLE Maps_Respondents(  
  
respondent_ID INT AUTO_INCREMENT,
```

```
map_id INT NOT NULL,  
  
username VARCHAR(20) NOT NULL,  
  
PRIMARY KEY (respondent_ID, map_id),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id),  
  
FOREIGN KEY (username) REFERENCES Persons(username)  
  
);
```

```
CREATE TABLE Respondent_Answers(  
  
respondent_ID INT NOT NULL,  
  
question_ID INT NOT NULL,  
  
Answer TEXT,  
  
PRIMARY KEY(respondent_ID, question_ID),  
  
FOREIGN KEY (respondent_ID) REFERENCES Maps_Respondents(respondent_ID),  
  
FOREIGN KEY (question_ID) REFERENCES Maps_Questions(question_ID)  
  
);
```

```
CREATE TABLE Shapes (  
  
shape_id INT AUTO_INCREMENT,  
  
category_ID INT,  
  
shape_creator VARCHAR(20),  
  
center POINT NOT NULL,  
  
area_or_path LINESTRING,
```

```
title VARCHAR(40),  
  
description TEXT,  
  
rate INT DEFAULT 3,  
  
respondent_ID INT,  
  
PRIMARY KEY(shape_id),  
  
FOREIGN KEY (category_ID) REFERENCES Maps_Categories(category_ID),  
  
FOREIGN KEY (shape_creator) REFERENCES Persons(username),  
  
FOREIGN KEY (respondent_ID) REFERENCES Maps_Respondents(respondent_ID)  
  
);
```

```
CREATE TABLE Feedbacks (  
  
feedback_id INT AUTO_INCREMENT,  
  
map_id INT,  
  
username VARCHAR(20),  
  
issue_timestamp TIMESTAMP DEFAULT now(),  
  
message TEXT NOT NULL,  
  
PRIMARY KEY (feedback_id, map_id),  
  
FOREIGN KEY (username) REFERENCES Persons(username),  
  
FOREIGN KEY (map_id) REFERENCES Maps(map_id)  
  
);
```

### 11.2.2 De brukte MySQL Spørringene i applikasjonen.

<\*\*\*\*\* Select Queries \*\*\*\*\*>

```
SELECT count(username) FROM Persons
WHERE BINARY username = aes_encrypt(%s,'enckey') AND BINARY login_pass =
aes_encrypt(%s,'enckey') LIMIT 0, 1;
```

```
SELECT cast(aes_decrypt(login_pass, 'enckey') AS CHAR(20)), first_name, last_name,
e_post, telephone
FROM Persons WHERE username = aes_encrypt('Username','enckey')
```

```
SELECT cast(aes_decrypt(username, 'enckey') AS CHAR(20))
FROM Persons where username = aes_encrypt('Username','enckey') or e_post = 'Erlend';
```

```
(SELECT map_id FROM Maps_Interviewers WHERE username =
aes_encrypt('User','enckey'))UNION ALL(SELECT map_id
FROM Maps_Administrators WHERE username = aes_encrypt('User','enckey'));
```

```
SELECT cast(aes_decrypt(map_creator, 'enckey') AS CHAR(20)), title, date(start_date),
end_date, description,
astext(st_Centroid(geo_boundery)), zoom, astext(geo_boundery) FROM Maps WHERE
map_id = 9;
```

```
SELECT cast(aes_decrypt(map_creator, 'enckey') AS CHAR(20)), title, description,
start_date, end_date,
astext(geo_boundery), zoom ,map_id FROM Maps WHERE map_id = 9;
```

```
SELECT category_ID, category_name, category_type, category_image_or_color
FROM Maps_Categories WHERE map_id = 9;
```

```
SELECT question_ID, question FROM Maps_Questions WHERE map_id = 9;
```

```
SELECT cast(aes_decrypt(map_creator, 'enckey') AS CHAR(20)), title, description,  
start_date, end_date,  
astext(geo_boundery), zoom ,map_id FROM Maps WHERE map_id = 9;
```

```
SELECT question_ID, question FROM Maps_Questions WHERE map_id = 9;
```

```
SELECT A.respondent_ID, A.question_ID, A.Answer  
FROM Respondent_Answers AS A JOIN Maps_Respondents AS R ON A.respondent_ID  
= R.respondent_ID WHERE map_id = 9;
```

```
SELECT S.shape_id, cast(aes_decrypt(S.shape_creator, 'enckey') AS CHAR(20)),  
astext(S.center), astext(S.area_or_path),  
S.title, S.description, S.rate, M.category_type, M.category_image_or_color,  
respondent_ID  
FROM Shapes S JOIN Maps_Categories M ON S.category_ID = M.category_ID WHERE  
map_id = 9;
```

```
SELECT question FROM Maps_Questions WHERE map_id = '9';
```

```
SELECT astext(geo_boundery) FROM Maps WHERE map_id = '9';
```

```
SELECT astext(S.area_or_path), astext(S.center), cast(aes_decrypt(S.shape_creator,  
'enckey') AS CHAR(20)),  
S.title, S.description, S.rate, M.category_type, S.category_ID, S.respondent_ID  
FROM Shapes S JOIN Maps_Categories M ON S.category_ID = M.category_ID WHERE  
map_id = '9' AND shape_id = '18';
```

```
SELECT Q.question, R.Answer FROM Respondent_Answers AS R JOIN  
Maps_Questions AS Q ON R.question_ID = Q.question_ID  
WHERE map_id = '9' AND R.respondent_ID = '32';
```



```
SELECT first_name, last_name, issue_timestamp, message FROM Feedbacks as F JOIN  
Persons as P On F.username = P.username  
WHERE F.map_id = 9;
```

```
SELECT first_name, last_name FROM Persons WHERE username =  
aes_encrypt('User','enckey');
```

```
SELECT cast(aes_decrypt(R.username, 'enckey') AS CHAR(20)), Q.question, A.Answer  
FROM (Respondent_Answers AS A JOIN Maps_Questions AS Q ON A.question_ID =  
Q.question_ID)  
JOIN Maps_Respondents AS R ON R.respondent_ID = A.respondent_ID WHERE  
Q.map_id = 9;
```

```
SELECT map_id, map_creator, title, description, end_date,  
astext(Centroid(geo_boundary)), zoom, astext(geo_boundary)  
FROM Maps WHERE map_creator = aes_encrypt('User','enckey') AND map_id = 9;
```

```
SELECT question_ID, question FROM Maps_Questions WHERE map_id = 9;
```

```
SELECT cast(aes_decrypt(username, 'enckey') AS CHAR(20))  
FROM Persons where username = aes_encrypt('Erlend','enckey') or e_post = 'Erlend';
```

```
SELECT cast(aes_decrypt(username, 'enckey') AS CHAR(20))  
FROM Persons where username = aes_encrypt('Erlend','enckey') or e_post = 'Erlend';
```

```
SELECT map_id, map_creator, title, description, end_date,  
astext(Centroid(geo_boundary)), zoom, astext(geo_boundary)  
FROM Maps WHERE map_creator = aes_encrypt('User','enckey') AND map_id = 9;
```

```
SELECT respondent_ID FROM Maps_Respondents WHERE map_id = 9;
```

```
(SELECT map_id FROM Maps_Interviewers
```

```
WHERE username = aes_encrypt('User','enckey'))
```

```
UNION ALL
```

```
(SELECT map_id FROM Maps_Administrators WHERE username =  
aes_encrypt('User','enckey'));
```

```
SELECT cast(aes_decrypt(map_creator, 'enckey') AS CHAR(20)), title, date(start_date),  
end_date, description,  
astext(st_Centroid(geo_boundery)), zoom, astext(geo_boundery) FROM Maps WHERE  
map_id = 9;
```

```
<***** Update Queries *****>
```

```
UPDATE Persons SET login_pass = aes_encrypt('PASS','enckey'),  
first_name = 'FIRST_NAME', last_name = 'LAST_NAME', e_post = 'MAIL', telephone =  
'TELE'
```

```
WHERE username = aes_encrypt('Username','enckey');
```

```
UPDATE Maps SET title = 'TITLE', description = 'DESCRIPTION', end_date =  
'END_DATE' WHERE map_id = MAP_ID;
```

```
UPDATE Shapes SET title = 'TITLE', description = 'DESCRIPTION', rate = RATING  
WHERE shape_id = 18;
```

```
<***** Delete Queries *****>
```

```
DELETE FROM Shapes WHERE shape_id = 18;
```

```
<***** Insert Queries *****>
```

```
INSERT INTO Persons(username, login_pass, first_name, last_name, e_post, telephone)  
VALUES(aes_encrypt(%s,'enckey'),aes_encrypt(%s,'enckey'),%s,%s,%s,%s);
```

```
INSERT INTO Maps(map_creator, title, description, end_date, geo_boundery, zoom)
```

```
VALUES(aes_encrypt(%s,'enckey'),%s,%s,%s,ST_GEOMFROMTEXT('POLYGON((58.920260993961975 5.570166018900636,...))'),%s);
```

```
INSERT INTO Maps_Interviewers(username, map_id)
VALUES(aes_encrypt(%s,'enckey'),%s);
```

```
INSERT INTO Maps_Administrators(username, map_id)
VALUES(aes_encrypt(%s,'enckey'),%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Questions(question, map_id) values(%s,%s);
```

```
INSERT INTO Maps_Respondents(map_id, username) VALUES(%s,
aes_encrypt(%s,'enckey'));
```

```
INSERT INTO Respondent_Answers(respondent_ID, question_ID, Answer)
VALUES(%s,%s,%s);
```

```
INSERT INTO Shapes(category_ID, shape_creator, center, area_or_path, title,
description, rate, respondent_ID)
VALUES(%s, aes_encrypt(%s,'enckey'), POINT(58.93913120440394,
5.69005432875565),
st_geomfromtext('LINESTRING(58.93913120440394
5.69005432875565,58.93913120440394 5.69005432875565)'), %s, %s, %s, %s);
```

```
INSERT INTO Feedbacks(map_id, username, message)
VALUES(%s,aes_encrypt(%s,'enckey'),%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Categories(category_name, category_type,
category_image_or_color, map_id) VALUES(%s,%s,%s,%s);
```

```
INSERT INTO Maps_Interviewers(username, map_id)
VALUES(aes_encrypt(%s,'enckey'),%s);
```

```
INSERT INTO Maps_Administrators(username, map_id)
VALUES(aes_encrypt(%s,'enckey'),%s);
```

```
INSERT INTO Maps_Questions(question, map_id) VALUES(%s, %s);
```

```
INSERT INTO Respondent_Answers(respondent_ID, question_ID, Answer)
VALUES(%s, %s, "");
```

### 11.3 ArcGis Python Skript som konverterer til kategoriene i 3 separate filer, skrevet av veilederen Erlend Tøssebro

```
# -*- coding: cp1252 -*-
```

```
import arcpy
```

```
import sys
```

```
# Denne funksjonen brukes for aa splitte opp CSV fila i sine enkelte attributter
```

```
def strengsplitter(streng):
```

```
    resultat = []
```

```
    startindeks = 0
```

```
    sluttindeks = 0
```

```
    nvindeks = 0
```

```
    parentesnivaa = 0
```

```
    while nvindeks < len(streng):
```

```
        if streng[nvindeks] == ';' and parentesnivaa == 0:
```

```
            sluttindeks = nvindeks;
```

```
            resultat.append(streng[startindeks:sluttindeks])
```

```
            nvindeks = nvindeks + 1
```

```
            startindeks = nvindeks
```

```
        elif streng[nvindeks] == '(':
```

```
            parentesnivaa = parentesnivaa + 1
```

```
            nvindeks = nvindeks + 1
```

```
        elif streng[nvindeks] == ')':
```

```
            parentesnivaa = parentesnivaa - 1
```

```
            nvindeks = nvindeks + 1
```

```
        else:
```

```
            nvindeks = nvindeks + 1
```

```
    return resultat
```

```
# Hoveddelen av koden
```

```
def konverterFraWKTEnkel(filnavn):
```

```
filkomponenter = filnavn.split(".")
```

```
fil = open(filnavn)
```

```
# Bruker en fast plassering siden jeg ikke har faatt relativ plassering til aa virke enda
```

```
workspace = "C:\\arcpy_tester\\"
```

```
arcpy.env.workspace = "C:\\arcpy_tester\\"
```

```
areaCoordinates = filkomponenter[0] + "_shapeArea.shp"
```

```
areaCoordinatestest = "C:\\arcpy_tester\\" + areaCoordinates
```

```
centers = filkomponenter[0] + "_centers.shp"
```

```
centerstest = "C:\\arcpy_tester\\" + centers
```

```
# Leser de to første linjene og ignorerer den som har titlene paa attributtene
```

```
linje = fil.readline()
```

```
linje = fil.readline()
```

```
# Lager to ArcGIS feature classes, en for linjer og en for punkter
```

```
arcpy.CreateFeatureclass_management(workspace, areaCoordinates,  
"LINESTRING")
```

```
arcpy.CreateFeatureclass_management(workspace, centers, "POINT")
```

```
# Legger til attributtene i feature classene
```

```
arcpy.AddField_management(areaCoordinatestest, "Map_ID", "LONG")
```

```
arcpy.AddField_management(centerstest, "ID", "LONG")
```

```
arcpy.AddField_management(areaCoordinatestest, "User", "TEXT")
```

```
arcpy.AddField_management(centerstest, "User", "TEXT")
```

```
arcpy.AddField_management(areaCoordinatestest, "Title", "TEXT")
```

```
arcpy.AddField_management(centerstest, "Title", "TEXT")
```

```
arcpy.AddField_management(areaCoordinatestest, "Descr", "TEXT")
```

```
arcpy.AddField_management(centerstest, "Descr", "TEXT")
```

```
arcpy.AddField_management(areaCoordinatestest, "Rating", "SHORT")
```

```
arcpy.AddField_management(centerstest, "Rating", "SHORT")
```

```
arcpy.AddField_management(areaCoordinatestest, "Catg_type", "TEXT")
```

```

arcpy.AddField_management(centerstest, "Catg_type", "TEXT")
arcpy.AddField_management(areaCoordinatestest, "Resp_ID", "SHORT")
arcpy.AddField_management(centerstest, "Resp_ID", "SHORT")

# Lager kursorer for innsetting i begge feature classene
areaCursor = arcpy.da.InsertCursor(areaCoordinatestest, ["Map_ID", "User",
"Title", "Descr", "Rating", "Catg_type", "Resp_ID", "SHAPE@"])
centersCursor = arcpy.da.InsertCursor(centerstest, ["SHAPE@", "ID", "User",
"Title", "Descr", "Rating", "Catg_type", "Resp_ID"])

# While-loop som leser fra fila og lager objektene i de to feature classene
while linje != "":
    kolonner = strengsplitter(linje)
    if kolonner[2] != "None":
        areaCursor.insertRow([int(kolonner[0]), kolonner[4], kolonner[5], kolonner[6],
int(kolonner[7]), kolonner[8], int(kolonner[9]), arcpy.FromWKT(kolonner[2], "")])
        print("Setter inn i Areas: " + kolonner[0] + " " + kolonner[4] + " " +
kolonner[5] + " " + kolonner[6] + " " + kolonner[7] + " " + kolonner[8] + " " +
kolonner[9])
    if kolonner[3] != "None":
        centersCursor.insertRow([arcpy.FromWKT(kolonner[3], ""), kolonner[0],
kolonner[4], kolonner[5], kolonner[6], kolonner[7], kolonner[8], kolonner[9]])
        print("Setter inn i centers: " + kolonner[0])
    linje = fil.readline()
del areaCursor
print("Etter sletting av kursor")
del centersCursor

# Kaller hovedmetoden med kommandolinjeargument 1
if __name__ == "__main__":
    konverterFraWKTEnkel(sys.argv[1])

```

## 11.4 Tilbakemeldinger

### 11.4.1 Tilbakemelding fra Doktorgradskandidaten i Institutt for industriell økonomi, risikostyring og planlegging ved TN-fakultetet, ved Universitet i Stavanger, Ana Llopis Alvarez.

#### FEEDBACK: CITY SHARE APP

Concerning this new APP called CITY SHARE, I just want to show my gratitude for all the effort behind it as well as the result.

As a PhD candidate at the Faculty of Science and Technology (Department of Industrial Economics, Risk Management and Planning), this APP has been a collaboration between two students of the IT Department and me.

In relation to my PhD programme: *“Residential conditions for immigrant population”*, there was a need of creating this APP in order to develop a new methodology system for interviewing people in the area of Storhaug. Consequently, the APP has been developed as my needs were showing up.

The project has been followed weekly, where the students and I were sharing opinions about the improvements of the APP. I must appreciate how the whole project was turned into all my desires, since every single requirement that I asked for, it has been accomplished.

It has definitely been a challenge to become part of each other project, since we had to share backgrounds and purpose of the whole of it. The students became interested on my PhD Programme, and with it, with the new methodology requirements.

It has totally been a transition between the first “product” and the result, since several changes have also been part of this project, but always with the intention of improvement.

The APP has been tested, and totally acquired as the APP we expected.

I would like to congratulate the students for the effort shown, and the hard work that it has been behind all over the process.



#### 11.4.2 Tilbakemelding fra Doktorgradskandidaten i Institutt for data- og elektroteknologi (IDE) ved TN-fakultetet, ved Universitet i Stavanger, Dario Garigliotti.

I find the overall design OK. It's a demo and likely comes from Bootstrap but anyways it looks clear and intuitive without a lot of heavy, "fancy" stuff.

I didn't know what the functionalities were about, and don't get most of the Norwegian words :p, yet I understood what it does so it should be easy to use.

Some stuff you might consider (yet I don't know your problem and desired/expected features):

- Maybe the interviewed could (I don't know if it makes fully sense) add new possible item categories, in particular new punkter that she might need for marking the map.

- Maybe you want to add some choices selection elements instead of text input fields in "Nationality" (should be easy to get predefined list of countries) and "Reason" (if you have some frequent reasons beforehand).

That could solve issues about if the user types wrongly or with different capitalization (e.g. "Work" and "work") so filtering would catch more results.

It maybe also split in "Nationality" and "Period" separated selections.

- Maybe that "Reason" select can have a "Other" with a field to input text.

Similar to "Feel..." questions: you may have "yes", "no", "Better let me explain" (with input text) radio buttons.

- Maybe you don't want to give predefined reasons to avoid biasing the interviewed.

- The review (vurdering) is fine, but could have a "very bad" and "very good" endpoint labels (or something like that) so it's clear what to do with the sliding circle.

- The grouping feature looks nice.

Does it work only for green stuff, or for red, black categories?

Might the analyzer want to optionally group stuff in a circle of a given radius=distance?  
(I.e., if it makes sense, and it's doable, it could be cool.)

-----

Can I ask you how you get

- the punkter categories (and their icons)?
- the grouping behavior?

Are those coming from the Google Maps API?

Hope it's useful.

Best,

Darío.

### **11.5 Spesifisert arbeidsmengde og timeliste**

Tabellen under spesifiserer antall timer jeg har brukt i arbeidet med dette prosjektet.

Pauser, arbeidet med essayet og møter med veilederen Erlend Tøssebro og oppdragsgiverne fra Instituttet for industriell økonomi, risikostyring og planlegging ved Universitet i Stavanger er ikke tatt med disse timene.

Vårsemesteret 2018									
Januar		Februar		Mars		April		Mai	
Dato	Timer	Dato	Timer	Dato	Timer	Dato	Timer	Dato	Timer
8/1	4,5	1/2	1	5/3	7	1/4	3,5	2/5	2,5
10/1	1,5	2/2	4,5	6/3	3	2/4	6	3/5	6
11/1	2	3/2	6	7/3	1,5	3/4	4	4/5	3,5
13/1	8,5	4/2	3,5	10/3	7	6/4	4	5/5	4
16/1	5,5	5/2	4,5	11/3	3,5	7/4	4,5	6/5	3
22/1	6	6/2	4,5	12/3	5	8/4	2,5	7/5	5
24/1	2	7/2	2	13/3	5	9/4	7,5	8/5	5
25/1	3	8/2	3	14/3	5,5	10/4	3	9/5	4
26/1	1,5	9/2	3	16/3	1	11/4	5	10/5	5
27/1	4	10/2	5	17/3	2,5	12/4	8	11/5	7,5
28/1	5,5	11/2	4,5	18/3	4	13/4	4	12/5	8,5
29/1	10	12/2	8	19/3	6,5	14/4	3,5	13/5	5,5
31/1	4	13/2	2,5	20/3	4	15/4	4	14/5	7
		14/2	9	21/3	3	16/4	9,5	15/5	5,5
		15/2	5,5	22/3	2,5	17/4	3,5		
		16/2	4	23/3	2,5	18/4	7,5		
		19/2	6	27/3	1	19/4	8		
		23/2	5	28/3	2,5	20/4	2,5		
				29/3	5	21/4	4,5		
				30/3	1,5	22/4	4,5		
				31/3	3,5	27/4	2,5		
Totalt i januar		Totalt i februar		Totalt i mars		Totalt i april		Totalt i mai	
58 timer		81,5 timer		77 timer		102 timer		72 timer	
Totalt i vårsemesteret 2018								390,5 timer	

## Referanseliste

- Figur 1: Jonvik, Merete & Lindland, Kristiane & Tvedt, Helge L. & Müller-Eie, Daniela & Melberg, Kjersti (2018) Hillevåg – En sosiokulturell stedsanalyse. IRIS. Rapport under produksjon.
- Figur 22: En sammenligning mellom bruk av stasjonære datamaskiner og bruk av Mobile enheter. (2017). Hentet den 19/02/2018 fra <http://www.startux.net/your-smartphone-vs-your-pc/>.
- Apple Inc (2018). WebFrame. Hentet den 20/04/2018 fra <https://developer.apple.com/documentation/webkit/webframe>.
- Elmasri R. og Navathe B. Shamkant. (2010). Fundaments of Database systems edition 6. United States: Pearson.
- Flask: Python Flask development frame. Hentet den 19/02/2018 fra <http://flask.pocoo.org/> og <http://jinja.pocoo.org/>.
- Google Inc (2018). *Maps JavaScript API*. <https://developers.google.com/maps/documentation/javascript/tutorial>.
- Google (2018). Building Web Apps in WebView. Hentet den 20/04/2018 fra <https://developer.android.com/guide/webapps/webview>.
- Hjelp og støtte for JavaScript, JQuery, CSS, Bootstrap: w3schools. 2018. <https://www.w3schools.com/>.
- Informasjoner om Python og PHP: Cloudways. (2017). PHP or Python— Which Language Should You Learn in 2017. Hentet den 10/01/2018 fra <https://hackernoon.com/php-or-python-which-language-should-you-learn-in-2017-3ced1fd75ee2>.
- Informasjoner om JavaScript og Ajax: Cloudways. (2017). PHP or Python— Which Language Should You Learn in 2017. wikipedia. 2018. Hentet den 10/01/2018 <https://no.wikipedia.org/wiki/JavaScript>, og [https://no.wikipedia.org/wiki/Ajax\\_\(programming\)](https://no.wikipedia.org/wiki/Ajax_(programming)).

- Informasjoner om Bootstrap: wikipedia. 2018. Hentet den 10/01/2018  
[https://en.wikipedia.org/wiki/Bootstrap\\_\(front-end\\_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)).
- Informasjoner om forskjellige kart APIs: Janet Wagner. 2015. Hentet den 10/01/2018  
<https://www.programmableweb.com/news/top-10-mapping-apis-google-maps-microsoft-bing-maps-and-mapquest/analysis/2015/02/23>.
- Informasjoner om MySQL: wikipedia. 2015. Hentet den 10/01/2018  
<https://no.wikipedia.org/wiki/MySQL>. Og [https://en.wikipedia.org/wiki/Database\\_trigger](https://en.wikipedia.org/wiki/Database_trigger).
- Hjelp og støtte: stackoverflow. 2018. <https://stackoverflow.com/>.
- Kartikoner: Nicolas Mollet. 2018. Hentet den 08/02/2018 fra  
<https://mapicons.mapsmarker.com>.
- Plugin for fargevelger: Spectrum. 2018. Hentet den 05/03/2018 fra  
<https://bgrins.github.io/spectrum/#methods-get>.
- Plugin for multi- og ikonvelger: HARVEST. 2018. Hentet den 06/03/2018 fra  
<https://harvesthq.github.io/chosen/>.