



**SALG  
ELLER  
UTLEIE**

# BACHELORPROSJEKT

Ved institutt for informasjonsteknologi

**OSLOMET**

Daniel Sørenes  
s326286

Anders Odin Andberg  
s320926

Stian Wingaard  
s326323

Magnus Unstad  
s235764



## Institutt for Informasjonsteknologi

Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo

Besøksadresse: Holbergs plass, Oslo

# BACHELORPROSJEKT

Telefon: 22 45 32 00

PROSJEKT NR.

21

TILGJENGELIGHET

Åpen

PROSJEKTOPPGAVENS TITTEL

Salg eller utleie – En webapplikasjon for Nabobil.no

DATO

25.05.2020

ANTALL SIDER / BILAG

59/12

PROSJEKTDeltakere

Magnus Unstad - s235764  
Anders Odin Andberg - s326286  
Stian Wingaard - s326323  
Daniel Sørenes - s320926

VEILEDER

Jorge Moreno-Trejo

OPPDRAGSGIVER

Nabobil.no

KONTAKTPERSON

Cristian Hager

SAMMENDRAG

Dette bachelorprosjektet ved OsloMet er utviklet og skrevet i samarbeid med Nabobil.no AS. Målet med oppgaven er å utvikle en progressiv webapplikasjon som skal sammenligne bruktbilens utleie- og salgsverdi. Tjenesten skal være tilgjengelig for allmenheten og fungere som en markedsføringsplattform. Forhåpentligvis fører dette til at flere velger å leie ut bilen på nabobil.no.

3 STIKKORD

Webutvikling

Delingsøkonomi

Prototyping

## 1 Forord

Denne rapporten er basert på prosjektet vi har gjennomført i samarbeid med Nabobil.no AS (heretter Nabobil). Rapporten er delt inn i flere deler som går nærmere inn på blant annet kravspesifikasjonen, planleggingen, design, dokumentasjon og selve løsningen. Relevante figurer er vedlagt fortløpende, som gjør det enklere å se det som diskuteres skriftlig.

Vi ønsker å rette en stor takk til Nabobil som valgte å samarbeide med oss, og håper at de blir fornøyde med produktet vi overleverer. Vi vil også rette en takk til vår veileder Jorge Moreno-Trejo for god oppfølging underveis i prosjektet.

Tjenesten vi har utviklet vil bli tilgjengelig for allmenheten gjennom domenet [www.salgellerutleie.no](http://www.salgellerutleie.no). Foreløpig kan den besøkes via <https://bilpris.azurewebsites.net>.

## Innholdsfortegnelse

1	Forord .....	3
2	Innledning.....	7
2.1	Gruppen.....	7
2.2	Oppdragsgiver .....	7
2.3	Bakgrunn .....	8
2.4	Prosjektbeskrivelse.....	8
2.5	Rammer og begrensninger .....	8
2.6	Mål.....	9
2.7	Utfordringer relatert til Covid-19 .....	10
3	Kravspesifikasjon .....	10
4	Planlegging og verktøy.....	12
4.1	Verktøy .....	12
4.2	Teknologier.....	13
4.3	Planleggingen av prosjektet .....	13
4.3.1	Fremdrift .....	14
4.3.2	Fossefallsmetoden .....	14
4.3.3	Frittstående eller integrert plattform .....	15
4.3.4	Risikoanalyse.....	15
5	Design av applikasjon .....	17
5.1	Generelle hensyn bak design .....	17
5.2	Konkurrenter .....	19
5.3	Lo-fi prototype .....	21
5.4	Hi-Fi prototype .....	23
6	Produktdokumentasjon.....	26
6.1	Forord .....	26
6.2	Arkitektur .....	26
6.3	Kjøring og hosting av applikasjonen.....	27
6.4	Samsvar mellom kravspesifikasjon og produkt.....	28
6.5	Backend .....	28
6.5.1	Verktøy.....	28
6.5.2	.Net Core .....	28
6.5.3	Microsoft Azure Portal .....	29

6.5.4	GraphQL .....	29
6.6	Arkitektur .....	29
6.6.1	Prosjektarkitektur og kodearkitektur .....	29
6.7	Databasen/skytjenesten .....	31
7.8.1	Caching .....	31
6.8	Datastrukturer .....	32
6.9	API'er .....	33
6.9.1	Nabobils bilinformasjon- og salgspris-API .....	33
6.9.2	Nabobils utleiepris-API .....	33
6.9.3	Post Brings API .....	34
6.10	Front-end .....	34
6.10.1	Verktøy .....	34
6.10.2	Create React App .....	34
6.10.3	Formik .....	34
6.10.4	Material-Ui .....	35
6.10.5	Styled Components .....	35
6.10.6	Chart.js .....	35
6.10.7	Axe-Core/React-Axe .....	35
6.10.8	i18next/react-i18next .....	35
6.11	Universell utforming i front end design .....	35
6.12	Tekniske løsninger front-end .....	36
6.12.1	Arkitektur & datastruktur .....	36
6.12.2	Datahenting .....	37
6.12.3	Theming og styling .....	39
6.12.4	Animasjon .....	41
6.12.5	Forms og validering .....	42
6.12.6	Bytting av språk .....	43
6.12.7	Søkemotoroptimalisering .....	43
7	Løsningen .....	45
7.1	Innledning .....	45
7.2	Versjon for stor skjerm .....	45
7.3	Mobilversjon .....	51
7.4	Visualisering .....	54

7.4.1	Visualisering .....	54
7.4.2	Farger .....	56
7.5	Forbedringspotensial .....	57
8	Konklusjon .....	58
9	Vedlegg .....	60
9.1	Prosjektlogg .....	60
9.2	Systemtest .....	60
9.2.1	Systemtesting .....	60
9.2.2	Testplan .....	60
9.2.3	Testprosedyre .....	62
9.2.4	Godkjenningskriterier .....	62
9.2.5	Testrapport systemtest .....	62
9.2.6	Konklusjon .....	65
9.3	Brukertest .....	65
9.3.1	Forventninger til bilens salgs- og utleieverdi .....	65
9.3.2	Selve nettsiden .....	66
9.3.3	Informasjon på nettsiden .....	66
9.3.4	Utleie på nabobil.no .....	66
10	Prosjektdagbok .....	68
11	Referanser .....	70

## 2 Innledning

Følgende del av rapporten vil redegjøre kort for de ulike delene og oppbyggingen av prosjektet.

### 2.1 Gruppen

Gruppen består av fire studenter på Anvendt datateknologi ved OsloMet – Storbyuniversitetet. Samlet har de kompetanse innenfor flere teknologier og fagfelt. I utdanningsløpet har medlemmene tilegnet seg kunnskap om interaksjonsdesign, programutvikling og hvordan mennesker interagerer med teknologi. Videre ser gruppen frem til å sette teori i praksis og presentere et ferdig produkt til oppdragsgiver.

Arbeidsoppgavene i prosjektet er i utgangspunktet fordelt på følgende måte, men medlemmene har også bidratt på tvers av dette.

Person	Stilling
Odin Andberg	Frontendutvikler
Daniel Sørenes	Backendutvikler og skyansvarlig
Stian Wingaard	Backendutvikler
Magnus Unstad	Interaksjonsdesigner og ansvarlig tekstforfatter

### 2.2 Oppdragsgiver

Nabobil (Nabobil.no, 2020) er en norsk gründerbedrift og drifter en nettbasert plattform for formidling av delingsutleie av bil, som vil si at de driver en virksomhet som opererer i den mye omtalte delingsøkonomien (Barland, 2015). Det kan på mange måter kalles en slags «AirBnB for biler» og er ledende i Norge innen bildeling.

På Nabobils plattform kan man registrere seg for å leie ut sin egen bil eller for å leie andres biler. Siden oppstarten i september 2015 har selskapet opplevd enorm vekst på sin plattform. Fordelene ved å leie ut privatbiler på Nabobils plattform er at forsikring og administrasjon av leieforholdet utføres av Nabobil, på den måten blir prosessen for utleier mindre tidkrevende og mer pålitelig for leietaker.

Fremover skal Nabobil satse på nøkkelløs teknologi, for å forenkle privatutleie av biler. Med denne teknologien kan leietakerne åpne og låse bilen med Nabobils applikasjon (Hopland, 2019). I utgangspunktet må utleier og leietaker møtes for å overlevere bilnøkklene, ved å eliminere dette skrittet er det blant annet enklere å leie bil på kort varsel. Med denne teknologien kan man leie ut bilen når man er på jobb, ferie eller på besøk hos bestemor.

Nabobil har siden oppstarten opplevd kraftig vekst, og i 2019 ble de kjøpt opp av den amerikanske bildelingsgiganten Getaround (Hopland, 2019). Med oppkjøpet følger et ansvar for nordisk ekspansjon (Børnick-Sørhaug, 2019). Nabobil skulle etter planen bli til Getaround våren 2020, denne prosessen er satt på vent grunnet utbruddet av Covid-19.

Gruppens ansvarlige kontaktperson hos Nabobil er Christian Hager. Andre kontaktpersoner i Nabobil er Jenny og Theodor.

## 2.3 Bakgrunn

Nabobil er en relativt ung start-up med begrensede ressurser og det kan medføre harde prioriteringer av arbeidsoppgaver. Markedsføring av plattformen er en stor del av virksomheten, spesielt med tanke på at markedet de opererer i er ukjent for mange.

I flere år har Nabobil ønsket å utvikle en tjeneste som gir brukerne innsikt i hvor mye bilen deres er verdt. Nabobils opplevelse er at de færreste har et bevisst forhold til egen bils verdi, og at dette utgjør et problem da bileiere ikke nødvendigvis ser verdien de innehar.

Ideen er at tjenesten skal bevisstgjøre brukerne på hvor mye de kan tjene dersom de leier ut bilen på Nabobils plattform. Webapplikasjonen vil tilby en tjeneste i form av verdivurdering av bilene, men samtidig være en markedsføringsplattform for Nabobil. Målet er at brukerne skal se hvor mye utleieverdi som ligger i den enkelte bil og derfor registrere bilen på Nabobils plattform.

Med begrensede ressurser har en slik tjeneste blitt nedprioritert. Derfor har nabobil engasjert gruppen for å utvikle det vi har valgt å kalle «Salg eller utleie».

## 2.4 Prosjektbeskrivelse

Prosjektet som er beskrevet i forrige avsnitt skal utvikles, implementeres og overleveres Nabobil før utgangen av mai 2020. Prosjektet skal foregå i flere faser, henholdsvis planlegging, design, utvikling og testing/feilretting. Dette beskrives nærmere senere i oppgaven.

I utgangspunktet er det ønskelig å benytte agile metoder gjennom utviklingsdelen av prosjektet. Det vil bidra til å bedre flyten i prosjektet slik at sluttproduktet blir utviklet på en effektiv og best mulig måte.

Webapplikasjonen som skal først og fremst designes. Nabobil har gitt gruppen frihet til å utforme applikasjonen slik de ønsker det, og har ingen kommentarer til selve designet, teknologi, utviklingsmetoder, fargevalg og lignende.

## 2.5 Rammer og begrensninger

I samarbeid med Nabobil er det utarbeidet en kravspesifikasjon som vist i del 5. Det er bestemt at gruppen skal holde seg innenfor kravspesifikasjonens punkter. Dersom punktene blir ferdigstilt raskere enn først antatt, vil gruppen sammen med oppdragsgiver vurdere om det er nødvendig å utarbeide flere punkter som et tillegg til kravspesifikasjonen.

I løpet av prosjektet ser ingen av partene behov for dette.

Gruppens oppgave er å lage en webapplikasjon som sammenligner salgs- og utleieverdien av en personbil. På Nabobil er det ikke mulig å leie ut andre typer kjøretøy, for eksempel



lastebil, bobil og motorsykel. Derfor er det ikke mulig å søke på registreringsnummer knyttet til slike kjøretøy.

Data om bilenes pris og utleieverdi får vi direkte fra API'er som Nabobil har gitt gruppen tilgang til. Å få tilgang til disse var en forutsetning for prosjektet. Dette vil si at gruppen ikke lager logikk som beregner denne informasjonen selv. Det vil kun bli gjort mindre justeringer av dataen vi får fra API'ene, som for eksempel reduksjon av salgsverdi basert på antall år frem i tid.

Registreringsnummer har vanligvis formatet «AA12345». I senere tid har det blitt mulig å søke om personlige bilskilt (Statens vegvesen, 2020). API'ene støtter dessverre ikke personlige bilskilt, derfor er det ikke mulig å søke opp disse i webapplikasjonen. I feltet hvor brukeren kan skrive inn registreringsnummeret vil vi legge til validering som støtter personlige bilskilt, forutsatt at API'ene i fremtiden støtter dette. Dersom brukeren har et personlig bilskilt, kan de likevel benytte det originale registreringsnummeret for å søke opp bilen.

Når webapplikasjonen er overlevert til Nabobil er de selv ansvarlige for å markedsføre tjenesten.

Når man leier ut formuesobjekter kan det utløse beskatning. Dette området er veldig komplisert og utenfor vår kompetanse. Webapplikasjonen vil derfor ikke ta for seg eventuell beskatning og avgifter knyttet til salg eller utleie av bil. På Nabobils hjelpesider henviser de brukere direkte til Skatteetaten. Om problemstillingen oppstår vil gruppens webapplikasjon henvise til samme sted.

I Nabobils eksisterende utleieverdikalkulator (Nabobil.no, 2020) er det utleieverdien avhengig av hvorvidt utleier installerer Nabobil Uten Nøkkel eller ikke. Dette ligger innenfor vår opprinnelige kravspesifikasjon, men Nabobils API inneholder ikke data om nøkkelfri teknologi og dette vil derfor ikke bli berørt i prosjektet. Nabobil må selv legge dette inn i webapplikasjonen etter overlevering.

## 2.6 Mål

Målet med oppgaven er å utvikle en tjeneste som har verdi for Nabobil. Gruppen håper at webapplikasjonen fører til at flere velger å tilgjengeliggjøre bilen for utleie på nabobil.no. Videre er det en målsetting at webapplikasjonen skal være den foretrukne tjenesten å bruke når man trenger et estimat på bilens verdi. På den måten fungerer webapplikasjonen som en tjeneste for allmenheten, men også en markedsføringsplattform for Nabobil.

Webapplikasjonen skal være integrert med sosiale medier, dette gir brukerne mulighet til å dele verdivurderingen av sin bil. Et av målene til gruppen og Nabobil er at så mange som mulig deler verdivurderingen. Det kan føre til at flere blir bevisst på tjenesten og forhåpentligvis tar den i bruk.

Gruppen ønsker gjennom prosjektet å oppnå betydelig læringsutbytte, særlig ved å benytte dagsaktuelle teknologier som brukes i næringslivet. Dette innebærer at en målsetting ved gjennomføring av prosjektet er å sette seg inn i flere nye teknologier. Nabobil har gitt oss

frie tøyler når det kommer til teknologier og gruppen vil derfor sette sammen sin egen teknologiportefølje, hvor noe er nytt og noe er kjent fra tidligere emner på studiet.

## 2.7 Utfordringer relatert til Covid-19

Det har vært spesielt krevende å utvikle bachelorprosjektet i den usikre tiden vi har vært gjennom. Universitetets lokaler har vært stengt, gruppens medlemmer har ikke kunne møtes og Nabobil har blitt påvirket negativt.

Nedstengingen av universitetet har gjort at det er vanskeligere å møtes for å jobbe og diskutere. Omstilling er nøkkelordet, som et alternativ benyttes Teams og andre digitale ressurser for å gjøre opp for tapt tid sammen.

Nabobil har blitt påvirket og overgangen til Getaround sin plattform er utsatt på ubestemt tid. I tillegg har de opplevd inntektsbortfall som følger av redusert utleieaktivitet. Til tross for utfordringene kunne prosjektet fortsette.

Som gruppe har vi tatt lærdom fra situasjonen og blitt mer bevisst på at uforutsette hendelser kan oppstå. Selv om risikoanalysen i forprosjektet inneholdt et punkt om force majeure-hendelser, var det nærmest utenkelig at dette skulle forekomme og det ble derfor ikke utviklet en god nok plan for hvordan en slik situasjon skulle håndteres.

## 3 Kravspesifikasjon

Etter oppstartsmøte med oppdragsgiveren ble det utformet en kravspesifikasjon som beskriver prosjektet i detalj. Det var enighet om at en god og velfungerende grunnstruktur var viktigere enn å produsere tilleggsfunksjonalitet. Tjenesten som overleveres oppdragsgiver skal være produksjonsklar. Dersom dette oppnås og gruppen fortsatt har mer tid til gode kan det vurderes om tilleggsfunksjonalitet skal utvikles.

Kravspesifikasjonen skal gi innsikt i hva som må utvikles, men gir også innsikt i hvordan brukerens reise gjennom webapplikasjonen er. Underveis i prosjektet ble kravspesifikasjonen brukt aktivt for å utvikle de forskjellige komponentene. Tabellen under er revidert underveis i prosjektet og avviker noe fra kravspesifikasjonen slik presentert i forprosjektrapporten.

Tabell 1 - Kravspesifikasjon

Nr.	Omhandler	Beskrivelse
1	Brukerens inputs	Brukeren skal kunne gjøre oppslag på bil ved å oppgi følgende input: <ul style="list-style-type: none"><li>• Bilens registreringsnummer</li><li>• Postnummer</li><li>• Bilens kilometerstand</li></ul>
2	Informasjon om bilen	Brukeren får informasjon om bilen sin og hvor mye bilen er verdsatt på bruktbilmarkedet.
3	Informasjon om bilutleie	Brukeren får informasjon om hvor mye lignende biler i samme område tjener ved å leies ut på nabobil.no.

4	Informasjon om nøkkelfri teknologi	Brukeren får informasjon om bilen støtter nøkkelfri teknologi, i så fall får en også informasjon om hvordan det kan installeres.
5	Interaktiv utleieprising	Brukeren skal kunne velge hvor mange dager i uken de ønsker å leie ut bilen og få et utleieprisestimat basert på dette.
6	Salgspris etter utleie	<p>Brukerne skal få informasjon om bilens verdi etter X antall år, i tillegg til årlig utleieinntekt de samme årene.</p> <p>På den måten kan brukeren sammenligne fortjenesten ved å selge bilen i inneværende år kontra fortjenesten ved å selge den om X antall år. Brukeren skal bruke et interaktivt element for å endre antall år.</p> <p>I dette regnestykket må det tas høyde for årlige kostnader med bilhold, tap årlig verdi på bilen, skatter og avgifter og fortjenesten ved utleie på Nabobil.</p> <p>Biler som er mer enn 20 år gammel kan ikke leies ut på nabobil. Brukerne skal ikke kunne velge flere år enn det er mulig å leie ut bilen. For eksempel, dersom en bil er fra 2010 vil maks år med utleie bli 10 år.</p>
7	Informasjon om Nabobil	<p>Alternativ oppgave ved ledig kapasitet:</p> <p>Brukeren skal få informasjon om hvordan det er å leie ut på Nabobil, for eksempel:</p> <ul style="list-style-type: none"> <li>• Historier fra andre utleiere</li> <li>• Informasjon om inkludert forsikring</li> <li>• Generell informasjon om Nabobil</li> <li>• Bloggdel på siden hvor vi deler/skriver artikler/innlegg</li> </ul>
8	Språk	Siden skal være på Norsk og Engelsk
9	Sosiale medier	Det skal være mulighet å dele informasjon om bilens verdi på sosiale medier.
10	Visualisering av pris	Fortjenesten/tapet fra punkt 5 og 6 skal visualiseres i en interaktiv graf.
11	Forbehold angående tjenesten	Vi må gi klar informasjon til brukeren om at prisforslagene er estimerer. Det må også komme tydelig fra at Nabobil ikke garanterer at noen ønsker å leie bilen, sel om man averterer den.
12	Videresende bruker til Nabobil	Brukere skal kunne trykke på knappen «Lei bilen min ut på nabobil» eller lignende. «Ta meg til Nabobil», «Registrer meg på Nabobil»
13	Miljøeffekten av delingsøkonomien	Det er et ønske at vi skal informere brukerne om miljøeffekten det har at flere tar del i delingsøkonomien.

## 4 Planlegging og verktøy

### 4.1 Verktøy

I tabellen under gis det en kort oversikt over alle verktøy som blir benyttet underveis.

Tabell 2 - Verktøy

Navn	Bruk	Beskrivelse
Adobe XD	Utvikling av webapplikasjonens prototype.	Verktøyet brukes for å lage interaktive prototyper av høy fidelitet.
Microsoft Visual Studio Code	Skrive prosjektets frontend programkode.	Et utviklermiljø optimalisert for å bygge moderne web- og skyapplikasjoner.
Miro	Administrere prosjektet og arbeidsflyt.	Miro er en sanntids-tavle og et samarbeidsverktøy for å administrere prosjekter.
Slack	Kontakt mellom gruppens medlemmer og Nabobil.	Er et kommunikasjonsverktøy som er spesielt vennlig for teknologibedrifter.
GitHub	Lagring og deling av kildekode mellom gruppen og Nabobil. Brukt til versjonskontroll, håndtering og distribuering av produksjonsversjon.	En plattform utviklet for deling av kodeprosjekter.
Azure Portal	Legge webapplikasjonen på en liveserver.	Er en cloud computing-plattform og infrastruktur levert av Microsoft.
Trello	En digital tavle for å holde oversikt over gjøremål som skal gjennomføres/er under arbeid/er ferdige.	Et samarbeidsverktøy som organiserer prosjekter på en digital tavle.
Visual Studio 8.5.4 (community 2019)	Skrive prosjektets back-end programkode.	Et integrert utviklingsmiljø (IDE) for Microsofts .NET platform.
Adobe Illustrator CS6	Brukt til å lage logo for Salgellerutleie.no.	Et vektorbasert verktøy for å lage grafiske elementer.
Axe-Core/React-Axe	Teste webapplikasjonen for universell utforming.	Et verktøy for testing av universell utforming basert på WCAG.
Cypress	Brukt til systemtesting.	Et JavaScript-basert bibliotek for å gjøre ende-til-ende testing.

## 4.2 Teknologier

Den følgende tabellen beskriver kort de teknologier som benyttes i utviklingen av webapplikasjonen. Tabellen er ment for å gi en enkel innføring, og teknologiene vil bli beskrevet i større detalj videre i oppgaven.

Tabell 3 - Teknologier

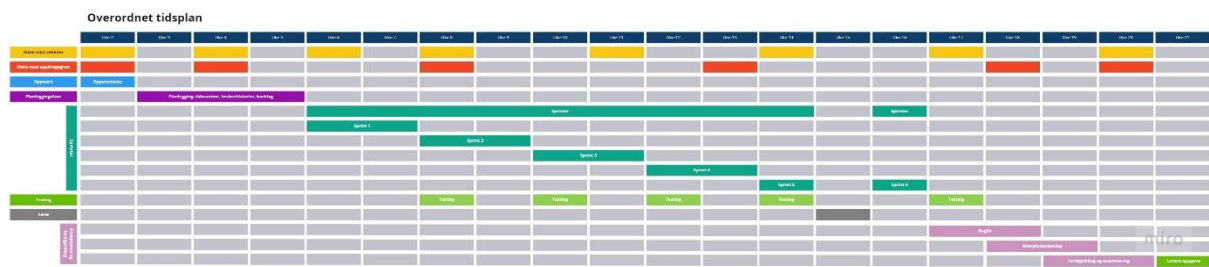
Navn	Bruk	Beskrivelse
React	Utvikle frontend.	Et JavaScript-bibliotek for frontend-utvikling, utviklet av Facebook.
C#	Utvikle backend.	Et objektorientert programmeringsspråk utviklet av Microsoft, basert på kodespråkene C++ og Java.
graphql	Spørringer på data til API'er fra Nabobil.	Er et kodespråk for API'er designet for å kjøre spørringer og manipulere API'ene.
Formik	Til skjemaer på webapplikasjonen.	Er et MIT-lisensiert bibliotek som tar for seg håndtering av states, validering og innsending av skjemaer i React.
Material UI	UI komponenter for frontend.	Et rammeverk med komponenter for React.
Chart.js	Til å visualisere utregningen av bilens verdi.	Et JavaScript-bibliotek for å lage interaktive visualiseringer, for eksempel grafer.
Create React App	Sette opp en React-app med utviklingsmiljø.	Et verktøy som setter opp React-prosjekter.
Styled Components	CSS styling direkte i JavaScript kode.	Et bibliotek for React som gjør det mulig å bruke CSS direkte i JavaScript kode.
.Net Core	Rammeverk for backend.	Programvare-rammeverk for operativsystemene Windows, Linux og macOS.
l18next/ reacti18next	Til å trekke ut all tekst som skulle oversettes til engelsk.	Trekker ut all tekst fra komponenter og erstatter det med en funksjon som henter tekst fra et JSON-objekt.
REST	Brukt av API'et til posten, Bilinfo API'et fra Nabobil og mellom back- og frontend i webapplikasjonen.	Er en arkitekturstil for å designe nettverk applikasjoner.

## 4.3 Planleggingen av prosjektet

Tidlig bestemte gruppen seg for å benytte agile metoder for få god struktur i arbeidet, valget falt derfor på å benytte Scrum. Fem sprinter ble planlagt og planen var at selve webløsningen skulle være klar til uke 17. Perioden etter uke 17 og frem mot levering skulle brukes til feilretting, testing og ferdigstilling av tjenesten. Dette kan ses i fremdriftsplanen vedlagt. Videre planlegger gruppen hyppige møter med veileder og oppdragsgiver. En

risikovurdering ble utarbeidet tidlig i prosjekteringsfasen for å vurdere hvilke utfordringer som kan oppstå og hvordan gruppen kan jobbe for å motarbeide disse.

#### 4.3.1 Fremdrift



Figur 1 - Overordnet tidsplan for prosjektet

En fremdriftsplan ble utviklet i Miro. Dette var den overordnede planen for gjennomføringen av prosjektet. Her planla vi fem sprints som skulle innebære at webapplikasjonen var ferdig uken etter påske. Det ble også lagt inn god tid inn til feilretting og akseptansetesting mot slutten av prosjektet.

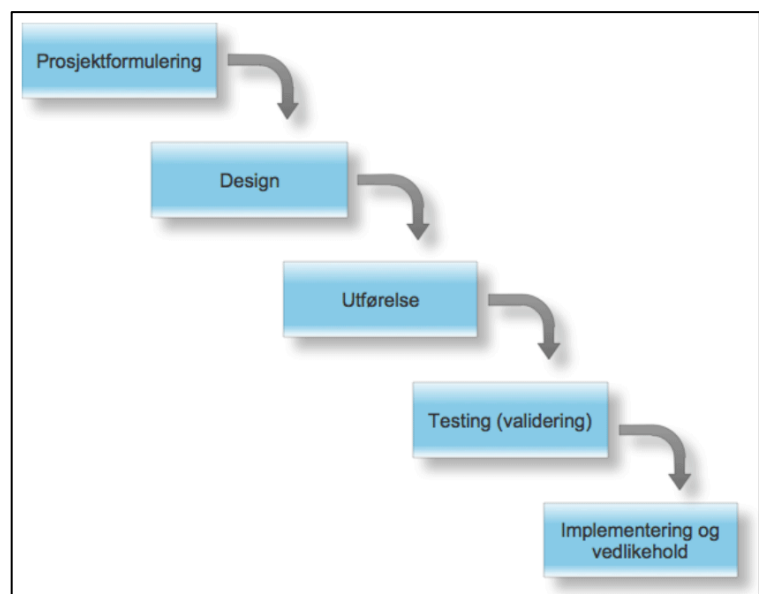
Grunnet Covid-19 var det begrenset hvor mye gruppen fikk sittet sammen og jobbet med prosjektet. Flere av gruppens medlemmer hadde i perioder en hverdag som ikke egnet seg for å jobbe med agile metoder. Underveis ble det fattet en avgjørelse om at fossefallsmetoden var mer egnet med tanke på omstendighetene. Sprintene ble omgjort til en utviklingsblokk hvor målet var å oppfylle alle punktene i kravspesifikasjonen før overleveringsfristen.

#### 4.3.2 Fossefallsmetoden

I denne utviklingsmetoden er målet å oppfylle alle kravene som stilles av oppdragsgiver. I fossefallsmetoden er det vanlig å ta for seg en oppgave av gangen og ferdigstille den delen av prosjektet før man går videre til neste del. På den måten vil prosjektet i større grad ha en lineær utviklingsprosess.

I figuren til høyre fremstilles stegene i metoden, hvor et steg ferdigstilles før neste igangsettes. I denne prosjektoppgaven ble først

kravspesifikasjonen utviklet, før design og prototyping av applikasjonen ble gjennomført. Utførelsen ble gjort i en kontinuerlig prosess og testing ble gjort mot slutten av prosjektet, før feilretting og overlevering av produktet til Nabobil.



Figur 2 - En fremstilling av fossefallsmetoden

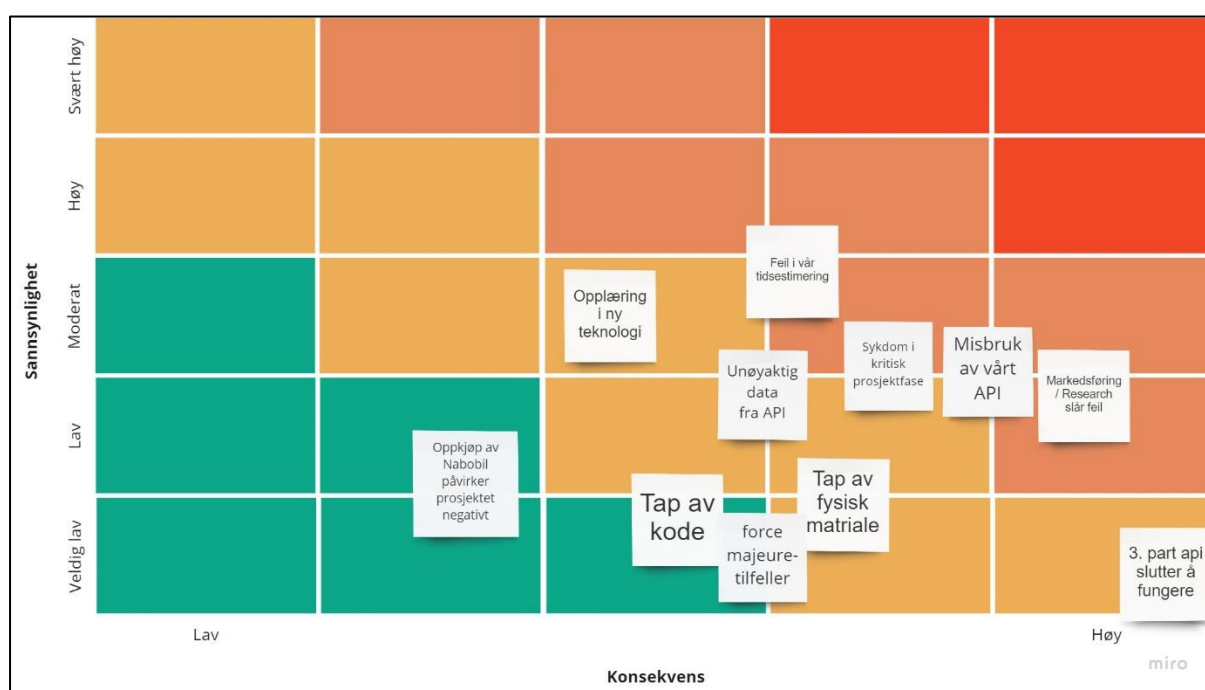
### 4.3.3 Frittstående eller integrert plattform

Et av spørsmålene som ble diskutert tidlig i planleggingsfasen var om webapplikasjonen skulle integreres på nabobil.no eller lanseres som et frittstående produkt. Nabobil som oppdragsgiver ønsket at denne applikasjonen skulle stå på egne ben. Denne avgjørelsen var viktig å fatte tidlig, da det har mye å si for hvordan webapplikasjonen skal se ut. Ved utvikling av et frittstående produkt har gruppen friere tøyler.

### 4.3.4 Risikoanalyse

I slutten av planleggingsfasen ble det utarbeidet en risikoanalyse i realtimeboardet gruppen bruker til å organisere arbeidet, Miro. Analysen ble utviklet fordi det er viktig å kjenne til utfordringer som kan oppstå underveis i prosjektet. Både forhold knyttet til det tekniske prosjektet, samt utenforstående forhold, for eksempel sykdom, ble vurdert.

I figuren under er vurderingene presentert. Y-aksen viser sannsynligheten for at en hendelse inntreffer, og x-aksen hvor stor konsekvens hendelsen vil få for prosjektet. I tabell 4 finnes en mer detaljert utredning av respektive punktene i risikoanalysen.



Figur 3 - En grafisk framstilling av risikoanalysen

Tabell 4 - Risikoanalyse

Hendelse	Sannsynlighet	Konsekvens
Oppkjøp av Nabobil påvirker prosjektet negativt	Nabobil er kjøpt opp av Getaround, men vi har blitt forsikret om at dette ikke kommer til å påvirke vår oppgave.	Prosjektet er basert på en skriftlig avtale, men det kan bli behov for omstruktureringer dersom oppkjøper krever det.

Opplæring i ny teknologi	Det kan tenkes at gruppen ikke mestrer teknologi som er nødvendig å bruke for optimal fremgang. Scopet i oppgaven er relativt lite og Nabobil vil være kjent med teknologien vi skal bruke. Det er likevel en moderat sannsynlighet for at dette kan inntreffe.	Kan forsinke fremgangen i prosjektet og i verste fall medføre at gruppen leverer et produkt som ikke oppfyller samtlige punkter i kravspesifikasjonen.
Tap av kode	Gruppen har versjonshåndtering av alt arbeidet i Github, dermed er sannsynligheten lav for å tape større deler av koden.	Vil som regel ha moderate konsekvenser, men vil avhenge av hvor mye kode som går tapt, og kan potensielt bli alvorlig.
Force majeure-tilfeller	Sannsynligheten for at slike tilfeller inntreffer er meget lav.	Kan i ytterste konsekvens innebære at oppgaven ikke kan fullføres, men mer sannsynlig med en moderat påvirkning på arbeidets fremgang samt leveringsfrist.
Unøyaktig data fra API	Nabobil er ansvarlig for prosjektet og leverer API'ene til prosjektet, derfor er sannsynligheten lav for at kvaliteten er dårlig.	Dersom dataen fra API'ene er unøyaktig, kan det påvirke verdien webapplikasjonen har for Nabobil og prosjektet kan bli skrotet.
Tap av fysisk materiale	Det er lav sannsynlighet for at vi mister fysisk materiale som er kritisk for prosjektet.	Kan føre til forsinkelser i prosjektet, men mer alvorlige konsekvenser motvirkes ved å lagre på en måte som gjør at tilgang på for eksempel kildekode vil være tilgjengelig fra hvilken som helst datamaskin.
Sykdom i kritisk prosjektfase.	Sannsynligheten er lav/moderat for at en av gruppemedlemmene kan bli syk.	Konsekvensen kan være forsinkelser i prosjektet. Gruppen vil sørge for at flere har kjennskap til prosjektets moduler slik at andre i gruppen kan overta arbeidsoppgaver ved sykdom.
Misbruk av vårt API	Det er lav/moderat sannsynlighet for misbruk av vårt API.	Konsekvensen for prosjektet er høy. Misbruk kan føre til betydelige forsinkelser, da feilretting må prioriteres. Det må ytes vederlag for oppslag i API, dette kan få negative økonomiske konsekvenser.
Markedsføring / reach i segmentet slår feil.	Sannsynligheten er lav til moderat for at produktet ikke vil bli populært i markedet. Prosjektet er foreslått av Nabobil,	Konsekvensene kan være store for den tekniske løsningen dersom ingen benytter den. For gruppens oppgave har det relativt lite å si,



	som har mye kjennskap til markedet.	men det er uheldig om Nabobil må legge ned den tekniske løsningen som er utviklet.
3. part API slutter å fungere	API'et er levert av Nabobil som har benyttet dette i lengre tid uten problemer. Sannsynligheten for at det slutter å virke er meget lav.	Dersom API'ene programmet mottar data fra slutter å fungere må prosjektet skrinlegges. For å fullføre prosjektet kan det utarbeides ikke reelle testdata for å presentere konseptet.

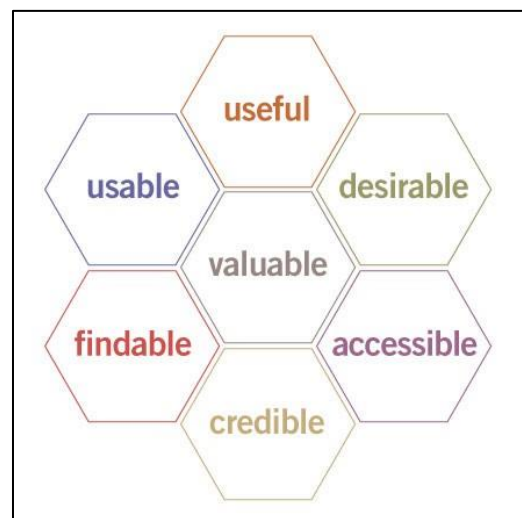
## 5 Design av applikasjon

I løpet av prosjektets første dager ble det designet LOFI-prototyper, som senere ble HIFI-prototyper i AdobeXD. Det ble lagt vekt på at brukeren raskt skal kunne tilegne seg den viktige informasjonen tjenesten vår tilbyr, altså salgspris kontra utleieverdi. Fokuset var videre at webapplikasjonen skal være funksjonell og enkel å bruke, uten for mange forstyrrende elementer. Brukeren skal settes i sentrum og webapplikasjonen skal designes så enkelt at det sjelden vil oppstå brukerfeil.

### 5.1 Generelle hensyn bak design

Når kravspesifikasjonen var avklart, ble en prosess for å designe brukergrensesnittet satt i gang. Brukerens opplevelse i webapplikasjonen er sentral for at tjenesten blir vellykket. Som utgangspunkt er det ønskelig å benytte god praksis fra UX-design, User Experience Design (Interaction Design Foundation, 2020).

UX-design handler om å skape løsninger som er meningsfulle og relevante for brukeren. Brukeropplevelsen skal være sømløs og produktet skal fremstå som verdifullt og godt satt sammen. I UX-design er brukergrensesnittet viktig. UX-design går imidlertid utover dette og ser også på andre elementer, for eksempel behov i markedet, merkevarebygging, funksjonalitet, motivasjon bak, verdiskaping med mer. En utredning av disse elementene må foretas før prototypingen av produktet starter. I UX-design er det vanlig å se på syv prinsipper som man baserer design på, også kalt «The User Experience Honeycomb» (Morville, 2020)



Figur 4 - The User Experience Honeycomb

#### 1. Useful (nyttig)

Det er viktig å spørre seg selv om dette produktet faktisk er nyttig. Det kan ses i sammenheng med spørsmålet «Hvorfor gjør vi det slik?». Dersom svaret er «fordi vi alltid har gjort det slik» er det alltid forbedringspotensial. Som designere må man bruke kreativiteten for å utvikle nye og nyttige produkter som er mer tilpasset

brukerens faktiske behov.

2. *Usable (Brukervennlig)*

Produkter må være brukervennlige. Brukergrensesnittet må være enkelt å bruke og designes ut fra prinsipper om menneske-maskininteraksjon. Produkter med god brukervennlighet har ofte et konkurransefortrinn og øker brukerens komfort. Dette punktet dekker mye av det som er UI-design (User Interface). UI-design er en kritisk del av UX-design, men er ikke alene tilstrekkelig for å lage et godt produkt.

3. *Findable (Navigerbart)*

En tjeneste må være oversiktlig, nettstedet med mye innhold må være enkle å navigere. Dersom brukere ikke klarer å lokalisere det de er ute etter på en enkel måte er sannsynligheten høy for at de avslutter bruk av tjenesten og heller benytter konkurrerende tjenester.

4. *Credible (pålitelig)*

For å oppnå godt UX-design må produktet være pålitelig. Informasjon og innhold på et nettsted må være riktig, tjenesten må fungere når brukeren har behov for den og brukeren må ha tillit til at tjenesten er trygg å ferdes på, spesielt med tanke på brukerens data. For å bygge slik tillit er det viktig å fremstå seriøs, og sørge for at tjenesten leverer det den lover.

5. *Accessible (Tilgjengelighet)*

En tjeneste skal kunne brukes av alle. Brukeren skal ikke begrenses av en eventuell funksjonsnedsettelse. Produktet skal være universelt utformet. I Norge er det lovpålagt å følge enkelte prinsipper regulert i WCAG 2.0-standarden utgitt av Digitaliseringsdirektoratet (Digitaliseringsdirektoratet, 2020). Viktige elementer er bruk av farger og kontrast som alle kan se, tekst alle kan lese, brukerens kontroll over video- og lydavspilling på nettsiden, innholdets rekkefølge osv.

6. *Desirable (Ønskelig)*

Tjenesten må være ønskelig for å lykkes. Dersom en tjeneste ikke oppfyller et behov i markedet, vil den sannsynligvis ikke lykkes. Merkevarerbygging, identitet og omdømme er kritisk for å øke bevisstheten rundt tjenesten, og har betydning for dens suksess.

7. *Valuable (Verdifull)*

Tjenester skal være verdifulle. Dette kan ta form av å dekke informasjonsbehov, presentere ideelle organisasjoner eller på annen måte bidra til økt omsetning i en privat organisasjon. Verdien av en tjeneste er produktet av hvor godt den oppfyller de andre prinsippene i modellen. Dersom tjenesten ivaretar alle prinsippene tilfredsstillende, vil sannsynligheten for at den er verdifull øke.

Gruppen ønsker å benytte teorien om UX-design for å skape et produkt og en tjeneste som vil lykkes. For oppdragsgiveren vil et produkt som er godt designet ha en mye større verdi, derfor er det viktig å skape en løsning som er pålitelig, nyttig og tilgjengelig, men som også ser fin ut og er brukervennlig.

For gruppen er det flere ting som er viktig når det kommer til design av brukergrensesnittet. Brukeren skal bidra med så lite informasjon som mulig. Det skal være tydelig hvor i brukergrensesnittet interaksjon er mulig, det skal være færrest mulig interaksjoner fra bruker, men nok til at man får den informasjonen som er nødvendig. Dette er ønskelig for å begrense antall brukerfeil i webapplikasjonen. Ettersom alle bileiere potensielt er i målgruppen, må webapplikasjonen være robust og kunne brukes av alle.

Universell utforming tas på alvor. Ettersom webapplikasjonen har et relativt lite scope, er det særlig viktig å oppfylle standardene for tilgjengelighet i utviklingen av tjenesten.

For å følge god praksis i webutvikling, designer vi en mobilvennlig versjon først og tar utgangspunkt i den når vi utvikler en egen versjon for desktop og større skjermer. En voksende del av populasjonen benytter i dag smarttelefonen til det meste, da er det viktig å ha en applikasjon som fungerer like godt på mobil som på datamaskin.

En applikasjon som er mobilvennlig vil øke sannsynligheten for at besøkende faktisk benytter tjenesten, og lengden på hvert enkelt besøk vil trolig øke samsvarende. Dersom responsivt design implementeres på en god måte er det enklere å bruke det samme kodegrunnlaget for å utvikle en egen versjon for større skjermer.

Webapplikasjonen gruppen skal utvikle er i samme kategori som mange andre tjenester på nettet, nemlig kalkulatorer. Tilsvarende tjenester kan for eksempel benyttes til å beregne skatt eller hvor mye drivstoff og bompenger bilen vil bruke på en gitt distanse. Et av problemene med slike kalkulatorer er at de ikke alltid har tilgang til alle relevante variabler for å gi nøyaktig informasjon.

For gruppens webapplikasjon innebærer dette at det kan være utfordrende å gi en korrekt verdivurdering. For verdsetting av bil blir dette spesielt utfordrende, da bruktbilmarkedet alltid vil være usikkert. Med begrenset informasjon om bilens egentlige stand er det vanskelig å gi en korrekt verdivurdering. Derfor er det viktig å påpeke ovenfor brukeren at verdiene som presenteres er estimer og det kan variere basert på bilens stand og trender i bruktbilmarkedet.

For å oppnå trygghet og tillit hos brukeren må det være tydelig at verdivurderingene er estimer og kun gir et utgangspunkt for å vurdere om man ønsker å leie ut eller selge bilen. Videre må det komme tydelig frem hva webapplikasjonens formål er. Åpenhet skaper tillit hos brukeren så lenge alle fakta legges på bordet. Derfor er det også viktig å få frem at denne webapplikasjonen er lagd i samarbeid med Nabobil.

## 5.2 Konkurrenter

Det er alltid viktig å se til konkurrenter som allerede er etablert i markedet når en ny tjeneste skal utvikles. Å undersøke hvilke produkter som eksisterer, hvilken funksjon de

tjener og hvor like de er produktet som skal utvikles er essensielt. Dersom dette gjøres på en god måte kan gruppen finne ut hvilke tjenester som mangler i markedet, om det er etterspørsel etter slike tjenester og hvordan eventuell etterspørsel kan møtes.

Et søk etter lignende tjenester viser at det ikke finnes mange, og ingen andre gir salgs- og utleieverdi i samme tjeneste. Dette er en god nyhet for prosjektet. Siden ingen andre webløsninger tilbyr samme tjeneste som vår applikasjon, må man snu blikket til de nærmeste konkurrentene. Dette er tjenester som leverer verdivurdering av bil på nettet ved å oppgi norske bilregistreringsnummer.

Gruppen fant to slike tjenester, Finn.no pristips for bruktbil (Finn.no, 2020) og Kvd Norge (Kvd Norge, 2020)

Verdikalkulatoren til Finn.no gir et estimat på hvor mye bilen er verdt på bruktbilmarkedet basert på registreringsnummer og kilometerstand. Finn.no oppgir at verdivurderingen baserer seg på lignende biler som tidligere er annonsert på Finn.no. Dette kan være en svakhet med tjenesten, da man ikke vil få en verdivurdering dersom få biler av samme type er annonsert tidligere, som vist i figur 5.

Dette er noe som gruppen ønsker å unngå i sitt prosjekt, men som vil avhenge av kvaliteten på data fra API'et Nabobil skal levere.



Figur 5 - Utsnitt fra Finn.no pristips for bruktbil

Finn.no sin løsning er ellers enkel og stilren, som gruppen ønsker å sikte mot i sin løsning. En annen ting Finn.no gjør godt er å forklare hva tjenesten er. De forklarer hvordan de regner ut prisen, hva som inngår i prisens nøyaktighet og forventningene til en bilselger. Dette er noe gruppen også ønsker å se nærmere på i prosjektet.

Kvd sin løsning krever på samme måte kun input av registreringsnummer og kilometerstand for å vurdere verdien på bilen. Denne tjenesten oppgir forventet kjøpspris for en bil enten hos privatperson eller forhandler. Hos Kvd er det mulig å få en utvidet verdivurdering ved å fylle inn mer informasjon om bilen. Dette kan gi en mer nøyaktig verdivurdering, men gjør tjenesten mer komplisert for brukeren. I gruppens prosjekt er det heller ønskelig å holde tjenesten så enkel som mulig, men tydelig informere om at prisen er et estimat og at den kan variere basert på bilens stand ellers. Gruppen vil også vurdere å ta dette med i sitt prosjekt hvis det kan løses på en enkel og brukervennlig måte.

For ordens skyld ønsker gruppen å kommentere Nabobils utleieverdikalkulator (Nabobil.no, 2020). Nabobil har tidligere utviklet en kalkulator som viser utleieverdien til en bil basert på registreringsnummer og postnummer. Vi anser ikke dette som en konkurrent da dette er oppdragsgivers programvare. I gruppens webapplikasjon vil vi hente data fra det samme API'et som denne kalkulatoren. Nabobils kalkulator tar også høyde for utleieverdi med og uten «Nabobil Uten Nøkkel». Som nevnt i del 2.5 faller dette utenfor kravspesifikasjonen.

### 5.3 Lo-fi prototype

Etter redegjørelsen for viktige prinsipper i UX-design ble en planleggingsfase med fokus på prototyping startet. Det benyttet penn og papir for å enkelt utforme og forklare ideer. Å tegne er en kosteffektiv måte å designe på med tanke på hvor tidkrevende det er å utvikle digitale prototyper. Det sparer oss for verdifull tid i denne fasen og gir en klarere retning for den digitale prototypen som senere skal utvikles. Flere faser med idémyldring ble gjennomført for å komme frem til et gunstig design som oppfyller flere av prinsippene i UX-design.

I starten av utviklingsprosessen hadde vi en diskusjon med Nabobil angående mengden innhold på siden. Gruppen la frem et forslag om at Nabobil kunne være ansvarlig for å legge ut innhold på siden om de ønsket det. Tanken var at brukeren skulle kunne lese suksesshistorier fra andre som har registrert bilen sin på nabobil.no eller lignende historier om fordeler ved å leie ut bilen sin. I møtet med Nabobil kom det imidlertid frem at de ønsker at applikasjonen skal fremstå som enkel og at den står støtt på egenhånd. Derfor er designet av applikasjonen kun fokusert på selve funksjonaliteten den tilbyr, annet innhold blir begrenset for å fronte selve kjernen av prosjektet.

For at webapplikasjonen skal bli navigerbar vil siden designes som en single-page application (SPA) (Skólski, 2020). Dette gjør at brukeren enkelt kan finne alt innholdet på en side. Webapplikasjonen er et relativt lite program, derfor er det fordelaktig å designe den som en SPA.

Tidlig i denne fasen var en av de sentrale problemstillingene å finne en god måte å sette utleieverdi opp mot salgsverdi. Ved å lage en container med salgsinfo og en med utleieinfo vil brukeren selv kunne vurdere de to valgene opp mot hverandre. På mobil er dette en utfordring siden skjermens størrelse er begrenset, derfor vil containerne i mobilvisningen plasseres loddrett over hverandre. Ved å prototype på denne måten er det lettere å utvikle en responsiv webapplikasjon, da containerne er like på de to skjermtypene og kun plasseringen av dem i forhold til hverandre varierer. Se tidlige skisser for hvordan brukergrensesnittet skal se ut på stor og liten skjerm vedlagt i figur 6-9 på neste side.

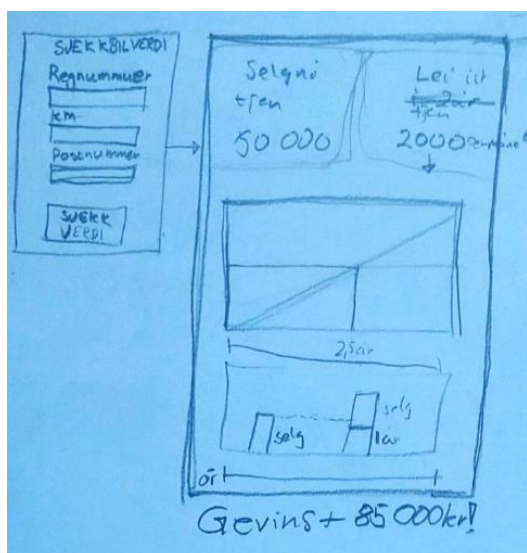
Ut fra kravspesifikasjonen er det flere store komponenter vi skal plassere, dette er:

- Inputfelt for registreringsnummer, kilometerstand, postnummer.
- En container med bilens info og salgspris.
- En container med bilens utleieverdi.
- En grafisk fremstilling av salgs- og utleieverdi.

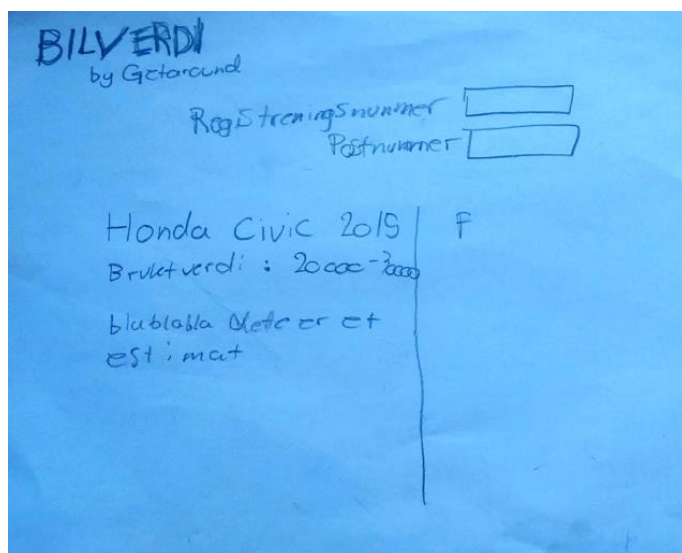
Gruppen startet med prototypen for mobilskjerm. I tegningen planlegger gruppen hvor hovedelementene skal ligge i forhold til hverandre. Det første gruppen ønsker at brukeren skal møte er et tekstfelt hvor informasjon plottes inn. Deretter skal brukeren trykke på en knapp som setter i gang verdivurderingen. Webapplikasjonen vil så vise generell informasjon om bilen i tillegg til salgs og utleieverdi. Dette vil vises i to containere på loddrett linje som vist i Figur 9 - Prototype av mobilversjon. I denne fasen diskuteres ikke tekstlig innhold og hvilke variabler om bilen som vil bli vist i webapplikasjonen. Neste punkt

er å vise en grafisk presentasjon av bilens verdi, som kan ses nederst i Figur 8 - Prototype av mobilversjon.

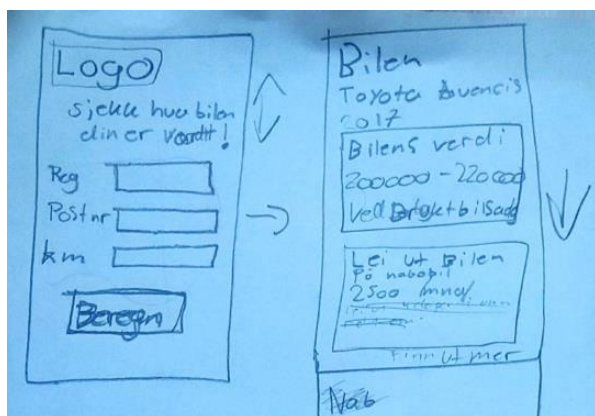
Som utgangspunkt vil vi først designe applikasjonen til mobil, deretter til større skjerm. Allerede i denne fasen lages en enkel illustrasjon for hvordan applikasjonen på større skjerm kan se ut, selv om den i stor grad vil designes med de samme elementene som mobilversjonen.



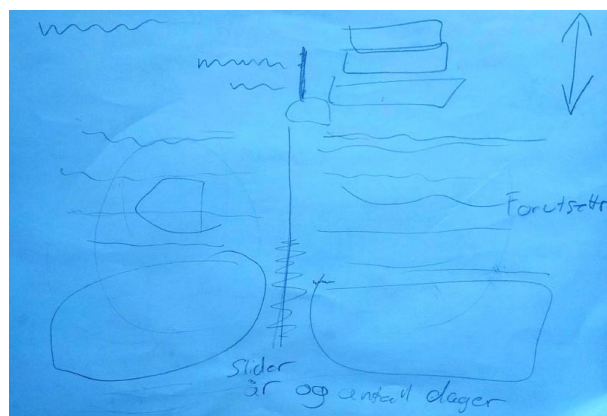
Figur 8 - Prototype av mobilversjon



Figur 6 - Skisse av applikasjonen for stor skjerm



Figur 9 - Prototype av mobilversjon

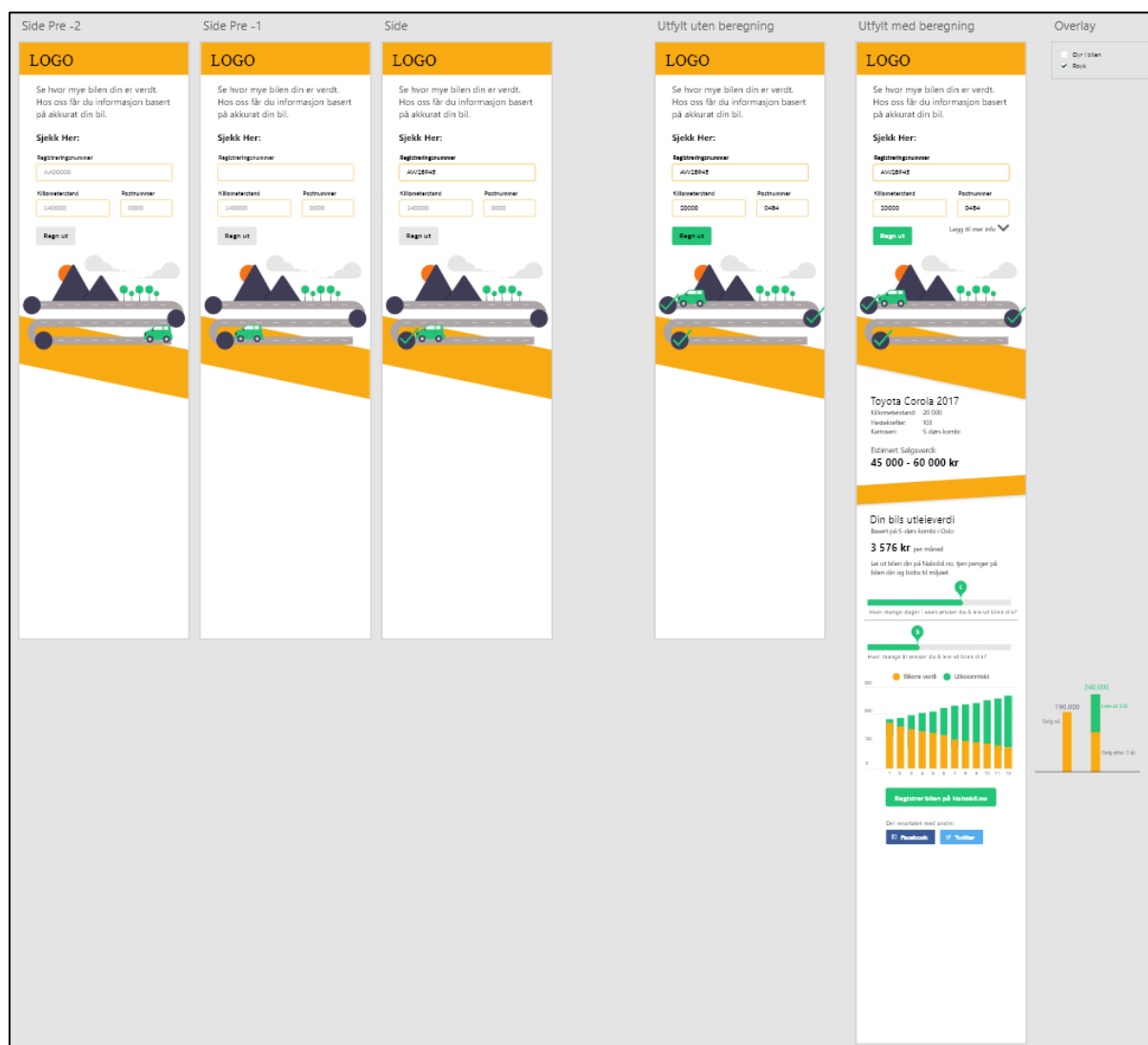


Figur 7 - Skisse av applikasjonen for stor skjerm



## 5.4 Hi-Fi prototype

Basert på prototypen på papir og etterfølgende diskusjon ble det bestemt at en Hi-Fi prototype skulle utvikles. Dette ble utført i Adobe XD.



Figur 10 - Komplette prototype av mobilvisning til applikasjonen

I figuren over er en oversikt over alle visningene en mobilskjerm har. En prototype av høyere kvalitet er tidkrevende å lage, men gir utvikleren en enklere jobb når webapplikasjonen skal utvikles.

Tilsvarende input-felt som i papirprototypene er tatt inn. Bilen som kjører langs veien skal validere brukerens input. Når en input er korrekt validert, vil bilen kjøre 1/3 frem. Når alle feltene er korrekt, vil bilen nå målet og knappen for å søke blir grønn.

Når et søk er gjennomført vil webapplikasjonen gjøre et kall til API'ene og finne informasjonen om bilen, for så å vise den til brukeren. I mobilvisningen får brukeren først informasjon om bilen og dens verdi, lengre ned finner brukeren informasjon om utleie av bilen.

Brukeren kan interagere med webapplikasjonen for å justere hvor mange år bilen skal leies ut og hvor mange dager i uken bilen skal leies ut. Leieinntekten vil automatisk justere seg ettersom brukeren endrer disse verdiene. Verdiene justeres ved å bruke en slider.

Brukeren får deretter visualisert salgs- og utleieverdi i en graf. Dette vil gjøre det enklere å se verdiene i sammenheng med hverandre. Grafene er dynamiske og endrer seg når utleieverdien justeres på siden.

Nederst på denne siden kan brukeren velge å gå videre til nabobil.no for å undersøke bilutleie nærmere. Det legges også til funksjonalitet for å dele verdivurderingen på Facebook.

Webapplikasjonen er designet med brukeren i sentrum, vi ønsker å minske antall brukerfeil ved å lage en applikasjon som er enkel å bruke. Brukeren skal ikke overøses med informasjon, men skal få informasjon som er nødvendig for at de skal føle seg sikre på at tjenesten er reell. Dette er viktig for å bygge tillit hos brukergruppen.

Ved å designe siden som en SPA reduserer vi muligheten til å navigere feil og gjør all informasjon tilgjengelig på den samme siden. Slik vil det ikke være utfordrende for brukere å finne den informasjonen de trenger.

Etter at prototypen til mobil var ferdigstilt ble de samme komponentene brukt til å prototype webapplikasjonen for stor skjerm. Dette kan ses i figuren på neste side.

I begge prototypene er det lagt inn tekst som fyll, men denne er ikke ferdigstilt.

The image shows a mobile app prototype for a car rental service. It features a yellow header with a 'LOGO' placeholder. The main content area is white with a light gray background. It includes a form for entering a license plate (AV28948), a mileage range (20000-40000), and a rental duration (5 days). A green 'Beregn ut' button is present. Below the form is a green bar with a white car icon and a green checkmark. The car details section shows 'Toyota Corola 2017' with a mileage of 20,000, a rental price of 100, and a rental duration of 5 days. The estimated sale value is '45 000 - 60 000 kr'. The 'Din bils utleieverdi' section shows a monthly rental value of '3 576 kr' and a monthly rental price of '100'. Below this is a slider for rental duration (1-12 days) and a bar chart showing the relationship between rental duration and rental price. The chart has two series: 'Bilens verdi' (orange) and 'Utleieinntekt' (green). The rental price increases with duration, while the rental value decreases. At the bottom, there is a green button 'Registrer bilen på Nabobil.no' and social media sharing options for Facebook and Twitter.

LOGO

Se hvor mye bilen din er verdt.  
Hos oss får du informasjon basert på akkurat din bil.

Sjekk Her:

Registreringsnummer  
AV28948

Kilometerstand  
20000

Poettnummer  
0484

Beregn ut

Legg til mer info

Toyota Corola 2017  
Kilometerstand: 20 000  
Hendelseskrav: 100  
Kartotett: 5 dagers kombi

Estimert Salgsverdi:  
45 000 - 60 000 kr

Din bils utleieverdi  
Baset på 5 dagers kombi i Oslo

3 576 kr per måned  
Le ut bilen din på Nabobil.no, ten penger på bilen din og bidra til miljøet.

Hvor mange dager i uken ønsker du å leie ut bilen din?

Hvor mange år ønsker du å leie ut bilen din?


Bilens verdi Utleieinntekt

Registrer bilen på Nabobil.no

Del resultatet med andre:  
Facebook Twitter

Figur 11 - Prototype for mobilvisning




**Bilpriskalkulator.no**


Sjekk hva bilen din er verdt - Helt uten kostnader  
på kun tre enkle steg!

Registreringsnummer

Kilometerstand

Postnummer

Regn ut



### Toyota Corola 2017

Kilometerstand: 20 000  
Hestekrefter: 103  
Karrosseri: 5-dørs kombi  
Farge: Grå

Estimert Salgsverdi:

**45 000 - 60 000 kr**

### Din bils utleieverdi

Basert på 5-dørs kombi i Oslo

Lei ut bilen din på Nabobil.no, tjen penger på bilen din og bidra til miljøet.

Estimert utleieverdi:

**3 576 kr** per måned

Her kan du se hvor mye du kan tjene på å leie ut bilen din!

Du velger antall dager i uken andre kan bruke bilen din.

Så velger du hvor mange år til du tenker å selge bilen, husk at bilen maks kan være 15 år gammel for å kunne leies ut på Nabobil.no

I grafen ser du verditapet på bilen og inntekten på utleie.

5

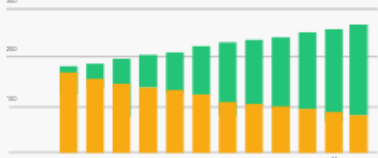
Hvor mange dager i uken ønsker du å leie ut bilen din?

9



Hvor mange år ønsker du å leie ut bilen din?

Bilens verdi

Utleieinntekt



Del resultatet med andre:

 Facebook
 Twitter

Registrer bilen på Nabobil.no

Figur 12 - Prototype av webapplikasjonen for stor skjerm

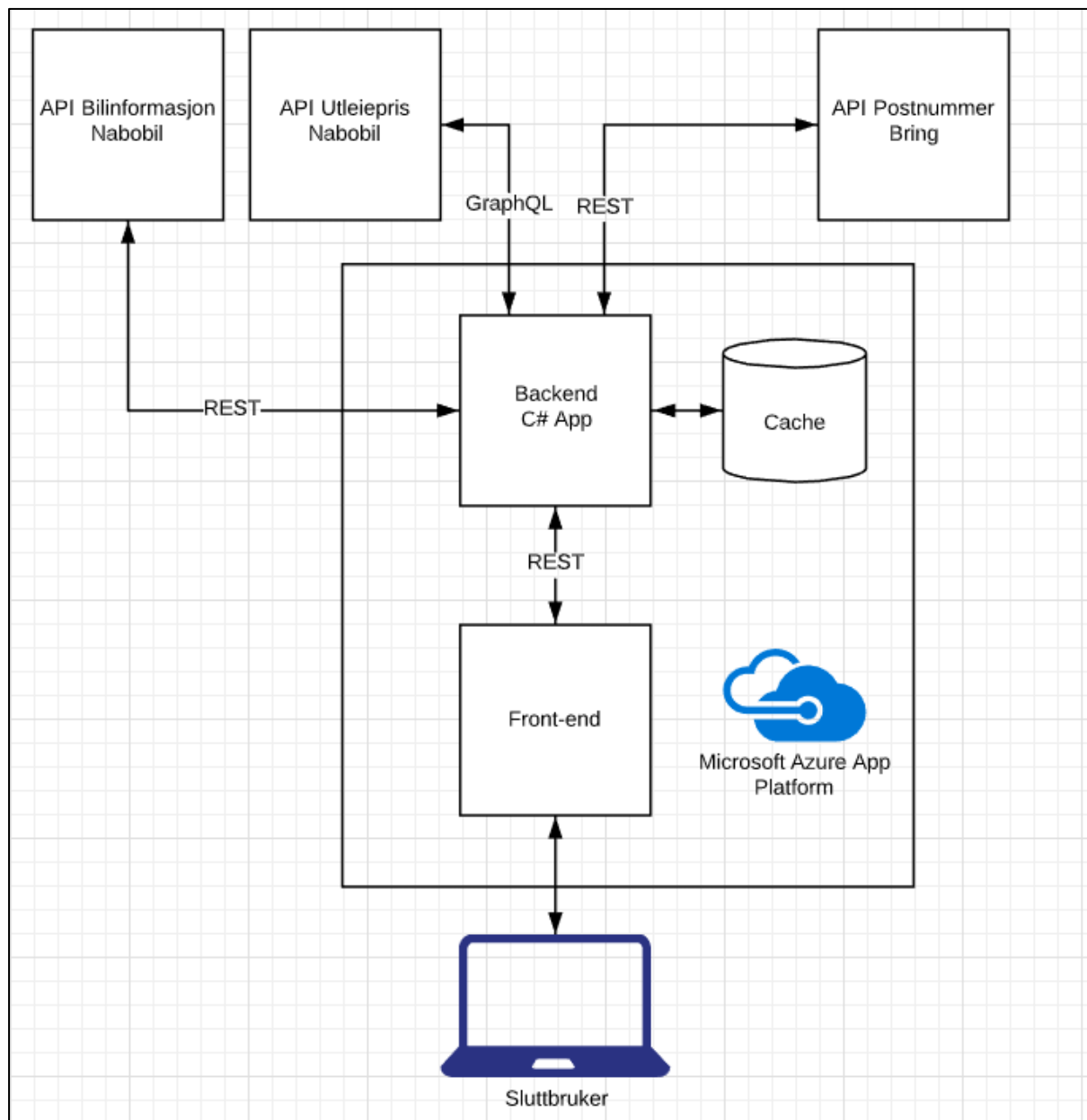
I denne versjonen av prototypen er salgsverdi og utleieverdi satt opp mot hverandre. Ellers bruker prototypen de samme elementene som for mobilvisning, med noen designendringer som er mer tilpasset stor skjerm.

## 6 Produktdokumentasjon

### 6.1 Forord

Produktdokumentasjonen er ment for de som skal installere, vedlikeholde eller modifisere systemet. Formålet er å gi en detaljert og teknisk oversikt over applikasjonen. Det forutsettes at leseren har teknisk kompetanse.

### 6.2 Arkitektur



Figur 13 – Arkitekturdiagram

Figur 13 beskriver systemets arkitektur og oppbygging. Som illustrert over kommuniserer backend med de eksterne API'ene fra Nabobil og Bring. Backend sender så data videre til frontend, som skal presentere disse for brukerne.

### 6.3 Kjøring og hosting av applikasjonen

Frontend bygges med kommandoen *yarn build*. Dette genererer en *build*-mappe som inneholder den transpilerte koden. Denne build-mappen er kopiert til backend sin *wwwroot* mappe som gjør at backend kan servere filene til sluttbrukeren, slik at de får tilgang til nettsiden. *wwwroot* mappen er en konvensjon i .Net Core for å servere selve frontenden.

Siden applikasjonen kjører på Azure Portal, Microsoft sin skytjenesteplattform, kan endringer i applikasjonen gjøres via integrasjon med Git og Github. Produksjonsversjonen for applikasjonen befinner seg i master-branchen på Github. Derfra blir koden og filene distribuert opp til applikasjonen i Azure Portal. Dette er distribueringsprosessen for applikasjonen. For å gjøre dette enklere er det laget et distribueringskript i bash for å automatisere prosessen.

```
#!/bin/bash

echo "Fetching potential new changes from remote repository.."
git pull
echo "Done!.."

echo "Building frontend production build.."
yarn build
echo "Removing old backend/wwwroot/ files..."
rm -R ../backend/wwwroot/
echo "Done!"

echo "Copying new build files to backend/wwwroot/ directory.."
cp -vR ../build/ ../backend/wwwroot/
echo "Done!"

echo "Adding newest transpiled production files to master"
git add ../
git commit -m "Pushing changes to production.."
echo "Pushing to master"
git push
echo "Done!.."

echo "Pushing changes to production server.."
git push azure master
echo "Server updated with newest changes!"
```

Figur 14

Det er også tilegnet en prosess for å gjøre distribueringen til produksjonsversjonen automatisk. Dette er et Python-skript som kjører på et spesifisert intervall og oppdaterer produksjonsversjonen automatisk dersom den finner ut at master-branchen har nye endringer. Hensikten med å automatisere denne prosessen er for å ha en oppdatert produksjonsversjon tilgjengelig, slik at testing og design kan skje parallelt med utviklingen. Dette eliminerer nødvendigheten for å sette opp et lokalt utviklermiljø for å interagere med applikasjonen. Samtidig lot dette oppdragsgiver raskt og kontinuerlig følge utviklingen av produktet.

```
//Runpython
import requests
import json
import subprocess
import time
import os
from datetime import datetime
load_data()

#Scheduled time is in seconds, 3600 seconds = 1 hour
SCHEDULED_TIME = 3600

#credentials for github is located in .env file
GITHUB_USERNAME = os.getenv("GITHUB_USERNAME")
GITHUB_PASSWORD = os.getenv("GITHUB_PASSWORD")

def initialize_newest_commit():
    r = requests.get("https://api.github.com/repos/Nabobil/gsic-met/commits", auth=(GITHUB_USERNAME, GITHUB_PASSWORD))
    data = r.json()

    #JSON data structure: https://developer.github.com/v3/repos/commits/#list-commits-on-a-repository
    return data[0]['commit']['date']

def check_for_commit(last_commit_date):
    r = requests.get("https://api.github.com/repos/Nabobil/gsic-met/commits?since={last_commit_date}", auth=(GITHUB_USERNAME, GITHUB_PASSWORD))
    if (r.status_code == 200):
        time.sleep(SCHEDULED_TIME)
        check_for_commit(last_commit_date)
    data = r.json()
    newest_commit_date = data[0]['commit']['date']
    print("check_for_commit: ", newest_commit_date)

    if (len(r.json()) > 1):
        print("Found a new commit since (last_commit_date):")
        print("Running deploy script")
        subprocess.run(["./deploy.sh"])

    print("Sleeping {SCHEDULED_TIME} seconds and checking for new commits again...")
    time.sleep(SCHEDULED_TIME)
    check_for_commit(newest_commit_date)

def main():
    last_commit_date = initialize_newest_commit()
    print("initialised: ", last_commit_date)
    check_for_commit(last_commit_date)

if __name__ == "__main__":
    main()
```

Figure 15

## 6.4 Samsvar mellom kravspesifikasjon og produkt

Applikasjonen oppfyller samtlige krav i kravspesifikasjonen, unntatt (i) punkt 4, «Informasjon om nøkkelfri teknologi», da dette mangler i Nabobils API, og (ii) punkt 7, «Informasjon om nabobil», da vi ikke har hatt kapasitet til dette. Punkt 7 er som nevnt i kravspesifikasjonen mindre viktig og avhenger av innhold fra Nabobil.

## 6.5 Backend

Backend-delen av systemet innebærer hovedsakelig henting og forvaltning av data. Når en bruker skal til frontend er det backend som sender de nødvendige filene til brukeren. Backend sin oppgave er å hente de relevante dataene som er knyttet til brukeren sitt ønske og sende det tilbake til frontend i et samlet og forutsigbart resultat. Dette gjøres via å kalle på flere programmeringsgrensesnitt (API). I tillegg har backend ansvar for å cache nettsidefilene og dataen slik at brukeren får en rask og dermed mer brukervennlig opplevelse.

### 6.5.1 Verktøy

For backend utvikling ble det bestemt å bruke Microsoft .Net Core. Dette er et rammeverk knyttet opp mot Microsoft sitt robuste programmeringsspråk C#, eller C skarp.

### 6.5.2 .Net Core

Grunnen til at C# og .Net Core rammeverket ble valgt er flere: (i) Rammeverket kan kjøre på både Linux- og Windows-maskiner. Dette skaper større valgmuligheter for hvilken server systemet kan kjøres på, og gir oppdragsgiver mer fleksibilitet med tanke på servermuligheter. (ii) I norsk næringsliv er Microsoft sitt økosystem meget utbredt, som gjør videre vedlikehold av systemet lettere. (iii) .Net Core rammeverket har allerede et modent økosystem som gir oss alle verktøyene fra start, slik at man slipper å benytte tredjeparter. Dette gjør systemet mer sikkert og mer portabel blant annet hvis ett tredjepartsbibliotek mister støtte. (iv) God implementasjon av asynkron funksjonalitet slik at flere programmeringsgrensesnittkall kan gjøres samtidig. Dette øker ytelsen og minsker

brukerens ventetid betraktelig. (v) Integrasjon med Microsoft Azure Portal for å enkelt hoste systemet i skyen for å muliggjøre testing og fasilitere et ordentlig produksjonsmiljø. (vi) Lett integrasjon med minnebasert caching for nettsidefiler og data fra programmeringsgrensesnitt.

### 6.5.3 Microsoft Azure Portal

Azure Portal er skytjenesten til Microsoft. Det er godt integrert med rammeverket .Net Core og gir muligheten for overvåkning og hosting av systemet. Videre gjør skytjenesten det lett å publisere systemet på et domene slik at det kan brukes globalt. Dette minimerer flaskehalsen hvis utviklingen bare skulle skje i et lokalt utviklermiljø.

Azure Portal gjør det mulig å simulere et ordentlig produksjonsmiljø som er brukt i reelle bedrifter rettet mot studieløpet. Dette skaper gode vaner og viktig erfaring som er relevant i arbeidslivet.

### 6.5.4 GraphQL

GraphQL brukes i tilknytning til kommunikasjon med Nabobil sitt programmeringsgrensesnitt. Dette programmeringsgrensesnittet (API) inneholder informasjon om kjøretøyet og salgspris og utleiepris.

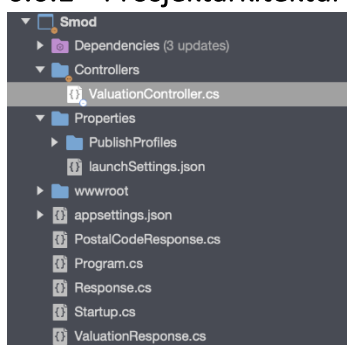
GraphQL er en mer dynamisk måte å utføre kommunikasjon på. I et tradisjonelt REST API så kan ikke klienten bestemme hva slags informasjon den får tilbake. Den sender informasjon til serveren, som bestemmer hva som kommer tilbake. Med GraphQL kan klienten spesifisere til serveren hva slags informasjon den skal ha tilbake. Dette oppnås ved at GraphQL bruker et slags spørringsspråk, slikt som SQL, som gir klienten muligheten til å spesifisere hva serveren skal sende tilbake.

I prinsippet gjør dette at serveren blir mer dynamisk. På den tradisjonelle REST måten må man lage metoder på serveren for hver enkelt brukerhandling. Med GraphQL trenger man i teorien bare å eksponere en metode. Dette gjør kodebasen mer leselig og håndterbar, samtidig som det skaper bedre utviklingsmiljø på klientsiden.

## 6.6 Arkitektur

Det er flere gjeldende arkitekturer på servernivået. Det er prosjektarkitektur og kodearkitektur og arkitektur for raskere feilhåndtering og datahenting.

### 6.6.1 Prosjektarkitektur og kodearkitektur



Figur 16

Prosjektarkitekturen følger hovedsaklig MVC (model-view-controller) arkitekturen. Model er assosiert med data og metoder som anvender dataen. Controller er den som kontrollerer henting og dirigeringen av dataen. Siden serveren er et programmeringsgrensesnitt er det ingen view knyttet til serveren, det løses av klientsiden.

I figur 16 ser man strukturen til prosjektet. ValuationController er hovedpunktet i serveren og det er der logikken med henting

og sending av data skjer. Serveren gjør tre eksterne API-kall. På figur 17 ser vi at metodene i ValuationController er koblet opp mot modellene PostalCodeResponse.cs, ValuationResponse.cs, VehicleInformationResponse.cs og Response.cs. Dette korresponderer med de tre eksterne API-kallene som blir knyttet opp mot PostalCodeResponse, ValuationResponse og VehicleInformationResponse. Dataene fra disse API'ene samles. Det er ValuationController sin oppgave å hente nødvendig informasjon fra andre programmeringsgrensesnitt og populære de relevante klassene med den relevante informasjonen.

Kodearkitekturen følger prinsipper fra det å programmere og strukturere et programmeringsgrensesnitt. I dette systemet er det en metode som er eksponert til internett, som klientsiden kan kalle for å hente den dataen som trengs. Denne metoden er GetResponse i figuren 17. Klientsiden sender inn registreringsnummer (licenseNumber), postnummer (postalCode) og kilometeravstand (mileage) til serveren, og serveren sender en tilbake data knyttet til den informasjonen.

```
[HttpGet("{licenseNumber}/{postalCode}/{mileage}")]
public async Task<ActionResult<Response>> GetResponse(string licenseNumber, string postalCode, int mileage)...

private async Task<ActionResult<PostalCodeResponse>> GetPostCode(string postalCode)...

private async Task<ActionResult<ValuationResponse>> GetValuation(string licenseNumber, string postalCode)...

private async Task<ActionResult<VehicleInformationResponse>> GetVehicleInformation(string licenseNumber, int mileage)...
```

Figur 17

Hovedmetoden GetResponse anvender de tre andre metodene GetPostCode, GetValuation og GetVehicleInformation nedenfor på figuren til datahentingen sin. Dette skaper en ryddig struktur der forskjellige oppgaver er innkapslet i hver sin metode. GetResponse får dataen fra klientsiden. Disse metodene er koblet opp mot hver sin modell, som nevnt ovenfor.

Alle metodene er asynkrone, slik at de kan kjøre samtidig. Hensikten med dette er økt ytelse, noe som bedrer brukeropplevelsen.

```
[HttpGet("{licenseNumber}/{postalCode}/{mileage}")]
public async Task<ActionResult<Response>> GetResponse(string licenseNumber, string postalCode, int mileage)
{
    var cacheKey = licenseNumber.ToUpper() + postalCode + mileage;

    if (memoryCache.TryGetValue(cacheKey, out Response responseFromCache))
    {
        return responseFromCache;
    }

    ActionResult<PostalCodeResponse> postalCodeResponse = await GetPostCode(postalCode);

    if (postalCodeResponse.Value == null)
    {
        return NotFound("Could not find postal code");
    }

    ActionResult<ValuationResponse> valuation = await GetValuation(licenseNumber, postalCode);

    if (valuation.Value == null)
    {
        return NotFound("Could not find license number");
    }

    ActionResult<VehicleInformationResponse> vehicleInformation = await GetVehicleInformation(licenseNumber, mileage);

    if (vehicleInformation.Value == null)
    {
        return NotFound("Could not find license number");
    }

    Response response = new Response(valuation.Value, vehicleInformation.Value, postalCodeResponse.Value);

    var cacheExpirationOptions = new MemoryCacheEntryOptions
    {
        AbsoluteExpiration = DateTime.Now.AddHours(6),
        Priority = CacheItemPriority.Normal,
        SlidingExpiration = TimeSpan.FromMinutes(6)
    };

    memoryCache.Set(cacheKey, response, cacheExpirationOptions);

    return response;
}
```

Figur 18

Kodestrukturen i figuren over er lagt opp slik at API med raskest responstid kalles først. Koden er lagt opp slik at dersom en av de asynkrone metodene ikke får den riktige dataen tilbake, sender serveren en feilmelding til klienten med en gang. Dette skjer uten å avvente til de andre metodene fullføres. Slik kan brukeren få levert feilmeldinger raskere, noe som forbedrer brukeropplevelsen.

## 6.7 Databasen/skytjenesten

### 7.8.1 Caching

Systemet har ingen tilrettelagt database, dette er ikke nødvendig. Ingen konkret database gjør at applikasjonen er enkel, har lite kode å vedlikeholde, og minsker muligheten for feil. I tillegg slipper oppdragsgiver å vedlikeholde samt lagre data. Systemer utnytter heller et av de mange innebygde tjenestene til .Net Core, nemlig IMemoryCache.

IMemoryCache er en lagringsmekanisme som lagrer data i minne slik at systemet lett har tilgang til den om det trengs. Hovedformålet med denne cachen er at, dersom samme forespørsel blir gjort på nytt i et relativt kort intervall (ekspirasjonstiden er satt til 5 minutter), blir dataen sendt til brukeren med en gang. Dette er noe som potensielt kan skape en raskere brukeropplevelse. Samtidig flater dette ut et kritisk punkt, dersom noen velger å dele resultatet sitt med andre utføres ingen ny spørring. Ved å ha denne informasjonen i cache kan resultater deles og oppleves raskt.

Ulempen med å ha cache i minne er at kapasiteten er knyttet til tilgjengelig RAM på maskinen som applikasjonen kjører på. Videre er det vanskelig å finne den optimale ekspirasjonstiden slik at minnet på maskinen ikke opptas for lenge, og at bruker fortsatt får nytte av cachen. Dette er noe som vil bli testet når applikasjonen er i offentlig domene.

Det kan argumenteres for at caching ikke er nødvendig for denne applikasjonen. De forskjellige spørringene brukere kommer til å utføre vil være varierte, da de mest sannsynlig vil utføre spørringer relatert til sin bil, sitt poststed og bilens kilometerstand. Dette betyr at det er mye varierende data. Likevel er muligheten for raskere deling og mindre venting verdt ressursene som IMemoryCache kommer til å trenge. Siden serveren gjør tre forskjellige API-kall per spørring minimeres ressursbruken på denne applikasjonen og på serverene til de eksterne API'ene.

## 6.8 Datastrukturer

Dataene fra de tre eksterne API'ene som applikasjonen bruker er JSON format. Det er også dette formatet serveren bruker for å sende data tilbake til klientsiden. C# og .Net Core gjør det enkelt å assosiere data fra et API med en modell i en MVC struktur. Alt av data fra Nabobil sitt API som omhandler bilinformasjon knyttes opp mot feltene i modellen i Figur 20. Utformingen på denne modellen er knyttet opp mot JSON-dataen applikasjonen får tilbake fra Nabobils API. Dette gjør at håndtering av data blir enkel på serversiden og at databasen får en god rigid struktur. Det samme er implementert i de andre modellene og API-kallene.

```
public class VehicleInformationResponse
{
    public Vehicle vehicle { get; set; }

    public class Vehicle
    {
        public string Colour { get; set; }
        public string Country { get; set; }
        public string FirstRegistered { get; set; }
        public string Fuel { get; set; }
        public string GearType { get; set; }
        public string BodyType { get; set; }
        public string LastRegistered { get; set; }
        public string LastMileage { get; set; }
        public string LicensePlate { get; set; }
        public string Make { get; set; }
        public string Model { get; set; }
        public int RegistrationYear { get; set; }
        public string RegistrationDistrict { get; set; }
        public string VehicleGroup { get; set; }
        public int Seats { get; set; }
        public Dimensions dimensions { get; set; }
        public Engine engine { get; set; }
        public int CurrentValue { get; set; }
        public int FutureValue { get; set; }
        public int Age { get; set; }

        public void CalculateAge()
        {
            Age = DateTime.Now.Year - RegistrationYear;
        }

        public class Dimensions...
        public class Engine...
    }
}
```

Figur 20 - VehicleInformationResponse

```
{
  - vehicleValuation: {
    - earningsCalculation: {
      expectedEarnings: 3415.5,
      rentalDays: 9,
      earningsPerDay: 379.5
    },
  },
  - vehicleInformation: {
    - vehicle: {
      colour: "white",
      country: "NO",
      firstRegistered: "27-10-2015",
      fuel: "gasoline",
      gearType: "automatic",
      bodyType: "station wagon",
      lastRegistered: "07-06-2016",
      lastMileage: null,
      licensePlate: "DP83970",
      make: "Volkswagen",
      model: "Passat",
      registrationYear: 2015,
      registrationDistrict: "N/A",
      vehicleGroup: "personal",
      seats: 5,
    },
    - dimensions: {
      length: 477,
      - weight: {
        gross: 1660,
        includingDriver: 1735,
        maxAllowed: 2250
      },
      width: 183
    },
    - engine: {
      - axles: {
        count: 2,
        driven: 1
      },
      - effect: {
        hp: 214,
        kw: 160
      },
      volume: 1395
    },
    currentValue: 246000,
    futureValue: 113800,
    age: 5
  },
  - vehiclePostalCode: {
    - postalCode: {
      postalcode: "0571",
      city: "OSLO",
      postalCodeType: "NORMAL"
    }
  }
}
```

Figur 19



Datastrukturen til cachen er basert på key -> value par. Data er assosiert med en nøkkel. Nøkkelen i dette tilfellet er registreringsnummer + postnummer + kilometeravstand, for eksempel "DP8397005711200000" -> JSON formatert data. Grunnen til at nøkkelen er formatert slik er at responsen API-serveren kaller på er avhengig av postnummer og kilometerstand og kan gi ulik informasjon. Ved å ha en cache-nøkkel som er knyttet opp mot dette eliminerer vi risikoen for at dataen i cachen ikke korresponderer med spørringen fra brukeren. Dataen i dette tilfellet er den kombinerte dataen fra de tre API-kallene serveren gjør i JSON formatet. Dette gjør det enkelt for serveren å sende dataen tilbake til brukeren dersom dataen allerede er i cachen.

## 6.9 API'er

Applikasjonen bruker tre forskjellige API'er.

- Nabobils utleiepris-API
- Post Brings postkode-API
- Nabobils bilinformasjon- og salgspris-API

Bilinformasjon- og salgspris-API og postkode-API er ordinære REST API, mens utleiepris-API er basert på GraphQL som er nevnt tidligere.

### 6.9.1 Nabobils bilinformasjon- og salgspris-API

Dette er hvor applikasjonen får informasjon knyttet til bilen. Serveren sender registreringsnummer og kilometerstand til Nabobil sitt API og får tilbake informasjonen som vist i figur 21. Nabobil har interne funksjoner for å kalkulere *currentValue* (nåværende salgspris på bilen) og *futureValue* (verdien på bilen når den er 15 år gammel) basert på modell, alder og kilometer kjørt. Dette API er ikke offentlig og er kostnadsbasert. Disse kostnadene dekker Nabobil.

```
{
  - vehicle: {
    colour: "white",
    country: "NO",
    dueForEuApproval: "31-10-2019",
    firstRegistered: "27-10-2015",
    fuel: "gasoline",
    gearType: "automatic",
    bodyType: "station wagon",
    lastEuApproval: null,
    lastRegistered: "07-06-2016",
    lastMileage: null,
    licensePlate: "DP83970",
    make: "Volkswagen",
    model: "Passat",
    registrationYear: 2015,
    registrationDistrict: "N/A",
    vin: "WVWZZZ3CZGE093251",
    vehicleGroup: "personal",
    seats: 5,
    doors: 5,
    + dimensions: {...},
    + engine: {...},
    + tires: {...},
    typeApprovalId: 62061852015,
    currentValue: 246000,
    futureValue: 113800
  }
}
```

Figur 21

### 6.9.2 Nabobils utleiepris-API

API gir applikasjonen dataen relatert til utleieverdi på en bil som leies ut på Nabobil. Utleieprisen er basert på postnummeret bilen befinner seg på og registreringsnummeret til bilen. Det er en intern funksjon hos Nabobil som kalkulerer dataen som applikasjonen har fått tilgang til. Dette er ikke et offentlig API. I denne API-responsen er det kalkulert at bilen vil ha en forventet inntekt på 3415 kr spredt over 9 dager med utleie.

```
- earningsCalculation: {
  expectedEarnings: 3415.5,
  rentalDays: 9,
}
```

Figur 22

Dette API'et tar ikke høyde for sesong og er et estimat basert på faktorer som Nabobil legger til grunn.

### 6.9.3 Post Brings API

Dette API'et er i bruk for å finne ut om postkoden fra spørringen er en korrekt postkode og for å vise brukeren hvilken by dataen er basert på. Dette API'et er for offentlig bruk, men man trenger en autorisasjon for å bruke det i form av en brukernøkkel og en bruker-ID.

```
- postalCode: {  
  postalcode: "0571",  
  city: "OSLO",  
  postalCodeType: "NORMAL"  
}
```

Figur 23

## 6.10 Front-end

Denne seksjonen omhandler presentasjonslaget, delen av systemet som har med presentasjon av data og brukerinteraksjon å gjøre. Dette er en sentral del av applikasjonen hvor det er svært viktig at universell utforming er ivarettatt.

### 6.10.1 Verktøy

Til frontend-utvikling ble det bestemt å benytte React, et JavaScript-bibliotek utviklet av Facebook. Dette er et MIT-lisensiert bibliotek, som betyr at det fritt kan benyttes til kommersiell bruk uten betaling eller attribusjon. Dette biblioteket er brukt av Nabobil i deres hovedprodukt, og gruppe medlemmene har tidligere erfaring med biblioteket.

I tillegg til selve React-biblioteket har det blitt benyttet flere verktøy og biblioteker for å assistere utviklingen.

### 6.10.2 Create React App

Create-React-App er et verktøy som setter opp en React-app med utviklertmiljø. Det inkluderer blant annet støtte for JSX og ES6-syntaks, CSS-autoprefikser, linter og bundler. Koden i et React-prosjekt består av mange filer, og bundleren (Webpack) sørger for å pakke kode i en enkelt JavaScript-fil som inneholder prosjektkode, biblioteker & moduler etc. som brukerens nettleser vil laste.

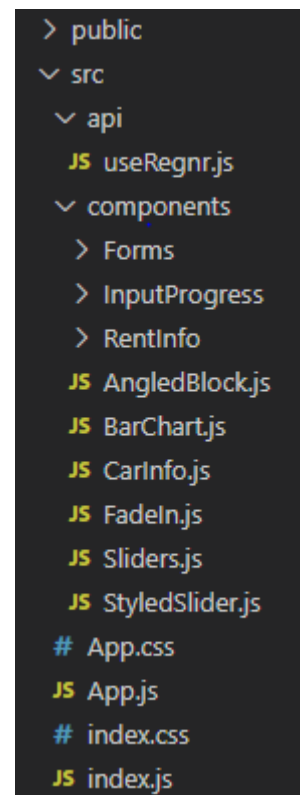
Støtte for JSX og ES6-syntaks gir mulighet til å skrive kode som benytter seg av moderne funksjoner, hvor verktøyet Babel transpilerer koden til en standard som også fungerer i eldre nettlesere.

Lint-verktøyet ESLint er et verktøy som utfører statisk kodeanalyse for å avdekke problemer med koden. Det sørger også for at formattering i koden er konsekvent, noe som er med på å øke kodekvalitet.

Til sammen dekker Create React App punktene som Sommerville beskriver at en utviklingsplattform bør inkludere. (Sommerville, 2011, s. 217)

### 6.10.3 Formik

Formik er et MIT-lisensiert bibliotek som tar for seg håndtering av state, validering og innsending av skjemaer i React.



```
> public  
v src  
  v api  
    JS useRegnr.js  
  v components  
    > Forms  
    > InputProgress  
    > RentInfo  
    JS AngledBlock.js  
    JS BarChart.js  
    JS CarInfo.js  
    JS Fadeln.js  
    JS Sliders.js  
    JS StyledSlider.js  
  # App.css  
  JS App.js  
  # index.css  
  JS index.js
```

Figur 24 - Et React-prosjekt består av mange filer

#### 6.10.4 Material-Ui

Material-Ui er et MIT-lisensiert rammeverk med komponenter for React. Det ble valgt for dette prosjektet fordi dets komponenter både er visuelt tiltalende og laget med god arkitektur. Material-Ui er et omfattende rammeverk, og i dette prosjektet blir kun et subsett brukt, deriblant form-komponenter, layout-komponenter og ikoner.

#### 6.10.5 Styled Components

Styled-Components er et MIT-lisensiert bibliotek for React som muliggjør bruk av vanlig CSS direkte i JavaScript-koden. Biblioteket sørger for å generere unike CSS-klassenavn ved bygging av prosjektet, slik at CSS-konflikter ikke oppstår. Samtidig øker det utviklerkomforten ved å gjøre det svært enkelt å injisere spesifikke stiler på et element ved hjelp av JavaScript.

```
export default styled(TextField)`  
  width: ${props => props.width};  
`;
```

Figur 25 - En styled-component

#### 6.10.6 Chart.js

Chart.js er et MIT-lisensiert bibliotek for å skape interaktive visualiseringsløsninger. Biblioteket er enkelt å bruke og godt egnet for visualiseringene i applikasjonen.

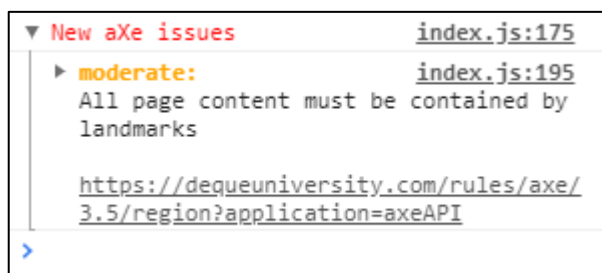
#### 6.10.7 Axe-Core/React-Axe

React-Axe er et MPL 2.0-lisensiert verktøy for testing av universell utforming. Baserer seg på WCAG.

#### 6.10.8 i18next/react-i18next

i18next er et MIT-lisensiert bibliotek for internasjonalisering av JavaScript & React-applikasjoner. react-i18next er den MIT-lisensierte React-integrasjonen av samme bibliotek. I kravspesifikasjonen heter det at applikasjonen skal være på norsk og engelsk.

Ved bruk av i18next trekker man ut all tekst fra komponentene og erstatter det med en funksjon fra i18next-biblioteket som henter ut tekst fra et JSON-objekt hvor applikasjonens innhold blir lagret med en nøkkel-verdi-struktur. Slik kan innholdet oversettes til andre språk uten at det for eksempel er nødvendig å duplisere applikasjonen for forskjellige språk.



Figur 26 - Eksempel på en Axe-issue

### 6.11 Universell utforming i front end design

Applikasjonen er utviklet for å være brukervennlig og universelt utformet. God brukervennlighet medfører blant annet økte markedsandeler, reduserte utviklingskostnader, økt produktivitet og reduserte supportkostnader. (Snadnes, 2011, s. 19)

FNs definisjon av universell utforming:

Universell utforming: Utforming av produkter, omgivelser, programmer og tjenester på en slik måte at de kan brukes av alle mennesker, i så stor utstrekning som mulig, uten behov for tilpassing og en spesiell utforming. Universell utforming skal ikke utelukke

hjelpemidler for bestemte grupper av mennesker med nedsatt funksjonsevne når det er behov for det. (Funksjonshemmedes Fellesorganisasjon, 2020)

Implementering av et universelt utformet system skjer i presentasjonslaget. Det har gjennomgående vært fokus på at applikasjonen skal være universelt utformet. Det er lovpålagt å følge WCAG-standardten:

Nettløsninger skal minst utformes i samsvar med standard Web Content Accessibility Guidelines 2.0 (WCAG 2.0)/NS/ISO/IEC 40500:2012, på nivå A og AA med unntak for suksesskriteriene 1.2.3, 1.2.4 og 1.2.5, eller tilsvarende denne standard. (Digitaliseringsdirektoratet, 2020)

I denne applikasjonen er verktøyet React-Axe benyttet, med et regelsett som varsler dersom WCAG 2.0 A & AA, WCAG 2.1 A & AA eller WCAG Best Practices blir brutt. Sammen med ESLint hjelper disse verktøyene utvikleren å skrive semantisk korrekt, tilgjengelig kode.

## 6.12 Tekniske løsninger front-end

### 6.12.1 Arkitektur & datastruktur

React baserer seg på en enveis dataflyt, hvor man ved hjelp av *props* sender data nedover i komponenthierarkiet. I denne applikasjonen blir *Context* benyttet. Context er et React API som gjør det mulig å dele samme data med mange komponenter uten å eksplisitt måtte sende *props* ned komponenttreet. (Facebook Inc, n.d.)

Applikasjonen har to *Providers*, som er komponentene som gjør kontekstdataen tilgjengelig for underkomponentene.

- ThemeProvider
  - Gjør theme variabel tilgjengelig i komponenttreet
- CarContext.Provider
  - Gjør API-resultatet med kjøretøysdata tilgjengelig for komponenttreet

Resten av frontends tilstandsdata er definert i *App.js* og blir ved hjelp av *props* sendt nedover hierarkiet.

```
const [step, setStep] = useState(0);
const [getVehicle, data, isLoading, error] = useApi();
const [daysInWeekToRent, setDaysInWeekToRent] = useState(3);
const [yearsToRent, setYearsToRent] = useState(3);
```

Figur 27 – applikasjonens tilstandsdata



Figur 28 – Komponenthierarkiet

## 6.12.2 Datahenting

Datahenting skjer i en *React hook*, *useApi.js*. Denne funksjonen henter ut spørringsparametere fra URL (hvis disse er tilstede), og utfører et en spørring mot API'et ved bruk av JavaScript *fetch()*. Etter henting skapes et bil-objekt fra funksjonen *createCar()*, og spørringsparametre settes i URL slik at lenken med resultatet kan deles.

```

import { useState, useEffect } from "react";
import createCar from "../utils/createCar";

const useRegnr = (...params) => {
  const [data, setData] = useState(null);
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);

  const queryParams = new URLSearchParams(window.location.search);

  useEffect(() => {
    if (
      queryParams.has("registration") &&
      queryParams.has("postalcode") &&
      queryParams.has("mileage")
    ) {
      fetchData(
        queryParams.get("registration"),
        queryParams.get("postalcode"),
        queryParams.get("mileage")
      );
    }

    //eslint-disable-next-line
  }, []);

  const fetchData = async (regnr, postnr, mileage) => {
    setIsLoading(true);
    setError(null);
    try {
      const url = `https://bilpris.azurewebsites.net/valuation/${regnr}/${postnr}/${mileage}`;
      const response = await fetch(url);

      if (!response.ok) {
        const errorText = await response.text();
        setError(errorText);
        throw Error(errorText);
      }

      const json = await response.json();

      const car = createCar({json, regnr, mileage, postnr});

      setData(car);

      setIsLoading(false);

      queryParams.set("registration", regnr);
      queryParams.set("mileage", mileage);
      queryParams.set("postalcode", postnr);

      window.history.replaceState(null, null, "?" + queryParams.toString());
    } catch (e) {
      setIsLoading(false);
    }
  };

  return [fetchData, data, isLoading, error];
};

export default useRegnr;

```

Figur 29 – useApi()

Overnevnte *createCar()* former om API-responsen slik at aktuell data er lett tilgjengelig.

```
export default ({json, regnr, mileage, postnr}) => {
  if (json && regnr && mileage && postnr) return {
    age: json.vehicleInformation.vehicle.age,
    bodyType: json.vehicleInformation.vehicle.bodyType,
    colour: json.vehicleInformation.vehicle.colour,
    currentValue: json.vehicleInformation.vehicle.currentValue,
    horsepower: json.vehicleInformation.vehicle.engine.effect.hp,
    firstRegistered: json.vehicleInformation.vehicle.firstRegistered,
    fuel: json.vehicleInformation.vehicle.fuel,
    futureValue: json.vehicleInformation.vehicle.futureValue,
    make: json.vehicleInformation.vehicle.make,
    model: json.vehicleInformation.vehicle.model,
    registrationYear: json.vehicleInformation.vehicle.registrationYear,
    seats: json.vehicleInformation.vehicle.seats,
    city: json.vehiclePostalCode.postalCode.city,
    postalCode: postnr,
    earningsPerDay: json.vehicleValuation.earningsCalculation.earningsPerDay,
    registration: regnr,
    mileage: mileage,
    isOlderThan15Years: new Date().getFullYear() - json.vehicleInformation.vehicle.registrationYear >= 15
  }
  throw new Error('createCar was called with missing arguments');
}
```

Figur 30 – createCar(), en factory-funksjon

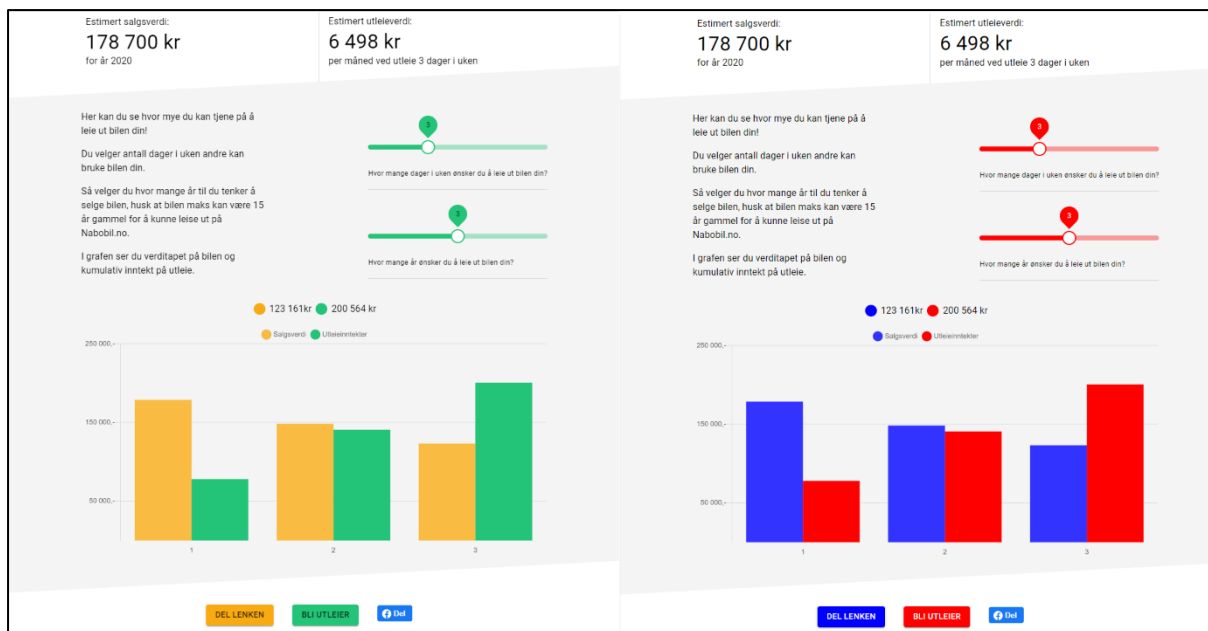
### 6.12.3 Theming og styling

Applikasjonen er bygget opp med to hovedfarger som defineres i konstanten *theme* i App.js.

Disse hentes dynamisk slik at det er svært enkelt å endre applikasjonens fargesammensetning.

```
const theme = createMuiTheme({
  palette: {
    primary: {
      main: "#23C477",
    },
    secondary: {
      main: "#F8AA13",
    },
  },
});
```

Figur 31



Figur 32 – endring av farger i hele appen ved endring i theme-objekt

Videre er resten av sidens styling utført som inline styles der det er egnet, for elementer som ikke blir gjenbrukt eller elementer som ikke trenger media-queries eller annen logikk.

```

{ /* Colored line under header */ }
<div style={{ width: '100%', height: '25px', background: theme.palette.secondary.main }} />

```

Figur 33 – eksempel på element som er stylet inline

Layoutkomponenter er lagret med *Container*-postfix i mappen *StyledContainers* i front-endens grunnmappe. For disse er det brukt styled-components.

Noen av komponentene benytter seg også av Material-UIS funksjon *makeStyles*. Dette er brukt for layout internt i komponenter og brukes for å enkelt ha tilgang på overnevnte *theme*-variabel dypere i komponenthierarkiet.

```

export const CarRentContainer = styled.section`
  @media (min-width: 768px) {
    margin: 0 auto;
    width: 800px;
    display: flex;
    padding: 24px 0px 0 16px;
  }

  @media (max-width: 768px) {
    padding: 0 0 3em 0;
  }
;

```

Figur 34



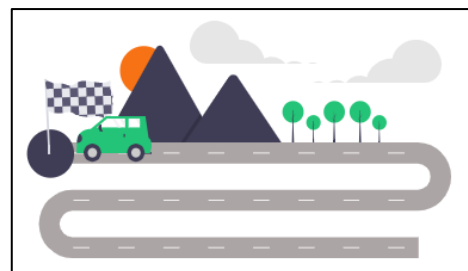
```
const useStyles = makeStyles((theme) => ({
  root: {
    padding: theme.spacing(0, 0, 3, 0),
    width: "100%",
    display: "flex",
    flexDirection: "column",
  },
  table: {
    fontFamily: theme.typography.fontFamily,
    borderCollapse: "collapse",
    "& tr": {
      borderBottom: "1px solid rgba(0,0,0,0.1)",
    },
    "& tr:last-child": {
      borderBottom: "none",
    },
    "& tr td": {
      padding: theme.spacing(1),
    },
  },
  grow: {
    flexGrow: 1,
  },
}));
```

Figur 35

#### 6.12.4 Animasjon

Applikasjonen viser en animasjon av en bil som kjører på en vei etter hvert som inputfeltene blir utfylt og validert. Grafikkelementene består av SVG-elementer hentet fra unDraw (Limpitsouni, 2020), som kan brukes kommersielt uten kostnad og uten attribusjon.

Ved validering i inputfeltene i applikasjonens inputfelt blir *animation*-attributten for elementet endret, som gjør at animasjonen blir utløst.



Figur 36



Figur 37

```

animation: ${props => {
  if (props.step === 0 && props.isReversing) {
    return css`
      ${firstAnimationReverse} 0.75s ease-in-out forwards
    `;
  } else if (props.step === 1 && props.isReversing) {
    return css`
      ${secondAnimationReverse} 1.5s cubic-bezier(.48,.17,0,.45) forwards
    `;
  } else if (props.step === 2 && props.isReversing) {
    return css`
      ${thirdAnimationReverse} 1.5s cubic-bezier(.48,.17,0,.45) forwards
    `;
  } else if (props.step === 1) {
    return css`
      ${firstAnimation} 0.75s ease-in-out forwards
    `;
  } else if (props.step === 2) {
    return css`
      ${secondAnimation} 1.5s ease-in-out forwards
    `;
  } else if (props.step === 3) {
    return css`
      ${thirdAnimation} 1.5s ease-in-out forwards
    `;
  }
}};
;

```

Figur 38

### 6.12.5 Forms og validering

For håndtering av form-tilstand og validering er *formik* og *yup* brukt. Formen er bygget opp som en vanlig HTML-form, hvor blant annet Material-Uis *TextField* komponent erstatter HTMLs *input*-komponent. Når noen av feltene i formen endres blir innholdet validert etter innholdet i *validationSchema.js*.

```

export default Yup.object({
  Registreringsnummer: Yup.string()
    .required("validation:licenseRequired")
    .min(2, "validation:min2")
    .max(7, 'validation:max7'),
  Kilometerstand: Yup.number()
    .min(0, "validation:mustBePositive")
    .max(420000, "validation:maxMileage")
    .typeError("validation:mileageNumber")
    .required("validation:mileageRequired"),
  Postnummer: Yup.string()
    .required("validation:postalcodeRequired")
    .matches("^[0-9]{4}$", "validation:max4")
});

```

Figur 39

### 6.12.6 Bytting av språk

For internasjonalisering blir react-i18next brukt. Konfigurasjonen for dette ligger i filen *i18n.js*. Her importeres språkfilene som utgjør det tekstlige innholdet. Disse ligger under *./src/locales/*.

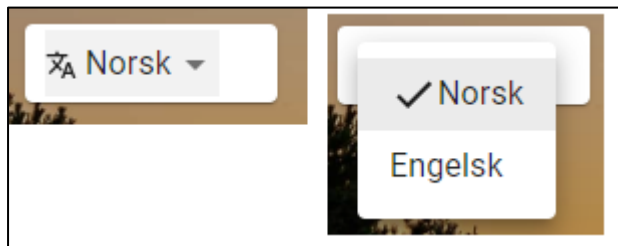
```
header: {  
  headLine: "Selge eller leie ut bilen?",  
  p1: "Økonomi og miljø kan gå hånd i hånd. Ved å leie ut bilen reduserer du  
  p2: "På denne siden kan du få et prisestimat på hvor mye du kan tjene på å
```

Figur 40 – utdrag fra norsk lokaliseringsfil

Teksten fra disse filene blir lastet inn i komponentene ved hjelp av react-i18next funksjonen *useTranslation()*. Se eksempel på bruk av denne i figur 41.

```
<h1> {t('header:headLine')} </h1>  
<p> {t('header:p1')} </p>  
<p> {t('header:p2')} </p>
```

Figur 41



Figur 42 – Dropdown for valg av språk

Når sluttbrukeren endrer språk vil innholdet fra samsvarende lokaliseringsfil bli lastet inn og gjengitt i applikasjonen.

### 6.12.7 Søkemotoroptimalisering

Det er gjort noen grep for å bedre søkemotoroptimalisering. Applikasjonen har fått definert *meta-tags*, som er HTML-elementer som representerer applikasjonens metadata. Her er beskrivelse og keywords definert. Dette gjør det enklere for søkemotorer å indeksere siden.

I tillegg til det blitt utfylt Facebook-spesifikk metadata, som Facebook bruker for å hente data om siden ved deling. (Facebook Inc, n.d.).

```
<meta property="og:url" content="https://salgellerutleie.no" />  
<meta property="og:title" content="Selge eller leie ut bilen?" />  
<meta property="og:description"  
  content="Økonomi og miljø kan gå hånd i hånd. Se hva du kan tjene på å leie ut bilen din." />  
<meta property="fb:app_id" content="132579716841917" />
```

Figur 43

Forsidebildet, overskrift og åpningsparagraf blir også først gjengitt i selve HTML-dokumentet, utenfor React. Hensikten med dette er å vise innhold uten at JavaScript må være aktivert. Dette gjør det lettere for søkemotorer å indeksere siden og innhold.

Dersom en sluttbruker endrer språk i applikasjonen vil elementet skjules (ved at display-attributten for elementet settes til «none»), for så å gjengis av React. Dette gjøres for at react-i18next skal kunne aksessere innholdet.

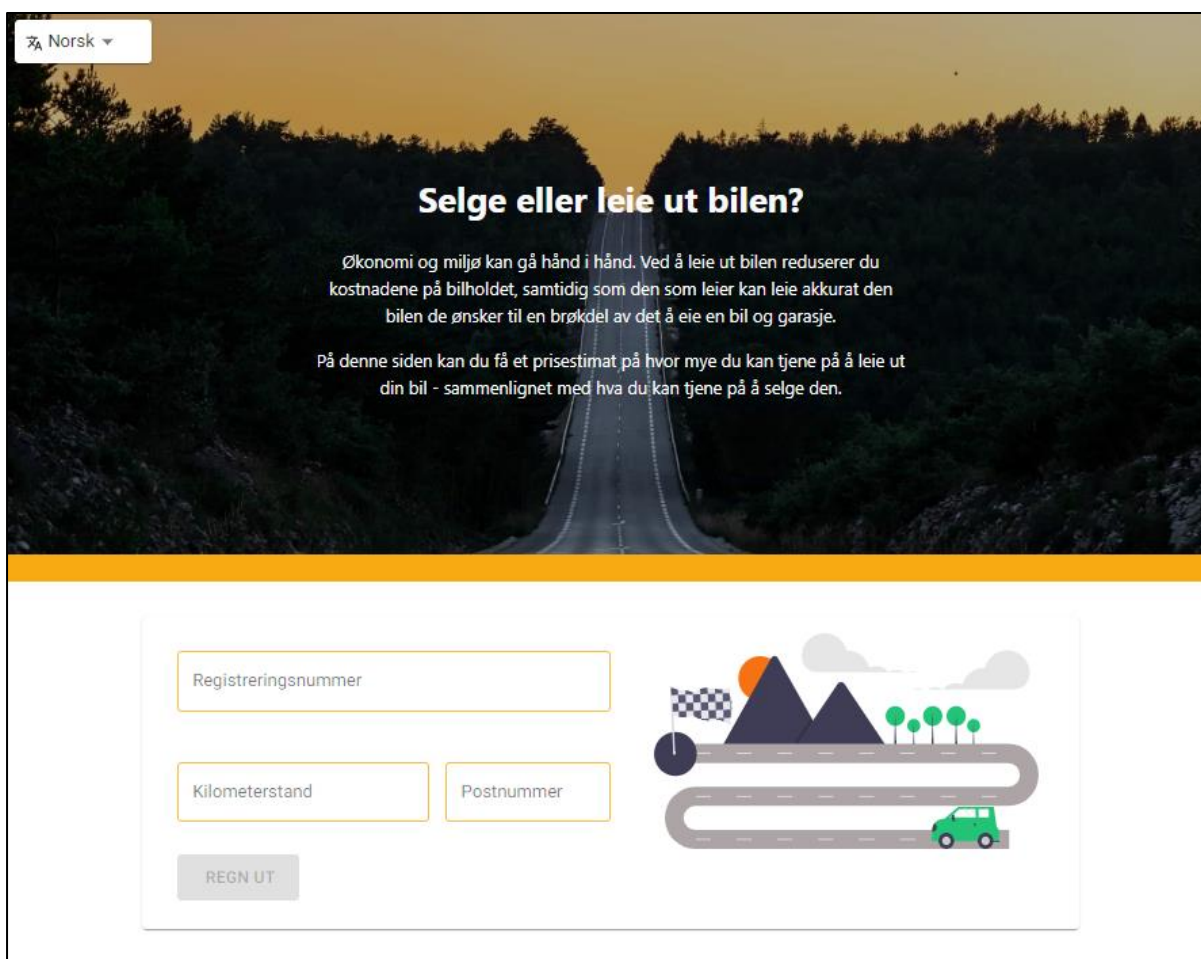
## 7 Løsningen

### 7.1 Innledning

I denne delen av oppgaven skal løsningen sluttbrukeren interagerer med presenteres. Løsningen, som er lagd som en Single Page Application (SPA), er produktet av kildekoden som er presentert i dokumentasjonen. Alle funksjoner og hvordan brukeren interagerer med dem blir presentert i denne delen.

Som nevnt tidligere vil samme kildekode bli brukt for å lage en versjon tilpasset mobilskjerm og en til større skjerm. I prinsippet er begge applikasjonene like og har samme funksjonalitet, men stylingen er tilpasset for å gjøre applikasjonen responsiv. Denne gjennomgangen vil derfor ta for seg versjonen på større skjerm, etterfulgt av en mindre gjennomgang av mobilversjonen.

### 7.2 Versjon for stor skjerm



Norsk ▾

## Selge eller leie ut bilen?

Økonomi og miljø kan gå hånd i hånd. Ved å leie ut bilen reduserer du kostnadene på bilholdet, samtidig som den som leier kan leie akkurat den bilen de ønsker til en brøkdel av det å eie en bil og garasje.

På denne siden kan du få et prisestimat på hvor mye du kan tjene på å leie ut din bil - sammenlignet med hva du kan tjene på å selge den.

Registreringsnummer

Kilometerstand Postnummer

REGN UT

Figur 44 - Brukerens første møte med nettsiden

Dette er brukerens første møte med webapplikasjonen. Det er gjort noen endringer fra prototypen som er presentert i del 5.4 Hi-Fi prototype. Den største endringen er at en banner er lagt til på toppen av siden, sammen med en tekst som forklarer tjenesten. Denne endringen ble gjort fordi nettsiden fremsto tom med kun de tre inputfeltene. Nettsiden fremsto også useriøs og lite troverdig uten mer innhold på forsiden.

Det er også lagt til en knapp for å endre språket til engelsk. Ikonet som er brukt er et standardikon.

Inputfeltene for registreringsnummer, kilometerstand og postnummer er relativt like som i prototypen. Knappen «Regn ut» er i dette tilfellet grå fordi riktig informasjon ikke er tilført inputfeltene.

Figur 45 - Validering på inputfelter

I figuren over vises valideringen på inputfeltene. I dette tilfellet er det skrevet inn informasjon i feltene som er fjernet. Da blir brukeren informert om at den må fylle inn informasjon i feltet. På feltene er det flere valideringer som må godkjennes for at brukeren skal kunne gjennomføre verdivurderingen, dette sikrer at brukeren fyller inn riktig informasjon i feltene. Som en ekstra sikring er det en begrensning på antall tegn i feltene, når maks antall tegn er nådd vil ikke feltet godta flere tegn. På samme format som vist i figuren over er det flere valideringer på de forskjellige feltene:

- Registreringsnummer, maks antall tegn 7:
  - «Vennligst fyll ut registreringsnummer»
  - «Fyll inn minst to tegn»
  - «Registreringsnummer kan kun bruke tall og bokstaver»
- Kilometerstand, maks antall tegn 6:
  - «Vennligst fyll ut kilometerstand»
  - «Kilometerstand må være et tall»
  - «Kilometerstand må være under 420 000 for å bruke denne siden»
- Postnummer, maks antall tegn 4:
  - «Postnummer må være fire tall»
  - «Vennligst fyll ut postnummer»

Når et felt registrerer en feil vil den aktuelle feilmeldingen vises under feltet som inneholder feilen og brukerens input blir endret til fargen rød for å signalisere at noe er galt.

Til høyre for inputfeltene er en figur. Denne henger sammen med valideringen og skal gi brukeren en morsom tilbakemelding på at inputen er godkjent. Bilen vil bevege seg 1/3 frem for hvert felt som blir validert riktig. I figuren til høyre har brukeren gitt riktig input i et felt, dermed har bilen kjørt frem 1/3. Når bilen når målflagget er alle feltene validert og knappen «Regn ut» vil bli grønn.



Figur 46 - Animasjon tilknyttet validering

I noen tilfeller vil inputene oppfylle alle kravene for godkjent validering, men det er ingen registrerte biler med registreringsnummer som er oppgitt. I dette tilfellet vil det dukke opp en tilbakemelding til brukeren om at det ikke er oppslag på dette registreringsnummeret og at de må prøve igjen.

Figur 47- Feil registreringsnummer som input

Når knappen «Regn ut» trykkes på vil knappen gjøres om til et ikon som viser at nettsiden laster. Det er viktig å gi brukeren en tilbakemelding på at nettsiden laster, slik at de må vente. Når nettsiden er ferdig lastet vil knappen endre seg tilbake til den originale slik at man kan søke etter en ny bil.



Figur 48 – Lasteikon

Når alle feltene er validert riktig og registreringsnummer brukeren oppgir finnes i bilregisteret vil resten av webapplikasjonen dukke opp, med informasjon fra bilen som tilknyttet registreringsnummeret. Dette kan ses i Figur 49. Til venstre på siden er informasjon om bilen, dette gjør at brukeren kan bekrefte at det er den riktige bilen. Under informasjon om bilen er det oppgitt estimert salgsverdi. Som nevnt tidligere i oppgaven er det viktig at webapplikasjonen hele tiden tar forbehold om nøyaktigheten av bilers verdi. På samme måte oppgis det at verddivurderingen gjelder for inneværende år. Bilverdi synker betraktelig for hvert år som går, derfor er det viktig å spesifisere dette.

Til høyre får brukeren informasjon om bilens utleieverdi. Først er det en del generell informasjon om Nabobil, faktisk inntekt og utleie. Dette er for å gjøre bruker oppmerksom på at det er usikkerhet forbundet med utleie av bil, for eksempel dersom etterspørselen synker drastisk. Tilsvarende som for bilens verdi spesifiseres det at dette er en estimert inntekt per måned.

Registreringsnummer

EK33181

Kilometerstand

20000

Postnummer

0484

REGN UT

### Peugeot Ion 2017

Karosseri:	Flerbruksbil
Antall seter:	4
Farge:	Hvit
Drivstoff:	Elektrisitet

Estimert salgsverdi:

**100 800 kr**

for år 2020

### Din bils utleieverdi

Basert på 4-seters flerbruksbil i Oslo

Lei ut bilen din på Nabobil.no, tjen penger på bilen din og bidra til miljøet.

Inntekten per dag er basert på utleie gjennomført på tilsvarende biler (alder og biltype) i området hvor bilen din står.

Faktisk pris og utleievarighet vil variere basert på etterspørsel.

Estimert utleieverdi:

**3 546 kr**

per måned ved utleie 3 dager i uken

Figur 51 - Etter at en bil er funnet av Api'et

Ifølge utleievilkårene til Nabobil.no er det ikke mulig å leie ut en bil som har kjørt mer en 300 000 kilometer. Webapplikasjonen må derfor varsle om dette. Selv om bilen har kjørt for langt for å leies ut kan brukeren fortsatt være interessert i bilens salgsverdi.

For å informere brukeren om utleievilkårene vil de få to varsler. Den første er under knappen «Regn ut», før oppslaget på bilen gjennomføres. Den andre er under seksjonen «Din bils utleieverdi», etter at oppslaget på bilen gjennomføres. Dette kan ses i figuren til høyre. I disse tilfellene vil resten av utleieteksten bli grå.

Tilsvarende funksjonalitet er lagt inn om bilen som gjøres oppslag på er mer enn 15

REGN UT

Obs! Når bilen når en kilometerstand på 300 000 kan den ikke lenger leies ut på Nabobil.

Figur 50 - For høy kilometerstand

tilsvarende biler (alder og biltype) i området hvor bilen din står.

Faktisk pris og utleievarighet vil variere basert på etterspørsel.

**Obs! Bilen du har søkt på har en kilometerstand på over 300 000 og kan derfor ikke leies ut hos nabobil.**

Estimert utleieverdi:

**3 546 kr**

per måned ved utleie 3 dager i uken

Figur 49 - Bil som har kjørt for langt til å leies ut



år gammel, som er aldersgrensen for bil som leies ut på nabobil.no.



Figur 52 - Resterende del av webapplikasjonen

Etter informasjonen om salgs- og utleieverdi kommer den grafiske delen av applikasjonen som brukeren kan interagere med. Først kan brukeren lese en forklarende tekst om den

neste delen av applikasjonen. Denne delen av applikasjonen avviker er stort sett tilsvarende prototypen, sett bort fra mindre endringer.

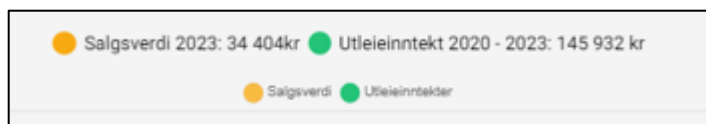
Til høyre for den forklarende teksten er det to slidere. Disse er lagd slik at brukeren kan justere utleieverdien på sin bil. Først kan brukeren velge hvor mange dager i uken bilen kan leies ut. Når antall dager endres vil utleieverdien og grafen automatisk justeres. Som standard er denne slideren satt til tre dager i uken, dette er ifølge Nabobil gjennomsnittlig antall utleiedager i uken for biler som leies ut på deres plattform.

Brukeren kan også velge hvor mange år frem i tid de ønsker å leie ut bil. Det er lagt inn tre år frem i tid som standard. Gruppen, sammen med Nabobil, mener at det er et realistisk tidsperspektiv for en eventuell utleier.

I en tidligere versjon av applikasjonen kunne brukeren velge opp til 15 år frem i tid dersom bilen var registrert i inneværende år. Antall år som er mulig å leie ut ble da begrenset til hvor mange år bilen var, slideren og grafen ble tilpasset dette automatisk. Eksempelvis var det for en bil registrert i 2005 kun mulig å velge mellom en og fem år med utleie dersom oppslaget ble gjort i 2020.

Etter en intern vurdering sammen med Nabobil ble det avgjort at det maksimalt var mulig å velge mellom en og fem år i webapplikasjonen. Dette løser flere problemer. For det første er var grafen uleselig på mobilskjerm med 15 søyler. For det andre er det urealistisk å tenke 15 år frem i tid når det gjelder bilhold. Derfor ønsker vi å gi brukeren er mer realistisk tidshorisont på maksimalt fem år frem i tid.

Når brukeren benytter slideren for å bestemme hvor mange år bilen kan leies ut vil grafen og verdiene over grafen automatisk endres. I

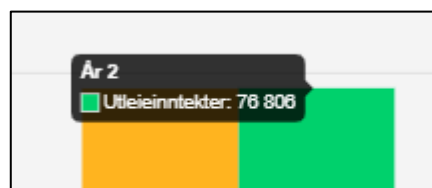


Figur 53 - Informasjon om samlet verdi

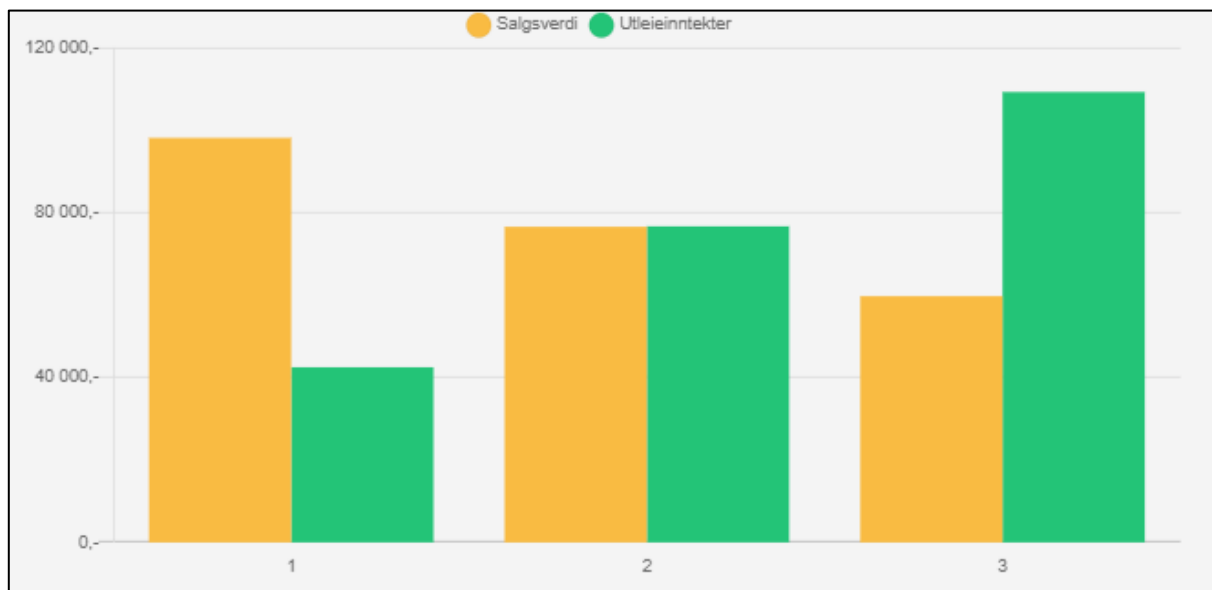
figuren til høyre er informasjon om bilens verdi og fortjeneste ved utleie basert på antall år brukeren har valgt. I dette eksemplet er tre år valgt, da oppgis salgsverdien til bilen om tre år i tillegg til samlet utleieverdi bilen kan opparbeide seg i dette tidsrommet.

Nest nederst på siden er en grafisk fremstilling av verdiene som er oppgitt tidligere i webapplikasjonen, dette kan ses i Figur 55 - graf av utleie og salgsverdi. Grafen fremstiller bilens salgsverdi i den oransje kolonnen og bilens samlede utleieinntekt i den grønne. Grafen er dynamisk og tilpasses automatisk når brukeren endrer antall år med utleie og antall dager i uken med utleie. På X-aksen vises antall år med utleie, mens y-aksen viser verdien i kroner. Salgsprisen i år en vil alltid tilsvare estimert salgsverdi i inneværende år for så å oppgi verditapet på bilen i årene fremover.

Ved å holde datamusen over kolonnene vil den eksakte verdien til kolonnen vises til brukeren. Dette gir brukeren mulighet til å undersøke de eksakte verdiene.



Figur 54 - Hoverfunksjon på kolonne



Figur 55 - graf av utleie og salgsverdi

Nederst på siden er det flere valg for hvordan brukeren kan gå videre. For det første kan de dele en lenke til sin spesifikke verdivurdering, videre er det en direktelenke til nabobil.no. Her kan brukeren lese mer om hvordan det er å være utleier og registrere seg heldigitalt med BankID.



Figur 56 - Brukerens valg

For det tredje kan brukeren dele verdivurderingen sin på Facebook.

### 7.3 Mobilversjon

Mobilversjonen er i stor grad lik versjonen for stor skjerm, men designet er tilpasset den begrensede plassen på skjermen. I praksis er det kun enkelte bakgrunnsfigurer og farger som varierer mellom de to versjonene.

Skjermdumper fra mobilversjonen kan ses i figurene på de to neste sidene.

 Norsk

## Selge eller leie ut bilen?

Økonomi og miljø kan gå hånd i hånd. Ved å leie ut bilen reduserer du kostnadene på bilholdet, samtidig som den som leier kan leie akkurat den bilen de ønsker til en brøkdel av det å eie en bil og garasje.

På denne siden kan du få et prisestimat på hvor mye du kan tjene på å leie ut din bil - sammenlignet med hva du kan tjene på å selge den.

Registreringsnummer  
E K 3 3 1 8 1

Kilometerstand  
20000

Postnummer  
0484

REGN UT

Figur 57 - Webapplikasjonen på mobil 1



## Peugeot Ion 2017

Karosseri:	Flerbruksbil
Antall seter:	4
Farge:	Hvit
Drivstoff:	Elektrisitet

Estimert salgsverdi:

# 100 800 kr

for år 2020

## Din bils utleieverdi

Basert på 4-seters flerbruksbil i Oslo

Lei ut bilen din på Nabobil.no, tjen penger på bilen din og bidra til miljøet.

Inntekten per dag er basert på utleie

Figur 58 - Webapplikasjonen på mobil 2

gjennomført på tilsvarende biler (alder og biltype) i området hvor bilen din står.

Faktisk pris og utleievarighet vil variere basert på etterspørsel.

Estimert utleieverdi:

# 3 546 kr

per måned ved utleie 3 dager i uken

---

Her kan du se hvor mye du kan tjene på å leie ut bilen din!

Du velger antall dager i uken andre kan bruke bilen din.

Så velger du hvor mange år til du tenker å selge bilen, husk at bilen maks kan være 15 år gammel for å kunne leise ut på Nabobil.no.

I grafen ser du verditapet på bilen og kumulativ inntekt på utleie.

3

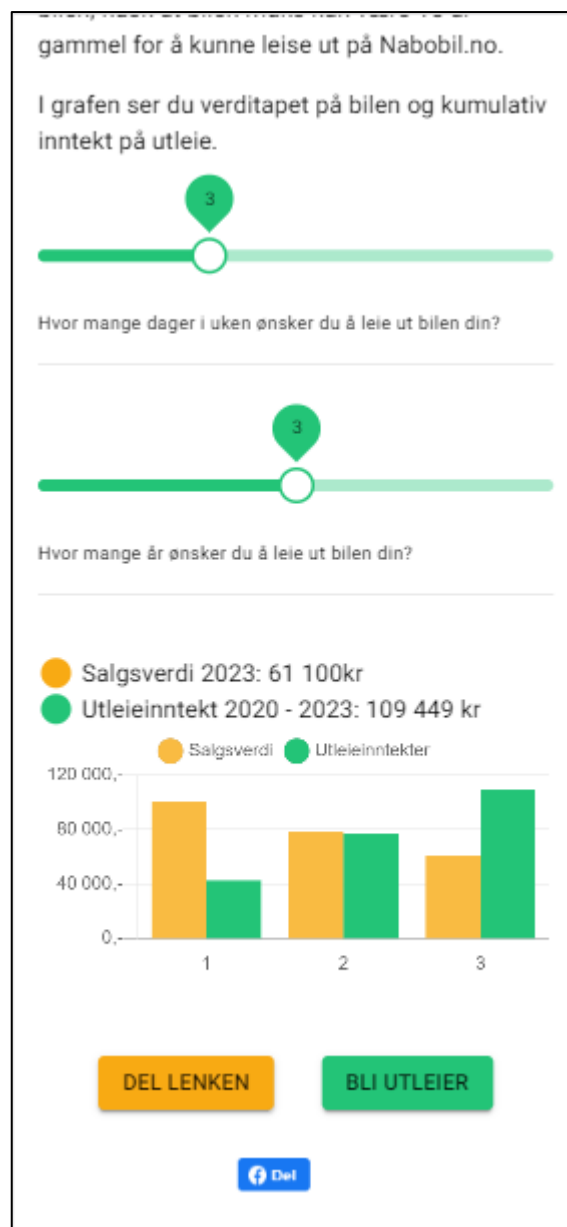
Hvor mange dager i uken ønsker du å leie ut bilen din?

---

3

Hvor mange år ønsker du å leie ut bilen din?

Figur 59 - Webapplikasjonen på mobil 3



Figur 60 - Webapplikasjonen på mobil 4

## 7.4 Visualisering

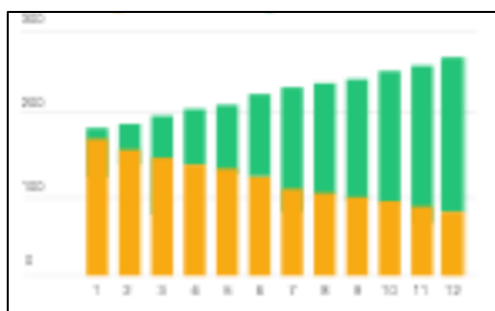
Formålet med denne delen av oppgaven er å se på visualiseringene i webapplikasjonen.

### 7.4.1 Visualisering

Visualisering er et virkemiddel som gjør det mulig for brukeren å enklere forstå data.

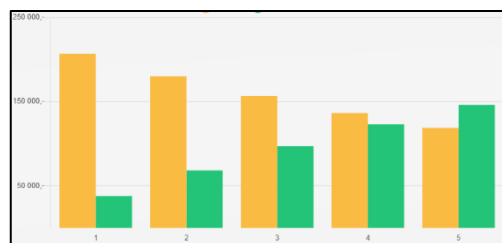
Brukeren gis et alternativt bilde av dataen, slik at de slipper å sette seg inn i en komplisert tabell med data. Dataen brukeren får oppgitt er salgspris for bilen og forventet utleieinntekt for hvert år framover. Disse verdiene er oppgitt i norske kroner og er numerisk kontinuerlig data, som blir representert på Y-aksen. X-aksen representerer antall år frem i tid, alt fra ett til fem år avhengig av hvor mange år brukeren velger. To datasett som skal sammenlignes gir bivariate data.

Først ble det forsøkt med en stacked bar chart vist i Figur 62 (Kirk, 2012, s. 161). En svakhet ved et slik diagram er at det kan være vanskelig å lese lengden på søylene og sammenligne dem med hverandre, fordi de ikke har en felles grunnlinje. Dermed blir det vanskelig for bruker å se utleieinntekter sammenlignet med bilens salgsverdi. Derfor ble det heller benyttet en gruppert bar chart sett i Figur 61.



Figur 62 - Stacked bar chart

Bar chart tillater for nøyaktig sammenligning mellom kategorier og datasett. Y-aksen starter på 0. Dermed viser visualiseringen dataen i full relasjon (Kirk, 2012, s. 254). Med en slik visualisering er det enklere for brukeren å sammenligne salgsverdier og utleieinntekter for hvert år. En svakhet er at det kan være vanskelig å se summen av salgsverdi og utleieinntekt for hvert år. Videre kan det være vanskelig for bruker å se om totalsummen etter ett år er større enn totalsummen om fem år. En løsning



Figur 61 - Grouped bar chart.

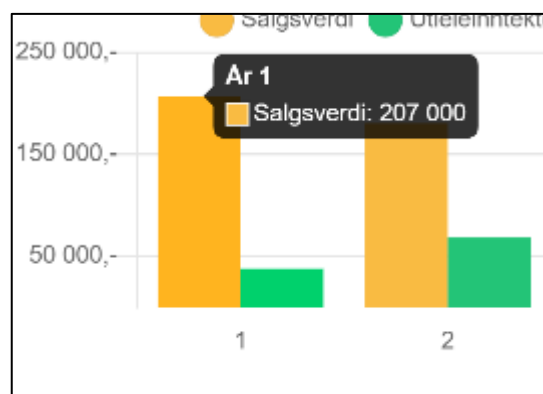
som ble vurdert gikk på å skrive ut summen for det siste året valgt, og sammenligne denne med salgsverdi til bilen slik vist i Figur 63.

Denne løsningen ble ikke så stilren, og sammenligner kun siste år valgt med salgsverdien. Det ble derfor besluttet å ha visualiseringen uten denne ekstra informasjonen, slik at det blir mer fokus på søylene og mindre støy for bruker. I en eventuell neste iterasjon kunne man sett på en mer elegant løsning på dette problemet.



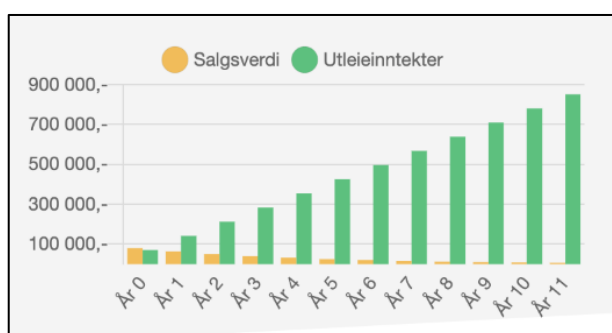
Figur 63 - Siste år sum sammenlignet med salgsverdi nå

Visualiseringen har interaksjonsmuligheter, bruker kan holde musepekeren over de forskjellige søylene på pc-versjonen, eller trykke på dem i mobilversjonen, og slik få opp informasjonen om søylene som vist i Figur 64. På mobilversjon ble plassen et problem dersom brukeren valgte mange år frem i tid.

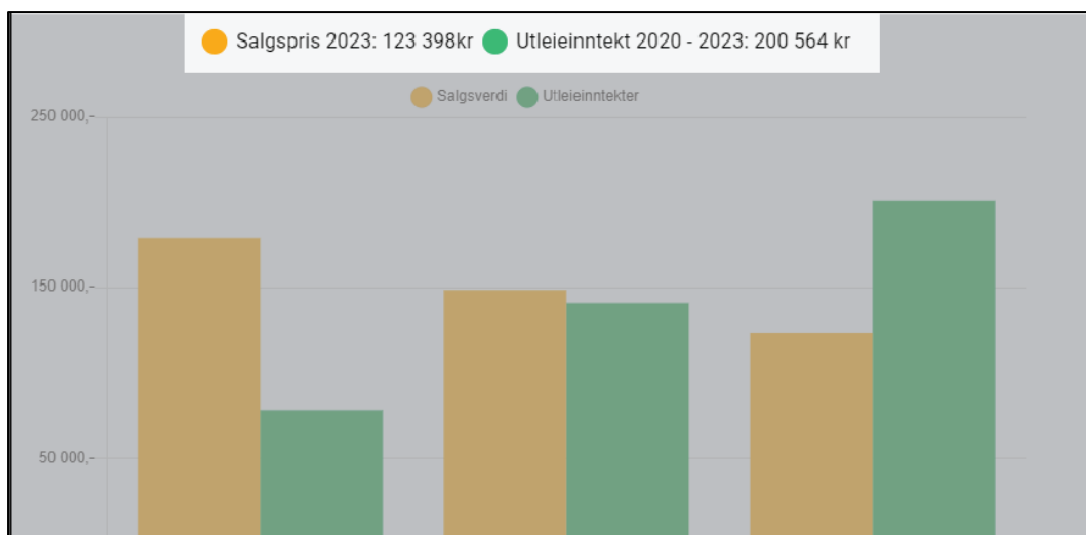


Figur 64 - Hover interaksjonsmulighet

Originalt gikk det å an velge opptil 15 år frem i tid, slik Figur 65 viser. Utleieinntektene ble da så høye at det var vanskelig å lese bilens salgsverdi, og utfordrende å trykke på søylene for å se den nøyaktige verdien. Etter et møte med Nabobil ble det bestemt å vise maksimalt 5 år fremover, blant annet fordi utregningen kan bli unøyaktig så mange år frem i tid. Dette løste plassproblemene. I tillegg ble det lagt til en indikator øverst som sier hvor mye søylene for det siste årets samlede verdi vist i Figur 66. Dette løser eventuelle vanskeligheter med å trykke på søylene på mindre skjermer.



Figur 65 - Visualisering med 11 år fram i tid.



Figur 66 - Indikatorer i toppen ble lagt til for å vise verdiene av barene.








## 7.4.2 Farger

Figur 67 viser fargepaletten brukt i webapplikasjonen.

Hovedfargene er grønn og oransje. Svart brukes for tekst, og hvit og lysgrå som bakgrunn slik at fargene skiller seg ut fra resten av webapplikasjonen. Valget av grønnfargen er inspirert av nabobil.no (Nabobil.no, 2020) hvor grønt er mye brukt i forskjellige varianter. Dette var et naturlig valg siden webapplikasjonen er utviklet for Nabobil, og designet er gjennomgående. Oransje ble valgt ved hjelp av et fargehjul-verktøy (Adobe, 2020) hvor oransje er triadfargen til grønn, og passet godt i designet. Siden to datasett blir visualisert for bruker, henholdsvis bilens salgsverdi og utleieinntekter, passet det godt med to farger for hele designet.

Fargene blir brukt for å fange brukerens oppmerksomhet.

Alle elementer bruker kan interagere med, eksempelvis knapper, skjema, slidere og visualisering, bruker enten fargen grønn eller oransje. Fargeverdien til både grønn og oransje er lys nok til at svart tekst på grønn gir et kontrastforhold på 9.23/1 og svart på oransje et kontrastforhold på 10.75/1, slik at fargene egner seg godt som knapper, hvor kontrasten er høyere enn høyeste nivå (AAA-nivå) på 7.0/1 (Digitaliseringsdirektoratet, 2020). Bruken av gråtoner, svart og hvitt er enten veldig lys eller veldig mørk, hvor mellomtoner unngås, da disse kan gi dårlig kontrast.

	Grønn #23C477		Lys grønn #A5E2C5
	Oransje #F8AA13		Rød oransje #F77214
	Hvit #FFFFFF		
	Lys grå #F4F4F4		
	Svart #000000		

Figur 67 - Fargepalett for webapplikasjonen.



## 7.5 Forbedringspotensial

Basert på tilbakemeldinger fra Nabobil og gruppens egen vurdering ser vi at det er forbedringspotensial i enkelte aspekter ved applikasjonen. Nabobil har uttalt at de er veldig fornøyde med produktet og at det gruppen har utviklet svarer til deres forventninger.

Et produkt kan aldri bli perfekt og man må alltid søke etter det optimale designet basert på de forutsetninger og begrensninger man har.

Even if you managed to find the mythical “optimal” solution within the cross domain constraints of a design problem, there is another reason that optimal design is impossible: things change. On the day your code is finished, your design stops changing, but the world keeps moving. (Berkun, 2020)

At gruppen kan overlevere et produkt som er perfekt er en naiv tankegang. Som Berkun antyder vil alltid forutsetningene og verden endre seg. Derfor er det viktig at gruppen gjennomfører en vurdering av webapplikasjonen og dens mangler, før overlevering av ferdig produkt. På den måten kan gruppen gi oppdragsgiver verdifull innsikt i utviklernes tankegang.

- Siden ser fortsatt noe tom og mangelfull ut før brukeren gjennomfører et oppslag på bil. Dette ble kommentert i del 7.2. Det ble gjennomført tiltak for å forbedre dette problemet, men det er til en viss grad fortsatt til stede. Gruppen anbefaler at det blant annet opprettes en footer. Eventuelt kan det legges inn brukerhistorier fra de som har registrert seg på nabobil.no via salgellerutleie.no.
- Grafen som fremstiller salgs- og utleieverdi over flere år, kan fremstå som utydelig for noen. Forbedringspotensialet for grafen diskuteres nærmere i del 7.4.1.
- Dersom gruppen hadde mer tid til disposisjon kunne det vært ønskelig å støtte deling av verdivurderingen på flere sosiale medier, som for eksempel Twitter.
- Dersom API'et tillater det i fremtiden er det ønskelig at nøkkelfri teknologi blir lagt til i tjenesten som en del av verdivurderingen for utleie. Dette vil markedsføre teknologien og kan få flere brukere til å migrere over på nøkkelfri teknologi.
- Estimert salgspris kan innsnevres ved å legge til enkelte variabler som brukeren kan krysse av ja eller nei for. Dette vil gi en mer nøyaktig verdivurdering og skape tillit til tjenesten. Flere eksempler som kan dekkes er:
  - Dyrehold i bilen, dette senker bilens verdi.
  - Røyking i bilen, kan senke bilens verdi drastisk.
  - Riper/bulker på bilen, det kan skilles på stor eller liten skade.
  - Har bilen Dab-radio.
  - Har bilen ekstra utstyrspakker.

- Om en som benytter tjenesten faktisk ønsker å selge bilen sin heller enn å leie den ut kan tjenesten henvise brukeren å heller registrere seg som leietaker på nabobil.no. I dag fungerer tjenesten mest som en markedsføringsplattform for de som ønsker å registrere bilen sin på nabobil.no, ikke de som ønsker å selge den og klare seg uten bil i hverdagen.
- På mobilvisningen er det litt lite «luft» mellom enkelte komponenter. Dette gjør at teksten kan oppleves som presset inn på siden.

Dette er noen problemstillinger som hadde vært interessante å se på i en eventuell ny iterasjon av webapplikasjonen.

## 8 Konklusjon

Nabobil.no ønsket at gruppen skulle gjennomføre et prosjekt som resulterte i et produkt for allmenheten som sammenlignet bilers salgs- og utleieverdi. Det endelige produktet som er presentert gjør nettopp det. Webapplikasjonen er slik det kommer frem i rapporten klar for lansering, noe det jobbes med fortløpende.

Selv om webapplikasjonen oppfyller kravspesifikasjonen som ble fastsatt tidlig i prosjektet er det som nevnt i del 7.5 fortsatt rom for forbedring. Gruppen har likevel brukt den kunnskapen de har tilegnet seg under utdanningen for å gjennomføre prosjektet på best mulig måte.

Selv om produktet ikke er ideelt, er det riktig å lansere det. På den måten kan gruppen og Nabobil måle effekten av at tjenesten lanseres. Siden dette ikke gjøres før innleveringsfristen av prosjektet er det vanskelig å si noe om hvordan tjenesten blir mottatt av markedet.

Til slutt håper gruppen at Nabobil ønsker å videreutvikle og markedsføre produktet på en måte som gjør at flere velger å benytte seg av tjenesten og at det fører til økt utleievirksomhet på plattformen deres.

Prosjektet har vært svært lærerikt for medlemmene i gruppen. Gruppen har i større grad enn ved tidligere prosjekter hatt forskjellige roller. Dette har ført til at ikke alle har oversikt over hele systemets oppbygging, men kun sine respektive deler. Denne måten å jobbe på er annerledes i forhold til tidligere gruppeprosjekter i andre fag, men reflekterer arbeidslivet i større grad og har gitt gruppen mer innsikt i hvordan arbeidshverdagen kan fungere.

Gruppen har lært å jobbe sammen på store prosjekter med muligheten til å møtes for diskusjon angående fremgang og problemstillinger, men også opplevd hvordan det er å jobbe desentralisert uten muligheten til å jobbe sammen. Å jobbe sammen i person og gjennom digitale løsninger er meget forskjellig fra hverandre og var helt klart en utfordring i prosjektet.

Gruppen har satt seg inn i nye teknologier og lært mye om samspillet mellom dem. Læringskurven var bratt underveis i prosjektet og har gitt gruppen mye kunnskap som kan tas med videre.

Til slutt ønsker gruppen å rette en stor takk til Nabobil for et veldig spennende og lærerikt prosjekt.

## 9 Vedlegg

### 9.1 Prosjektlogg

### 9.2 Systemtest

#### 9.2.1 Systemtesting

I en systemtest blir noen eller alle komponentene i systemet testet som en helhet (Sommerville, 2011, s. 232). I dette prosjektet er det valgt å bruke verktøyet Cypress, et MIT-lisensiert JavaScript-basert bibliotek for ende-til-ende testing.

#### 9.2.2 Testplan

Det ble utformet en testplan for å planlegge hvilke områder av løsningen som skal testes.

Tabell 5 - Testplan

	Formål med testen	Hvilket område av programmet gjelder testen?	Hvilken metode skal brukes?
1. Form test	Kontrollere at applikasjonens inputs godtar gyldige data, ikke godtar og viser feilmelding ved ugyldige data	Front-end, inputfelt	Automatisk testing med Cypress
2. Visualisering/slider test	Kontrollere at visualisering er synlig, og at endring på tilhørende «slider-inputs» påvirker verdi-estimerer	Front-end, input	Automatisk testing med Cypress
3. Render test	Kontrollere at datahentning fungerer og resulterer i visning av bilinformasjon	Front-end, back-end	Automatisk testing med Cypress
4. Translate test	Kontrollere at det er mulig å endre språk	Front-end	Automatisk testing med Cypress

Basert på testplanen over ble det utarbeidet en rekke tester for å dekke områdene:

▼ Form test

☐ is not possible to submit without filling the form
 ☐ shows an error if registration is too short
 ☐ shows an error if registration is too long
 ☐ does not show error if registration is valid
 ☐ shows an error if mileage is negative
 ☐ shows an alert if mileage is 250 000 or over
 ☐ shows an error if mileage is over 420 000
 ☐ does not show an error if mileage is between 0 and 420000
 ☐ shows an error if postcode is less than 4 numbers
 ☐ shows an error if postcode is not a number
 ☐ shows an error if postcode is more than 4 numbers
 ☐ does not show an error if postcode is 4 numbers
 ☐ Can fill the form and submit

Figur 68 – Form test

▼ Sliders and BarChart test

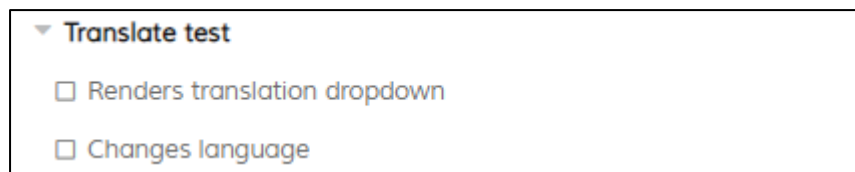
☐ Barchart is visible after fetching data
 ☐ Increasing day slider increases estimated rent income
 ☐ Increasing year slider decreases sales value

Figur 69 – Visualiserings/Slider test

▼ Render test

☐ Renders all sections after data fetching
 ☐ Renders an estimated sales value after data fetching
 ☐ Renders an estimated rent value after data fetching
 ☐ Renders share buttons after data fetching
 ☐ Renders an alert if fetched car is 15 years or older

Figur 70 – Render test



Figur 71 – Translate test

### 9.2.3 Testprosedyre

I Cypress blir testene beskrevet med JavaScript-kode, dette muliggjør automatisk gjennomføring av testene. Ved kjøring av tester med Cypress startes en instans av en nettleser, og de beskrevne handlingene blir utført direkte på grensesnittet. Testene kan kjøres lokalt på utviklingsserver eller mot applikasjon som er hostet i skyløsning. Dette styres fra konfigurasjonsfil for Cypress «cypress.json» i prosjektets grunnmappe.

Testene kjøres ved kommandoen «yarn test»

```
it('does not show an error if mileage is between 0 and 420000', () => {
  const randomNumberBetween0And420000 = Math.floor(Math.random() * 420001).toString();

  cy.visit('/');
  cy.get('form');

  cy.get('input[id="Kilometerstand"]').type(randomNumberBetween0And420000).should('have.value', randomNumberBetween0And420000).blur();
  cy.get('p[id="Kilometerstand-helper-text"]').should('not.exist');
});

it('shows an error if postalcode is less than 4 numbers', () => {
  cy.visit('/');
  cy.get('form');

  cy.get('input[id="Postnummer"]').type('1').should('have.value', '1').blur();
  cy.get('p[id="Postnummer-helper-text"]').contains('Postnummer må være fire tall');

  cy.get('input[id="Postnummer"]').type('11').should('have.value', '11').blur();
  cy.get('p[id="Postnummer-helper-text"]').contains('Postnummer må være fire tall');

  cy.get('input[id="Postnummer"]').type('111').should('have.value', '111').blur();
  cy.get('p[id="Postnummer-helper-text"]').contains('Postnummer må være fire tall');
});
```

Figur 72 – Utdrag fra form-test

### 9.2.4 Godkjenningskriterier

Systemtestene anses som godkjent når ingen av testene feiler.

### 9.2.5 Testrapport systemtest

Testene har blitt utført som beskrevet i testplanen og testprosedyren over. Den automatiske testingen bør gjennomføres hver gang systemet bygges.

Under følger en rapport fra testing 12. Mai 2020.

Running: barchart.spec.js

(1 of 4)

#### Sliders and BarChart test

- ✓ Barchart is visible after fetching data (7855ms)
- ✓ Increasing day slider increases estimated rent income (8860ms)
- ✓ Increasing year slider decreases sales value (5849ms)

3 passing (22s)

#### (Results)

```
Tests:      3
Passing:    3
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   21 seconds
Spec Ran:   barchart.spec.js
```

Figur 73 – Resultat visualisering/slider test

Running: form.spec.js

(2 of 4)

#### Form test

- ✓ is not possible to submit without filling the form (2011ms)
- ✓ shows an error if registration is too short (753ms)
- ✓ shows an error if registration is too long (1585ms)
- ✓ does not show error if registration is valid (1235ms)
- ✓ shows an error if mileage is negative (902ms)
- ✓ shows an alert if mileage is 250 000 or over (2013ms)
- ✓ shows an error if mileage is over 420 000 (1838ms)
- ✓ does not show an error if mileage is between 0 and 420000 (931ms)
- ✓ shows an error if postcode is less than 4 numbers (1733ms)
- ✓ shows an error if postcode is not a number (1160ms)
- ✓ shows an error if postcode is more than 4 numbers (1235ms)
- ✓ does not show an error if postcode is 4 numbers (1225ms)
- ✓ Can fill the form and submit (1846ms)

13 passing (19s)

#### (Results)

```
Tests:      13
Passing:    13
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   18 seconds
Spec Ran:   form.spec.js
```

Figur 74 – Resultat form test

```

Running: render.spec.js (3 of 4)

Render test
  ✓ Renders all sections after data fetching (7912ms)
  ✓ Renders an estimated sales value after data fetching (6046ms)
  ✓ Renders an estimated rent value after data fetching (5571ms)
  ✓ Renders share buttons after data fetching (5610ms)
  ✓ Renders an alert if fetched car is 15 years or older (5597ms)

5 passing (31s)

(Results)

Tests:      5
Passing:    5
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   30 seconds
Spec Ran:   render.spec.js

```

Figur 75 – Resultat render test

```

Running: translate.spec.js (4 of 4)

Translate test
  ✓ Renders translation dropdown (1791ms)
  ✓ Changes language (1502ms)

2 passing (4s)

(Results)

Tests:      2
Passing:    2
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   3 seconds
Spec Ran:   translate.spec.js

```

Figur 76 – Resultat translate test

(Run Finished)

Spec		Tests	Passing	Failing	Pending	Skipped
✓ barchart.spec.js	00:22	3	3	-	-	-
✓ form.spec.js	00:24	14	14	-	-	-
✓ render.spec.js	00:30	5	5	-	-	-
✓ translate.spec.js	00:03	2	2	-	-	-
✓ All specs passed!	01:20	24	24	-	-	-

Done in 111.92s.

Figur 77 – Testrapport fra Cypress



### 9.2.6 Konklusjon

«Testing can only show the presence of errors, not their absence!» - Dijkstra (Sommerville, 2011)

Testene som er utført her kan ikke bevise at applikasjonen er feilfri. Allikevel testes de viktigste delene og funksjonene i applikasjonen, slik at sannsynligheten for at en endring som skaper feil blir ført til produksjon minsker.

### 9.3 Brukertest

Da tjenesten var ferdig utviklet ble det sendt ut en kvantitativ brukertest/vurdering av tjenesten. Ideelt sett hadde gruppen gjennomført kvalitative brukertester i person, men dette er dessverre vanskelig grunnet av Covid-19.

Resultatene fra brukerundersøkelsen er vedlagt oppgaveteksten. Her kan alle spørsmål og svar ses i tillegg til beregninger gjort av gruppen i ettertid av brukertesten. Testen hadde 17 respondenter. Et av kravene for å ta testen var at testsubjektet må eie eller benytte en bil de kjenner godt.

Gruppen velger å trekke frem enkelte aspekter vi mener er interessante ved testen.

Av 17 respondenter hadde seks stykker biler som hadde kjørt lengre enn 300 000 km eller var mer enn 15 år gamle, disse får ikke spørsmål om bilens utleieverdi.

#### 9.3.1 Forventninger til bilens salgs- og utleieverdi

Respondentene ble stilt følgende spørsmål:

- Hvor mye tror du din bil er verdt på bruktbilmarkedet i dag? Oppgi svar i NOK
- Hvor mye tror du, du kan tjene per måned ved å leie ut din bil 3 dager i uken? Oppgi svar i NOK

Respondentene blir så bedt om å benytte tjenesten, med følgende spørsmål oppfølgingsspørsmål:

- Hvor mye sier nettsiden salgsverdien på din bil er i dag?
- Hvor mye sier nettsiden utleieverdien på din bil er per måned om du leier ut 3 dager i uken?

Resultatet viser, basert på vår webapplikasjons verdivurdering at: (i) 8 stk. tror bilen er mer verdt enn de blir vurdert til. (ii) 5 stk. tror bilen er mindre verdt enn den blir vurdert til og (iii) 3 stk. tror bilen er verdt ca. det samme som den blir vurdert til. Dette bekrefter til en viss grad Nabobils utsagn om at folk flest nødvendigvis ikke har et forhold til sin bils verdi.

Av de 11 respondentene som fikk spørsmål om utleieverdi, trodde 10 av 11 at de kunne tjene mer ved å leie ut bilens sin enn hva vår applikasjon tilsa. Dette kan være en av grunnene til at folk flest ikke vil registrere bilen sin på en delingsplattform, fordi de har for høye forventninger til bilens utleieverdi.

### 9.3.2 Selve nettsiden

Respondentene fikk følgende spørsmål:

- På en skala fra 1-5, hvor enkel opplever du at nettsiden er å bruke? Hvor 1 er «ikke enkel» og 5 er «meget enkel».

15 av 17 respondenter svarte 5 på dette spørsmålet og to svarte 4. Dette gir oss et snitt på 4,882. Dette viser at gruppen har oppnådd målet om å utvikle en nettside som er enkel å bruke.

Respondentene fikk følgende spørsmål:

- Ville du anbefalt denne nettsiden til noen du kjenner? (Ja/Nei)

15 av 17 respondenter svarte ja på spørsmålet, de resterende to svarte kanskje/vet ikke. Dette lover godt for sidens popularitet og er et tegn på at brukerne var fornøyde med tjenesten.

### 9.3.3 Informasjon på nettsiden

Respondentene fikk følgende spørsmål:

- Utforsk resultatene på søket ditt, føler du at du finner all informasjon du vil ha? (Ja/Nei)
  - Hvis Nei: Hvilken informasjon føler du mangler?
- Kom det klart frem hvor mye du kan tjene ved å leie ut bilen din. (Ja/Nei)
  - Hvis Nei: Hvorfor kom ikke dette klart fram? Hva var uforståelig?

På begge spørsmålene svarte 15 av 17 respondenter ja, dette er en god indikator på at tjenesten for det meste oppfyller det informasjonsbehovet den er ment til å dekke. På oppfølgingsspørsmålet om fortjeneste ved utleie var svarene: «Bilen var for gammel, og hadde gått for langt» og «Bilen var 15 år, og hadde gått over 300000». Dette vil si at disse også kunne ha svart ja på dette spørsmålet, siden de ikke kan tjene noe på bilen.

Det var usikkerhet rundt hvorvidt spørsmålet om fortjeneste ved utleie skulle stilles til de som oppga at bilen var for gammel eller hadde kjørt for langt. Beslutningen falt på å stille dette spørsmålet til alle da de bilene som ikke kan leies ut i teorien har null i utleieverdi. Dette virker det som om alle respondentene har forstått.

### 9.3.4 Utleie på nabobil.no

Respondentene fikk til slutt følgende spørsmål:

- Etter å ha benyttet tjeneste, vurderer du å leie ut bilen din på Nabobil.no? (Ja, Nei, Kanskje/Vet ikke)
  - Hvis Nei eller kanskje/vet ikke: Hvis du ikke vil eller er usikker, hvorfor vil du ikke leie ut bilen din på nabobil.no?

Dette spørsmålet gikk kun til de respondentene med biler som kvalifiserer til utleie på nabobil.no.

Av 17 respondenter svarte 7 «Kanskje/vet ikke», 6 «nei» og 4 «ja». Dette betyr at det er mye usikkerhet rundt det å registrere bilen sin for utleie, men at man med tilpasset markedsføring kan få flere til å vippe mot «ja». Dette kan gjøres ved å vise til fordelene ved utleie, hvordan prosessen fungerer og at forsikring er inkludert. Man kan også vise til andre som faktisk har erfaring med utleie av bil på Nabobils plattform.

På oppfølgingsspørsmålet fikk vi følgende tilbakemeldinger:

- Redd for skade
- Frihet...
- Hvis man ser bort i fra korona, så er det nok mer med at jeg bruker bilen hver dag, hadde jeg bodd nærmere jobben så kunne jeg ha syklet, for å så leie ut bilen, å kun bruke den selv når jeg drar på hytta for eksempel👍
- Jeg er for redd for bilen min til å låne den ut til fremmede
- Trenger den daglig selv
- Ukjent med hvordan tjenesten er i praksis
- Bruker den mye selv
- Er for avhengig av den selv i dette øyeblikk.

En gjennomgående trend er at folk benytter bilen sin daglig og derfor ikke kan leie den ut. Informasjon om utleieprosessen kan også få flere av disse til å velge å leie ut bilens sin når de har mulighet til det. Ved å registrere seg på nabobil.no kan man selv bestemme når man leier ut bilen. En mulighet er for eksempel å kun leie ut bilen sin når man selv er på ferie andre steder i landet, eller i helgene når man selv ikke trenger bilen.

Mer informasjon om selve utleieprosessen og de mulighetene som finnes kunne også fått disse personene til å leie ut bilen sin på nabobil.no.

## 10 Prosjektdagbok

14. Januar

Første møte med Nabobil. Diskuterte forskjellige forslag til prosjekt og landet på ett prosjektforslag (bilpriskalkulator).

20. Januar

Møte - utforming av skisser / low-fi prototyper. Diskutering/planlegging av videre fremdrift.

21. Januar

Utforming av hi-fi prototype i React

29. Januar

Andre møte med Nabobil. Fått klarhet i APIer, oversendt kontrakt, presentert prototype. Senere møte med veileder hvor vi presenterte prosjektet vi skal gjøre, presenterte prototype.

3. Februar

Møte - Utformet forprosjektrapport, fikset og publisert nettside. Utformet kravspesifikasjoner, risikomatrise etc.

4. februar

Møte - Ferdigstilt forprosjektrapport

13. Februar

Fortsatt ikke fått APIer fra Nabobil. Utformet sekvensdiagrammer etc. Og backend-arkitektur, API med mock-data. Utformet flere frontend-sketcher.

14. Februar

Begynt å kode front-end, jobbed med front-end sketche

2. Mars

Utformingen av sluttrapporten settes i gang, vi venter fortsatt på tilgang på Nabobils API for å igangsette backend. Frontend er påbegynt og utvikles foreløpig med mock-data.

Resten av mars

Resten av mars er det lite fremgang på prosjektet da andre ting prioriteres.

7. April

Vi har møte med Nabobil og får tilgang til API'er og kan påbegynne backend. Oppgaveskrivingen er påbegynt i de delene som er mulig å skrive før ferdigstilling av produktet.

29. april

Den tekniske delen av prosjektet er så å si ferdigstilt og vi har et møte med nabobil hvor de gir oss tilbakemeldinger på tjenesten. Vi får et par endringsforslag som vi tar med oss videre

11-25. juni

Prosjektet blir ferdigstilt og blir tatt godt imot hos nabobil. Vi jobber nå med å få siden opp på domenet.

Resten av tiden før innlevering går med på rapportskriving, og alle er med på denne prosessen. Vi har også hyppige møter med veileder i denne tidsperioden.

## 11 Referanser

- Adobe. (2020). *Color wheel, a color palette generator*. Hentet fra Adobe Color: <https://color.adobe.com/create/color-wheel>
- Barland, M. (2015, Desember 5). *The sharing economy*. Hentet fra Teknologirådet: The sharing economy
- Berkun, S. (2020, Mai 20). *The Myth of a Perfect Design*. Hentet fra Scott Berkun: <https://scottberkun.com/essays/myth-of-perfect-design/>
- Børnick-Sørhaug, L. (2019, Juni 25). *Verdens største bildelingsaktør Getaround kjøper Nabobil.no for over 100 millioner NOK*. Hentet fra Nabobil blogg: <https://blogg.nabobil.no/blog/getaround>
- Digitaliseringsdirektoratet. (2020, Mai 02). *WCAG 2.0-standard*. Hentet fra Digitaliseringsdirektoratet: <https://uu.difi.no/krav-og-regelverk/wcag-20-standard>
- Facebook Inc. (u.d.). *A Guide to Sharing for Webmasters*. Hentet fra facebook for developers: <https://developers.facebook.com/docs/sharing/webmasters/>
- Facebook Inc. (u.d.). *Context*. Hentet fra Context - React: <https://reactjs.org/docs/context.html>
- Finn.no. (2020, Mai 2). *Pristips for bruktbil*. Hentet fra Finn.no: <https://www.finn.no/pristips/bruktobil>
- Funksjonshemmedes Fellesorganisasjon. (2020). *Bli kjent med FN-konvensjonen om rettighetene til mennesker med nedsatt*. Oslo: Funksjonshemmedes Fellesorganisasjo. Hentet 2020 fra Funksjonshemmedes Fellesorganisasjon.
- Hopland, S. (2019, Juni 25). *Amerikansk bildelingsgigant kjøper Nabobil.no for over 100 millioner*. Hentet fra E24: <https://e24.no/naeringsliv/i/naV4VJ/amerikansk-bidelingsgigant-kjoeper-nabobilno-for-over-100-millioner>
- Interaction Design Foundation. (2020, Mai 9). *User Experience (UX) Design*. Hentet fra Interaction Design Foundation: <https://www.interaction-design.org/literature/topics/ux-design>
- Kirk, A. (2012). *Data Visualization: a successful design process*. Pact Publishing.
- Kvd Norge. (2020, Mai 2). *Kvd Norge*. Hentet fra Kvd Norge: <https://www.kvdnorge.no/bilvardering>
- Limpitsouni, K. (2020, 05 20). *unDraw*. Hentet fra unDraw - Open source illustrations for any idea: <https://undraw.co/>
- Morville, P. (2020, Mai 07). *User Experience Design*. Hentet fra Semantic Studios: [http://semanticstudios.com/user\\_experience\\_design/](http://semanticstudios.com/user_experience_design/)
- Nabobil.no. (2020, Mars 25). *Nabobil*. Hentet fra Nabobil: <https://nabobil.no>

Nabobil.no. (2020, Mai 8). *Tjen penger på bilen din* . Hentet fra For utleiere:  
<https://nabobil.no/for-utleiere>

Skólski, P. (2020, Mai 18). *Single-page application vs. multiple-page application*. Hentet fra Neoteric: [https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/?utm\\_source=medium.com&utm\\_medium=social&utm\\_content=neo&utm\\_campaign=blog](https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/?utm_source=medium.com&utm_medium=social&utm_content=neo&utm_campaign=blog)

Snadnes, F. E. (2011). *Universell utforming av IKT- systemer - brukergrensesnitt for alle*. Oslo: Universitetsforlaget .

Sommerville, I. (2011). *Software Engineering*. London: Pearson Education Limited.

Statens vegvesen . (2020, Februar 6). *Personlig bilskilt*. Hentet fra Statens vegvesen:  
<https://www.vegvesen.no/kjoretoy/Eie+og+vedlikeholde/skilt/personlig-bilskilt>