

# DriveMe

---

SOFTWARE ENGINEERING HIØ 2022

Marcus Galdal Tollefsen, Yunus Øzdemir, Ridwan Abukar, Abdala Ali,  
Gorgos fares Tammo

## Innholdsfortegnelse

<b>GRUPPEMEDLEMMER.....</b>	<b>3</b>
<b>INFO OM APPLIKASJONEN: .....</b>	<b>4</b>
<b>KRAV .....</b>	<b>5</b>
GENERELLE KRAV: .....	5
<i>Funksjonelle krav:.....</i>	<i>5</i>
<i>Ikke funksjonelt krav: .....</i>	<i>5</i>
LOGGE INN SOM BRUKER .....	6
<i>Funksjonelle krav:.....</i>	<i>6</i>
<i>Ikke funksjonelle krav.....</i>	<i>6</i>
REGISTRERE BIL.....	6
<i>Funksjonelle krav:.....</i>	<i>6</i>
<i>Ikke funksjonelle krav.....</i>	<i>6</i>
BETALING .....	7
<i>Funksjonelle krav:.....</i>	<i>7</i>
<i>Ikke funksjonelle krav:.....</i>	<i>7</i>
LEIE BIL .....	7
<i>Funksjonelle krav:.....</i>	<i>7</i>
<i>Ikke funksjonelle krav:.....</i>	<i>7</i>
UTLEIE AV BIL .....	8
FUNKSJONELLE KRAV: .....	8
IKKE FUNKSJONELLE KRAV: .....	8
<b>PERSONAS.....</b>	<b>9</b>
PERSONAS 1: .....	9
PERSONAS 2: .....	10
PERSONAS 3: .....	11
PERSONAS 4: .....	12
PERSONAS 5: .....	13
<b>USER STORIES-1.....</b>	<b>14</b>
<b>USER-STORIES-2 .....</b>	<b>15</b>
<b>USER CASE: .....</b>	<b>15</b>
USER CASE FIGUR 1: .....	16
<b>KODE FOR USER CASE FIGUR 1: .....</b>	<b>17</b>
<b>USE-CASE 2.....</b>	<b>20</b>
USE-CASE: US1 .....	20
<b>USE-CASE 2 FIGUR .....</b>	<b>21</b>
<b>SEKVENNS DIAGRAM .....</b>	<b>22</b>
SEKVENNS DIAGRAM FIGUR 1: .....	23
<i>Kode for figur 1: .....</i>	<i>24</i>
<b>SEKVENNS DIAGRAM FIGUR 2: .....</b>	<b>27</b>
BESTILLING AV EN BIL.....	27
<b>AKTIVITETS DIAGRAM.....</b>	<b>29</b>
LÅNE UT BILEN.....	29

<i>Aktivitets diagram figur låne ut bilen</i> .....	29
<i>Kode for figur låne ut bilen:</i> .....	29
INNLOGGING .....	31
<i>Aktivitets diagram figur innlogging</i> .....	31
<i>Kode for Aktivitets diagram figur innlogging:</i> .....	31
BIL REGISTRERING .....	33
<i>Aktivitetsdiagram figur registrere bil</i> .....	33
<i>Kode for registrering av bil</i> .....	33
REGISTRERING AV BRUKER .....	33
<i>Aktivitetsdiagram figur for registrering av bruker</i> .....	34
<b>DATAFLYT DIAGRAM :</b> .....	<b>34</b>
<b>DATAFLYT DIAGRAM 2:</b> .....	<b>36</b>
<b>PROBLEMSTILLING OG DOMENET:</b> .....	<b>37</b>
<b>LØSNING TIL PROSJEKTGRUPPEN</b> .....	<b>37</b>
<b>PROTOTYPE</b> .....	<b>39</b>
VEILEDNING FOR Å ÅPNE PROSJEKTET (DIREKTE METODE).....	39
VEILEDNING FOR Å SETTE OPP MAVEN (I TILFELLE 7.1 IKKE FUNGERER).....	42
<i>Litt om prototypen</i> .....	46
PROTOTYPENS LAYOUT OG FUNKSJON .....	48
<b>TESTING VED BRUK AV JUNIT</b> .....	<b>55</b>

## Gruppemedlemmer

Yunus Øzdemir

- [yunusao@hiof.no](mailto:yunusao@hiof.no)

Ridwan Abukar

- [ridwanaa@hiof.no](mailto:ridwanaa@hiof.no)

Marcus Galdal Tollefsen

- [marcusgt@hiof.no](mailto:marcusgt@hiof.no)

Abdala Ali

- [abdalama@hiof.no](mailto:abdalama@hiof.no)

Gorgos Fares Tammo

- [gorgosft@hiof.no](mailto:gorgosft@hiof.no)

## Info om Applikasjonen:

DriveMe er en applikasjon som gjør det mulig for deg som ikke har en bil, men som gjerne trenger å få tilgang til en bil i en kort periode. Vi har utviklet et system hvor kunden kan opprette en bruker via bankid. Systemet lar deg velge mellom alle de tilgjengelige bilene som du ønsker og leie. Systemet lar deg som bruker også registrere egne biler og leie de ut etter behov. Før du kan ta i bruk appen så må du registrere at man har et gyldig førerkort og at du er 18år eller eldre. Ved innlogging gjennom Bankid kan systemet verifisere din alder. I systemet kan du velge mellom hva slags type bil som passer til din tur.

Når du har laget deg bruker og har kommet helt fram til betalingen så er det enkelt og utføre prosessen etter det. Etter godkjent betaling vil kunden få tilsendt kvittering med adresse til der bilen er plassert. Bileier og kunde kommer frem til hvordan de skal overlevere nøkler. Prisen regnes ut automatisk når du avslutter reisen i appen. For brukere som ønsker å registrere seg så finner man mye informasjon i appen før man tar i bruk bilene og registreringen. Betalinger skjer etter turene er ferdig der kostnader som bom, parkering og drivstoff er med i regningen på slutten. Målgruppen blir jo da alle som er over 18, men er mest egnet for folk som trenger å låne en bil i mellomtida, så her er det stor målgruppe.

## Krav

### Generelle krav:

### Funksjonelle krav:

1. Kunde skal kunne logge inn i systemet ved hjelp av en knapp.
2. Kunde skal kunne legge til en eller flere biler for utleie gjennom å registrere de i systemet.
3. Kunde skal få bekreftelse på e-post.
4. Kunde skal kunne logge ut ved å trykke på «logg ut» knappen.
5. Kunde skal kunne velge mellom betalings metoder.
6. Kunden skal kunne betale med vipps, ved å trykke på «Betale med vipps» knappen.
7. Kunden skal kunne betale med bankkort, ved å trykke på «Betale med bank kort» valgalternativet
8. Kunde skal kunne velge om h\*n vil låne bil ved å trykke på «lån bil» knappen
9. Kunde skal kunne velge om h\*n vil låne ut bil ved å trykke på «registrer bil» knappen
10. Kunde skal kunne gå tilbake til forside ved å trykke på «meny» knappen
11. Kunde skal kunne velge om h\*n vil låne ut bil ved å trykke på «registrer bil» knappen

### Ikke funksjonelt krav:

1. Systemet må være raskt, systemet skal ikke bruke mere enn 5 sekunder på å laste inn den gjeldene siden
2. Systemet må være tilrettelagt for enhver, det skal ikke ta en ny kunde mere enn et minutt til å låne en bil.
3. Systemet må fungere på alle plattformer.
4. systemet må handtere 60 000 samtidige brukere per 24 timer.
5. Sjekken om betalingen er fullført skal ikke ta mere enn 3-5 sekunder.

## Logge inn som bruker

### Funksjonelle krav:

1. Skal kunne logge inn via Bankid
2. Konto må valideres gjennom Epost

### Ikke funksjonelle krav

1. Retur til hovedside skal ta mindre enn 5 sekunder
2. bruker logges automatisk ut etter 30min med inaktivitet

## Registrere bil

### Funksjonelle krav:

1. Utleier skal kunne skrive inn bilens registreringsnummer
2. Utleier skal kunne spesifisere antall seter
3. Utleier skal kunne spesifisere om det er manuell eller automatgirkasse
4. Utleier skal kunne spesifisere hva slags drivstoff motoren bruker
5. Utleier skal kunne spesifisere status ved bilens registrering

### Ikke funksjonelle krav

1. Informasjonen lagres i database
2. systemet registrerer utleiestatus i databasen

## Betaling

Funksjonelle krav:

1. Skal kunne velge å leie bil med vipps
2. skal kunne velge å leie bil med kort

Ikke funksjonelle krav:

1. Sørge for at brukeren gjennomfører betalingen ved bruk av HTTPS eller Vipps for sikker betaling

## Leie bil

Funksjonelle krav:

1. Bruker skal kunne bestille bil ved å trykke på «Rent car with X» knappen
2. Bruker skal kunne se hvilke biler som er ledige ved å trykke på «Rent car»
3. Bruker skal kunne se alle biler som er registrert
4. Bruker skal kunne avbestille bilen

Ikke funksjonelle krav:

1. Bilens utleiestatus skal endres i databasen når den blir leid eller avbestilt
2. Utleier skal få en varsel når bilen er leid eller avbestilt
3. Utleier og kunde skal få en bekreftelse på Epost eller SMS når bilen er leid eller avbestilt



## Utleie av bil

### Funksjonelle krav:

1. Utleier skal kunne endre status på bil
2. Utleier skal kunne kansellere en ordre dersom det er 4 timer før bilen skal tas i bruk av en kunde

### Ikke funksjonelle krav:

1. SMS eller Epost bekreftelse skal bli sendt til utleier og kunde dersom bestilling blir kansellert

## Personas

Appen vår går ut på at folk som ønsker seg å låne bile eller ut låne bil til andre. Bruker kan søke etter biler til å låne så de kan gjøre

### Personas 1:

## Elina Eriksen



Famile er alt

Alder: **39**  
jobb: **lærer**  
Famili: **Married, kids.**  
Bosted: **halden**

### Mål:

- kan reise med barna rundt i Norge
- ha det gøy med barna
- bruke tid med familien

### Frustrasjoner


- kolektivtransport går ikke der familien vil dra
- kolektivtransport tar mye tid

### Bio

Eina er en kvinne som er har jobber masse, men hun liker å bruke tiden hun har til vare med familien sin. Hun liker barna sine, hun vil at det skal ha det gøy i helger og ferier


Elina Eriksen er en mor som jobber mye i hverdagen, hun har barn som for ikke mye tid med dem. Så hun prøver alltid å bruke så mye tid ho kan med barna i da helgene og i ferier. Så hun tenker å låne enn bil som er behagelige å sitte i til en stor Familie. Hun tenker å ta barna til tusenfryd og Bø sommerland.

## Personas 2:

 <p>James Abdi</p>	<p><b>Bio</b></p> <p>James er en nylig pensjonert mann som ønsker en bil innemellom for korte reiser til venner og familie.</p>	<p><b>Demographic info</b></p> <p>Age 72</p> <p>Location Lillehammer</p> <p>Family Status Gift</p> <p>Education level Master i historie</p> <p>Income level 600000</p> <p>+ Add field</p>
<p><b>Quote</b></p> <p>Safe enviroment, safe life.</p>	<p><b>Frustrations (pain points)</b></p> <p>For mye folk i byen og det er mange folk, mye rush.</p>	


Denne personen vil ha god bruk for applikasjonen der han behøver og besøke familien sin innimellom, leie bilene vil være kjekt for korte turer innimellom. James han er en mann som har nylig pensjonert seg, han tenker nå å låne en bil i mellomtida for besøk til barn og barnebarna sin sammen med sin kone, så dette ville være bra for han og kona å kunne låne en bil, det er enkelt der de kan gjøre alt via appen. For James som kunde er det viktig for han at applikasjonen er enkelt som mulig og at det er til stor hjelp for han til alle tider.

### Personas 3:

 <p>Ahmed Hussain</p>	<h4>Bio</h4> <p>Ahmed Hussain er en advokat som er ute etter en sjapp transportmiddel som ikke vil ta så lang tid. Hadde vært bra med en billig leiebil som kan benyttes til en del reiser uten store kostnader</p>	<h4>Info</h4> <p>Age 26</p> <p>Location Trondheim</p> <p>Family Status ugift</p> <p>Education level Master i rettsvitenskap</p> <p>Income level 800 000</p> <p>+ Add field</p>
--	---	--

Ahmed er ung kar har en del jobb i byen, men han synes transportmidler tar lang tid, og han jobber som advokat så han må være i god tid på jobben sånn at han ikke blir forsinket, og han tenker å låne en bli som ikke koster mye og som kan få han til jobben

Personas 4:



Ole Normann

**Bio**

Ole Normann er en 23 år gammel mann som nettopp er ferdig på høyskolen. I dag jobber han som Fotojournalist og har et behov for å komme seg rundt til alle tider i døgnet

**Info**

Age

23

Location

Oslo

Family Status

Singel

Education level

Bachelor i fotografi

Income level

450000

+ Add field

**Quote**

“Et bilde er en portal til fortiden”


**Mål**

- Finne en god løsning for å komme seg dit han skal.
- Ønsker å spare tid på reise
- Finne en transportmåte som gir han mulighet til å ta med masse utstyr

**Frustrasjoner**

- Å eie bil i Oslo er dyrt
- Å reise kollektivt er ikke tidsbesparende
- Vanskelig og ta med mye utstyr når man reiser kollektivt

Personas 5:

 <p>Adam Green</p>	<p><b>Info</b></p> <p>Age 39</p> <p>Location Wales</p> <p>Family Status Gift med barn</p> <p>Education level Frisør Svennebrev</p> <p>Income level 1000000</p>	<p><b>Quote</b></p> <p>“Skilled barbers aren't cheap. Cheap barbers aren't skilled.”</p>
<p><b>Bio</b></p> <p>Adam er en 39 år gammel mann med mange år som frisør under beltet. Han driver sin egen salong i Swansea og er svært suksessrik. Adam elsker å dra på ferie men reiser også mye med jobben, spesielt til norge.</p>	<p><b>Kontakt</b></p> <p>SoMe Twitter, Facebook, Snapchat</p> <p>Andre Mail, Telefon</p>	<p><b>Mål</b></p> <ul style="list-style-type: none"><li>- Kjøre rundt omkring i norge</li><li>- Komme seg til Frisørkonvensjoner smertefritt</li><li>- Alltid ha en bil tilgjengelig</li></ul> <p><b>Frustrasjoner</b></p> <ul style="list-style-type: none"><li>- Kollektivtransport krever for mye planlegging</li><li>- Kollektivtransport kjører ikke dit han vil</li></ul>

## User Stories-1

- Som kunde så vil jeg gjerne kunne logge inn i systemet
- Som kunde så vil jeg gjerne kunne se detaljer om bilen
- Som kunde så vil jeg gjerne kunne betale med vipps eller bank kort
- Som kunde så vil jeg gjerne kunne få bekreftelse på E-post eller på SMS
- Som kunde så vil jeg gjerne kunne søke på forskjellige kategorier på bilen
- Som kunde så vil jeg gjerne kunne registrere bil i systemet
- Som kunde så vil jeg gjerne kunne låne ut bilen min
- Som kunde så vil jeg gjerne kunne endre epost
- Som kunde så vil jeg gjerne kunne endre passord
- Som kunde så vil jeg gjerne kunne endre på person detaljer
- Som admin så vil jeg gjerne kunne se kundes betalinger
- Som admin så vil jeg gjerne kunne se endre på bestallinger
- Som admin så vil jeg gjerne kunne se kundes avbestille bestallinger
- Som admin så vil jeg gjerne kunne se kundes se alle biler
- Som admin så vil jeg gjerne kunne fjerner biler fra systemet
- Som admin så vil jeg gjerne kunne se kundes bestillinger

## User-stories-2

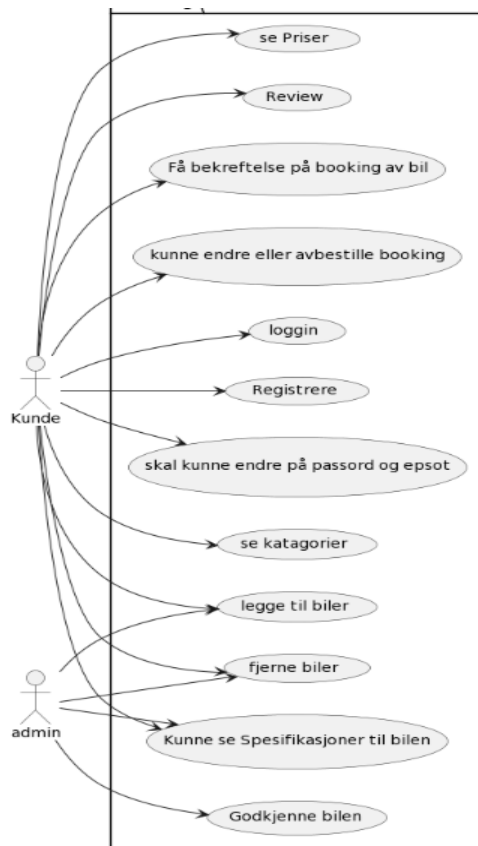
- Som bruker vil jeg kunne opprette en bruker
- Som bruker vil jeg kunne se ledige biler
- Som bruker vil jeg kunne bestille en bil
- Som bruker vil jeg kunne leie ut en bil
- Som bruker vil jeg kunne velge en ønsket bil
- Som bruker vil jeg kunne få bekreftelse på E-post/SMS
- Som admin vil jeg kunne se brukerens bestillinger
- Som admin vil jeg kunne endre bestillinger
- Som admin vil jeg kunne kansellere bestillinger
- Som admin vil jeg kunne svare til bruker når de har levert bilen
- Som admin vil jeg kunne svare på tilbakemeldinger

## User case:

Her er en USER CASE som viser hva en bruker og en admin kan gjøre i vårt system. Det finnes ting som en bruker ikke kan gjøre eller ikke har tilgang til, men som kan gjøres av en admin. Ett eksempel på det kan være at bruker ikke får registrert bilen, men det kan admin. Kunden kan logge inn og registrere seg i appen, kunden kan også se på spesifikasjoner av bilen altså detaljer om bilen, det kan også admin gjøre. Kunden skal kunne endre på eposten og passordet sitt med det kan ikke en admin gjøre for kunden det er kunden som har ansvar for det. Kunden skal kunne endre og avbestille bestilling selv, men hvis kunden trenger hjelp så kan admin hjelpe.



User case figur 1:



## Kode for user case figur 1:

@startuml

left to right direction

actor Kunde as g

actor "admin" as fc

package billoning{

usecase "loggin" as UC01

usecase "Registrere" as UC02

usecase "skal kunne endre på passord og epsot" as UC012

usecase "se katagorier" as UC1

usecase "se Priser" as UC2

usecase "Review" as UC3

usecase "legge til biler" as UC4

usecase "fjerne biler" as UC04

usecase "Få bekreftelse på booking av bil" as UC5

usecase "kunne endre eller avbestille booking" as UC05

usecase "Kunne se Spesifikasjoner til bilen" as UC6

usecase "Godkjenne bilen" as UC7

}

fc --> UC4

fc --> UC04

fc --> UC6

fc --> UC7

g --> UC01

g --> UC02

g --> UC012

g --> UC1

g --> UC2

g --> UC3

g --> UC4

g --> UC04

g --> UC5

g --> UC05

DriveMe 2022

g --> UC6

@enduml

## Use-Case 2

Navn: Klient

Use-Case: US1

Beskrivelse: Dette viser da hvordan brukeren kan benytte systemet og applikasjonene.

Utløser: Skriver inn applikasjonen på Appstore eller google butikk.

Pre-betingelser:

- Har internett
- Bruker har skrevet inn riktig navn/adresse på applikasjonen.

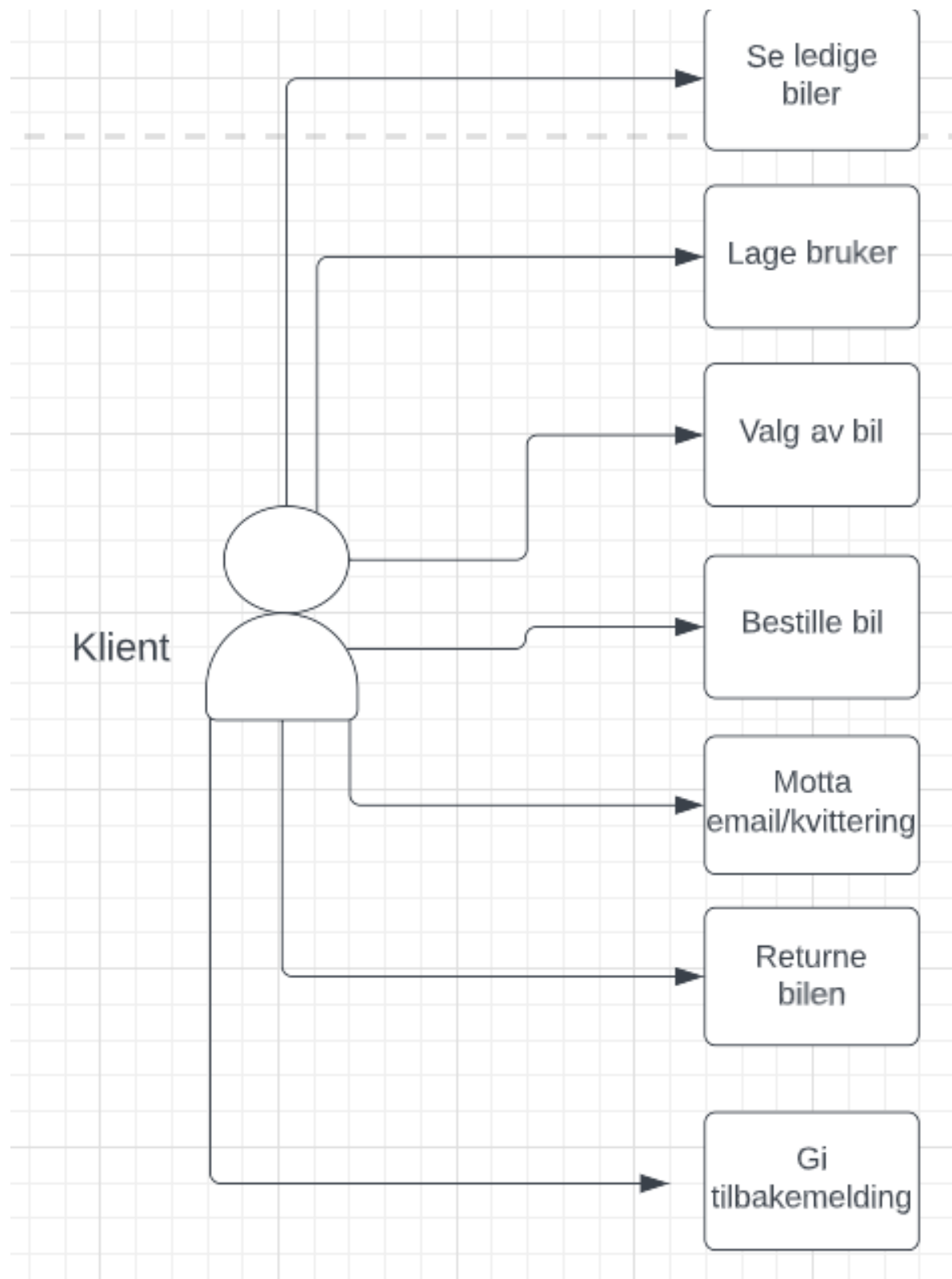
Post-betingelser:

- Se ledige biler
- Lage bruker
- Valg av bil
- Bestille bil
- Motta kvittering/email
- Returnere bil
- Gi tilbakemelding

Normal hendelse flyt:

1. En bruker åpner websiden eller appstore eller google butikk
2. Server får da informasjonen om dato og tidspunktet av database etter da spørringen.
3. Server sender HTML dokumentet til klient sånn at de får opp google søket om de brukte dette
4. Systemet sender da brukerne til enten appstore eller Google butikken
5. Bruker lager da bruker for å benytte appen
6. Bruker blir sendt videre til menyen, ledige biler eller leie ut en bil.
7. Bruker benytter seg av menyen uavhengig av situasjonen
8. Use-case slutter.

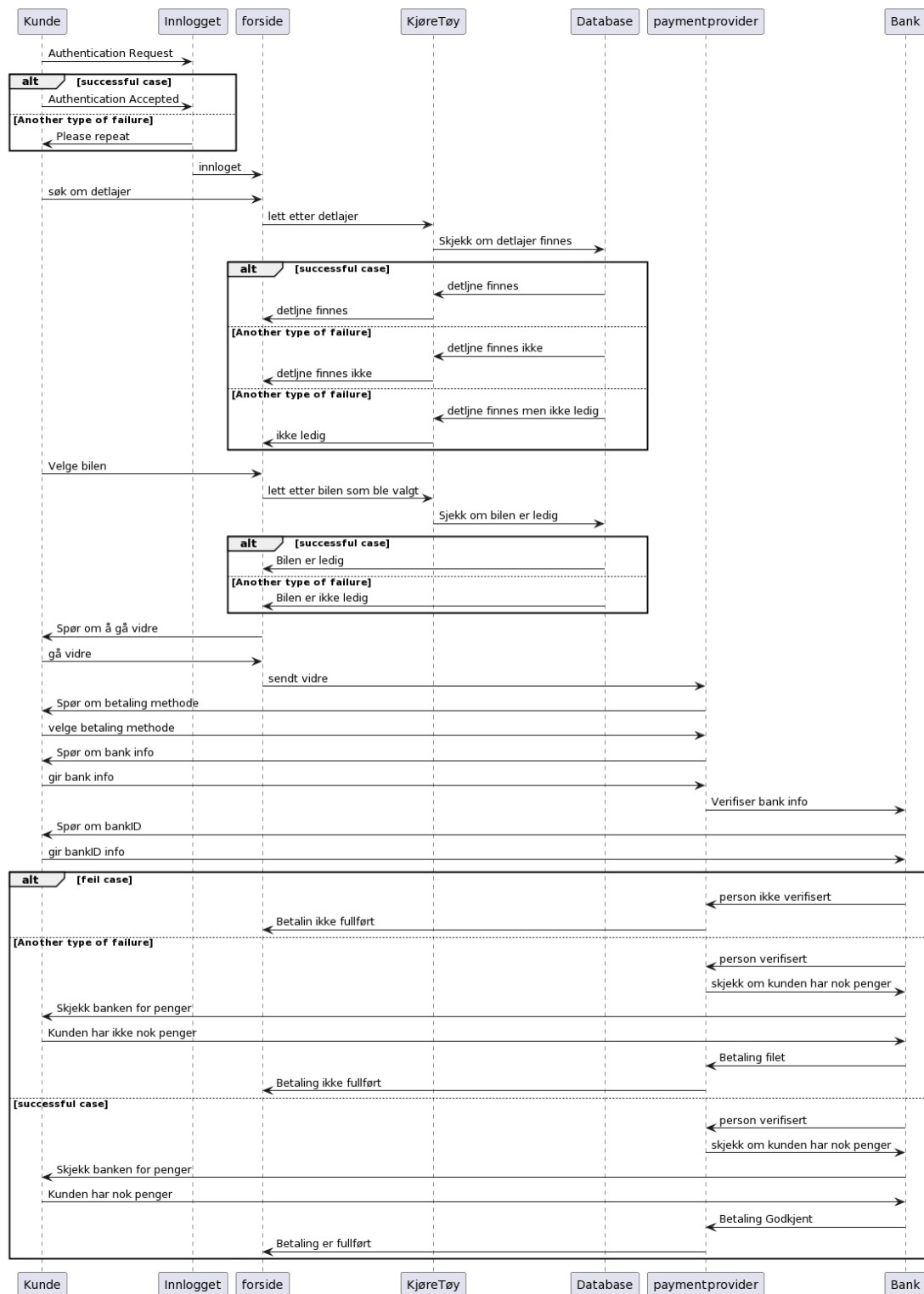
## Use-case 2 Figur



## Sekvens diagram

Dette sekvens diagrammet viser hvordan systemet fungerer og hvordan data går gjennom systemet., og hvordan systemet håndtere ting, altså den viser hvordan ting kommer til å kjøre. Hvis kunde logger inn med feil passord så vil brukeren få en respons som sier «prøv igjen». Med andre ord sekvens diagrammet viser hvordan data i et system samhandler i en prosess. Den viser data som går mellom brukeren og systemet og i hvilken rekkefølge de kjøres.

Sekvens diagram figur 1:





Kode for figur 1:

@startuml

Kunde -> Innlogget : Authentication Request

alt successful case

Kunde -> Innlogget: Authentication Accepted

else Another type of failure

Innlogget -> Kunde: Please repeat

end

Innlogget -> forside: innloget

Kunde -> forside: søk om detaljer

forside -> KjøreTøy: lett etter detaljer

KjøreTøy -> Database: Skjekk om detaljer finnes

alt successful case

Database -> KjøreTøy: detljne finnes

KjøreTøy -> forside: detljne finnes

else Another type of failure

Database -> KjøreTøy: detljne finnes ikke

KjøreTøy -> forside: detljne finnes ikke

else Another type of failure

Database -> KjøreTøy: detljne finnes men ikke ledig

KjøreTøy -> forside: ikke ledig

end

Kunde -> forside: Velge bilen

forside -> KjøreTøy: lett etter bilen som ble valgt

KjøreTøy -> Database: Sjekk om bilen er ledig

alt successful case

Database -> forside: Bilen er ledig

else Another type of failure

Database -> forside: Bilen er ikke ledig

end

forside -> Kunde: Spør om å gå videre

Kunde -> forside: gå videre

forside -> paymentprovider: sendt videre

paymentprovider -> Kunde: Spør om betaling methode

Kunde -> paymentprovider: velge betaling methode

paymentprovider -> Kunde: Spør om bank info

Kunde -> paymentprovider: gir bank info

paymentprovider -> Bank: Verifiser bank info

Bank -> Kunde: Spør om bankID

Kunde -> Bank: gir bankID info

alt feil case

Bank -> paymentprovider: person ikke verifisert

paymentprovider -> forside: Betalin ikke fullført

else Another type of failure

Bank -> paymentprovider: person verifisert

paymentprovider -> Bank: sjekk om kunden har nok penger

Bank -> Kunde: Sjekk banken for penger

Kunde -> Bank: Kunden har ikke nok penger

Bank -> paymentprovider: Betaling filet

paymentprovider -> forside: Betaling ikke fullført

else successful case

Bank -> paymentprovider: person verifisert

paymentprovider -> Bank: sjekk om kunden har nok penger

Bank -> Kunde: Sjekk banken for penger

Kunde -> Bank: Kunden har nok penger

Bank -> paymentprovider: Betaling Godkjent

paymentprovider -> forside: Betaling er fullført

end

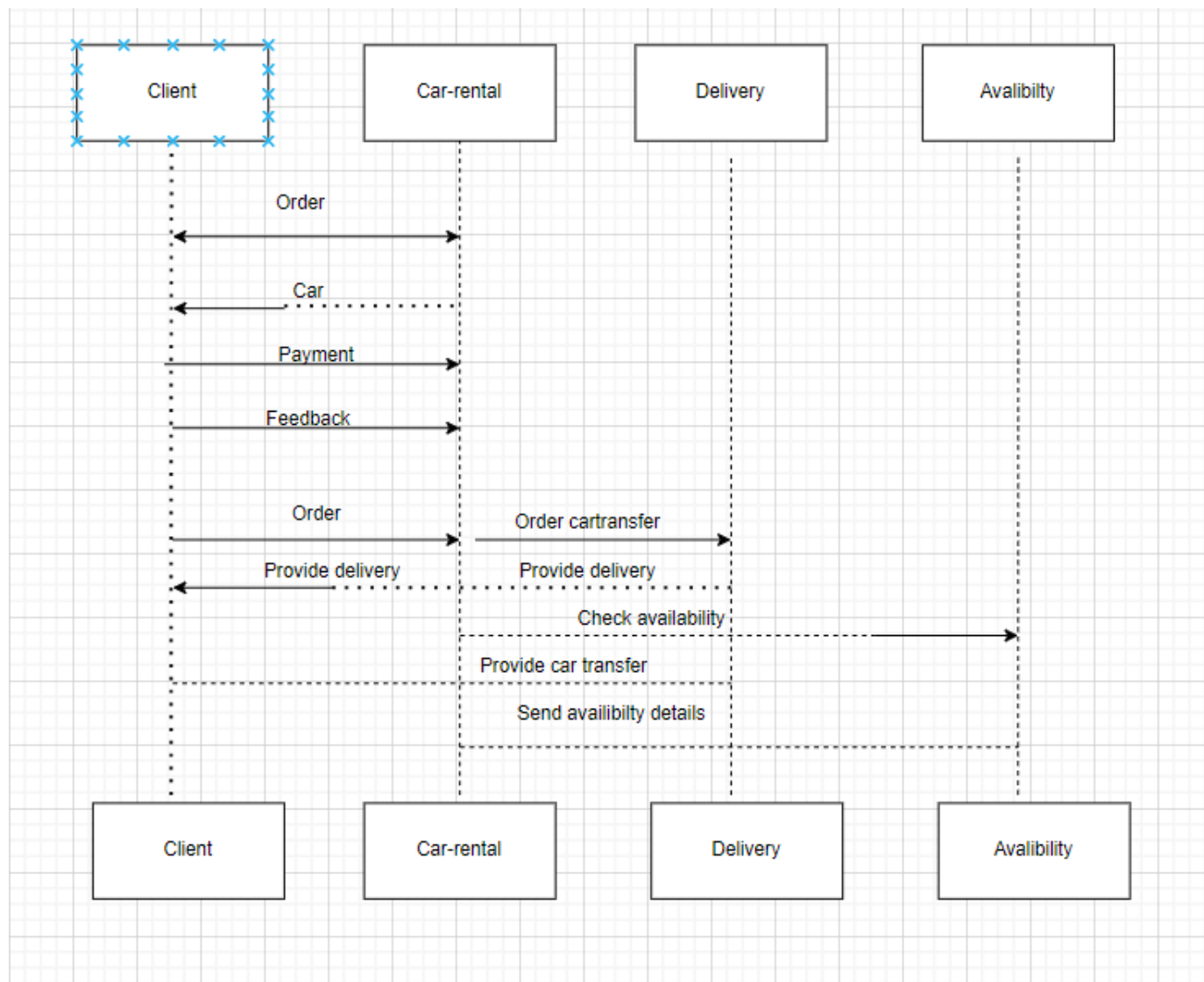
@enduml

## Sekvens diagram figur 2:

### Bestilling av en bil

Her er et sekvens-diagram om brukere/kunder som skal bestille en bil. Her viser diagrammet prosessen fra å bestille en bil, til betalingen og bestillingen. Også i dette tilfellet appen/nettsiden som viser deg hvilke biler som er tilgjengelige og hvordan informasjonen når kunden

Figuren viser når kunden skal foreta seg bestilling av en lånebil. Systemet viser da prosessen kunden må foreta seg for å nå målet sitt, her er andre elementer på plass som selskapet. Til slutt mottar selskapet om ledige biler, som de da sender informasjon videre til kunden.



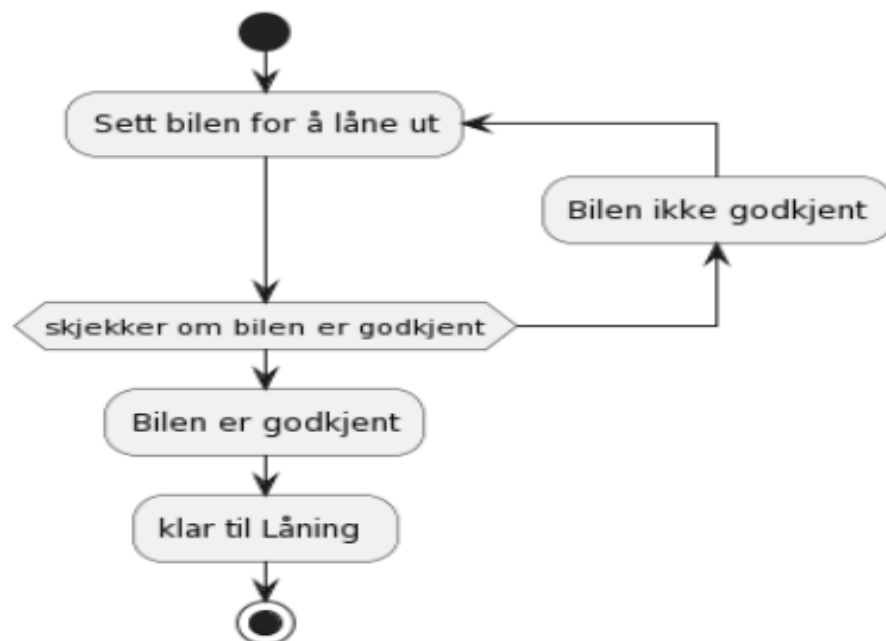


## Aktivitets diagram

### Låne ut bilen

Hvis kunden vil låne ut sin bil må det være enkelt for h\*n å gjøre det. Kunden må logge seg inn systemet deretter trykke på “Registrer bil” knappen å låne ut bilen, deretter sette bilen i systemet. Etter dette vil systemet sjekke om bilen er registrert på riktig måte. Kunden kan ikke selv si at bilen er godkjent. Hvis bilen er godkjent så blir man sendt videre, men vis bilen er ikke godkjent så får kunden en melding på det.

Aktivitets diagram figur låne ut bilen



Kode for figur låne ut bilen:

```
@startuml
```

```
start
```

```
repeat :Sett bilen for å låne ut;
```

```
backward:Bilen ikke godkjent;
```

repeat while (skjekker om bilen er godkjent)

:Bilen er godkjent;

:Låne ;

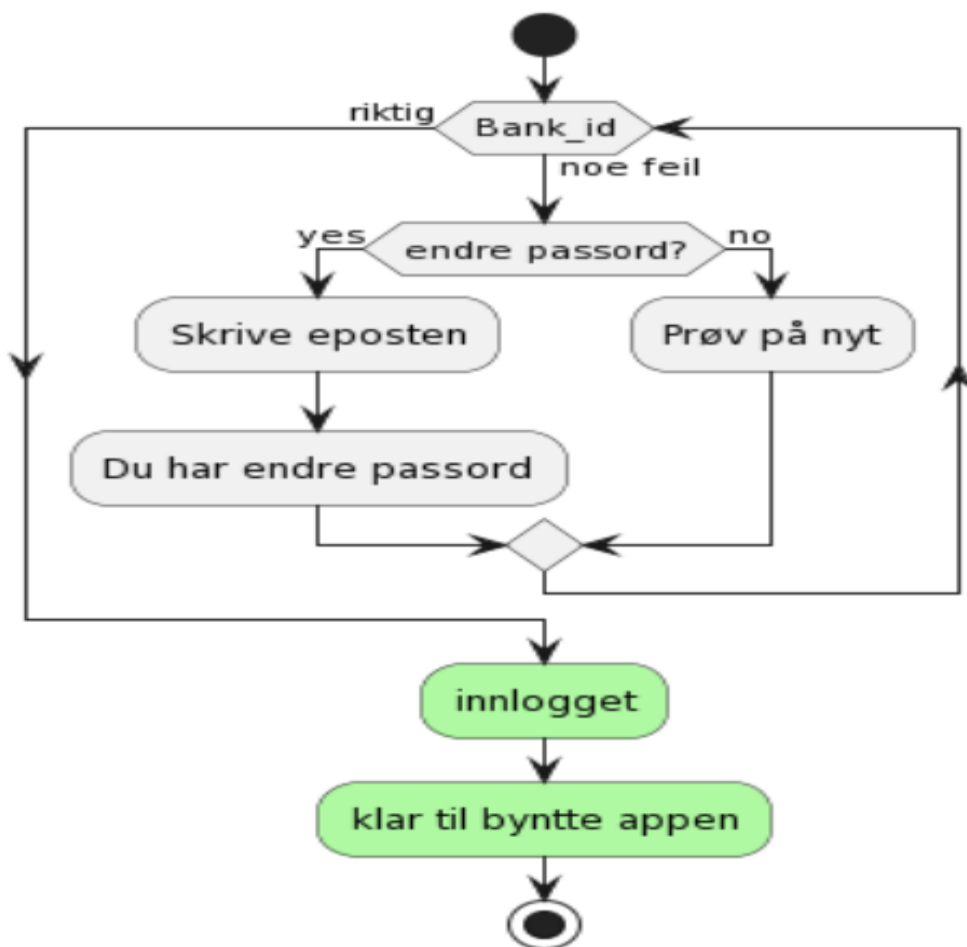
stop

@enduml

## Innlogging

Hvis Kunden vil logge seg inn i systemet så det skal være enkelt for brukeren å gjøre det. Det første kunde må gjøre er å trykke på "BankID" knappen, deretter blir brukeren sjekket og hvis alt er som det skal så blir brukeren logget inn, hvis ikke så vil brukeren få en melding om at BankID feilet

Aktivitets diagram figur innlogging



Kode for Aktivitets diagram figur innlogging:

```
while (Bank_id) is (noe feil)
```



if (endre passord?) then (yes)

:Skrive eposten;

:Du har endre passord;

else (no)

:Prøv på nytt;

endif

endwhile (riktig)

#palegreen:innlogget;

#palegreen:klar til byntte appen;

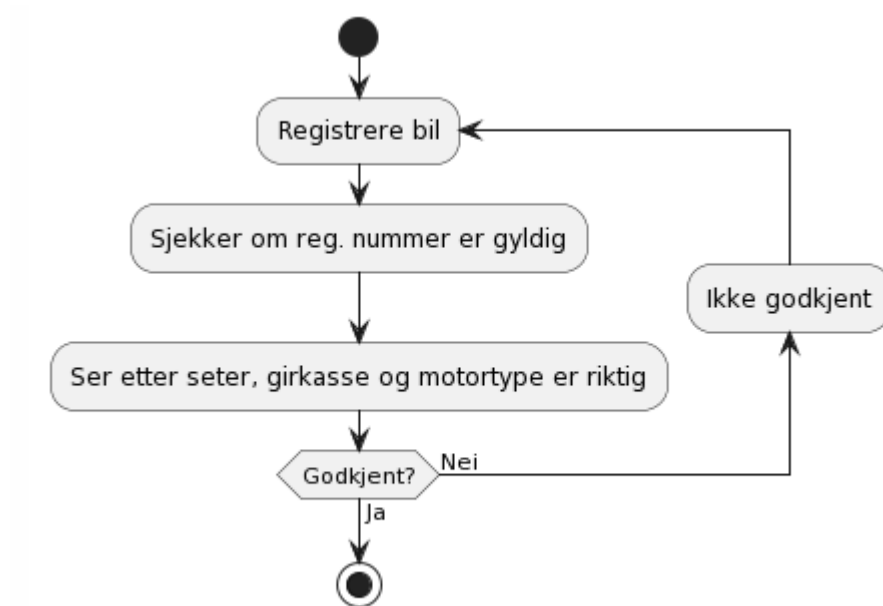
stop

@enduml

## Bil registrering

For at kunden skal kunne leie en bil så sjekker vi om registreringsnummeret tilsvarer det vi har i vårt system. Er dette godkjent vil kunden skrive inn antall seter, girkasse og motortype er gyldig, er dette riktig vil bilen bli lagt ut i applikasjonen og andre får muligheten til å leie bilen.

### Aktivitetsdiagram figur registrere bil



### Kode for registrering av bil

```

@startuml
start
repeat :Registrere bil;
:Sjekker om reg. nummer er gyldig;
:Ser etter seter, girkasse og motortype er riktig;
backward :Ikke godkjent;

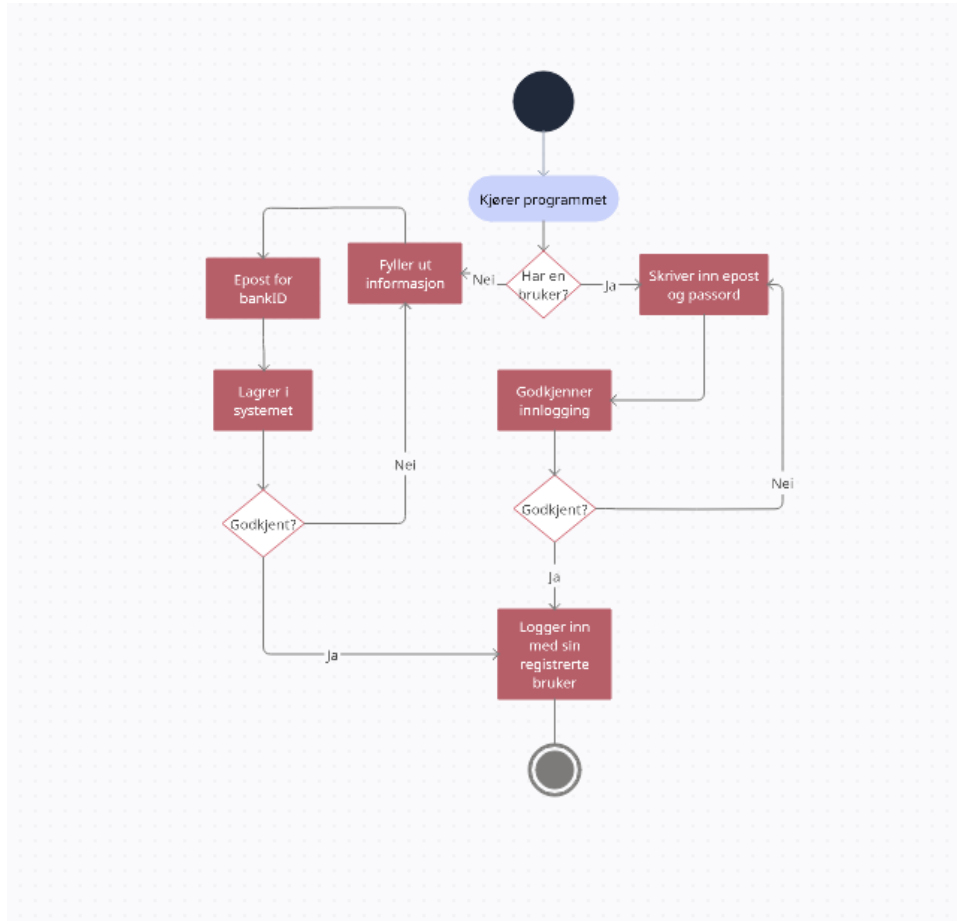
repeat while (Godkjent?) is (Nei) not (Ja)
stop
@enduml
  
```

## Registrering av bruker

For registrering av bruker vil kunden fylle ut skjemaet som kommer opp, dette inkluderer: fornavn, etternavn, telefonnummer og epost. Brukeren vil da få en epost som er en videre følger, på den nye siden vil han skrive inn et passord som han vil benytte, han må også logge inn med enten bankID på mobil eller med kodebrikke. Er dette riktig uten feilmelding vil han være logget inn og kan benytte applikasjonen. For at kunden skal kunne logge inn må han

bruke epost adressen som ble brukt i tillegg til passordet sitt. Da vil han få tilgang til applikasjonen.

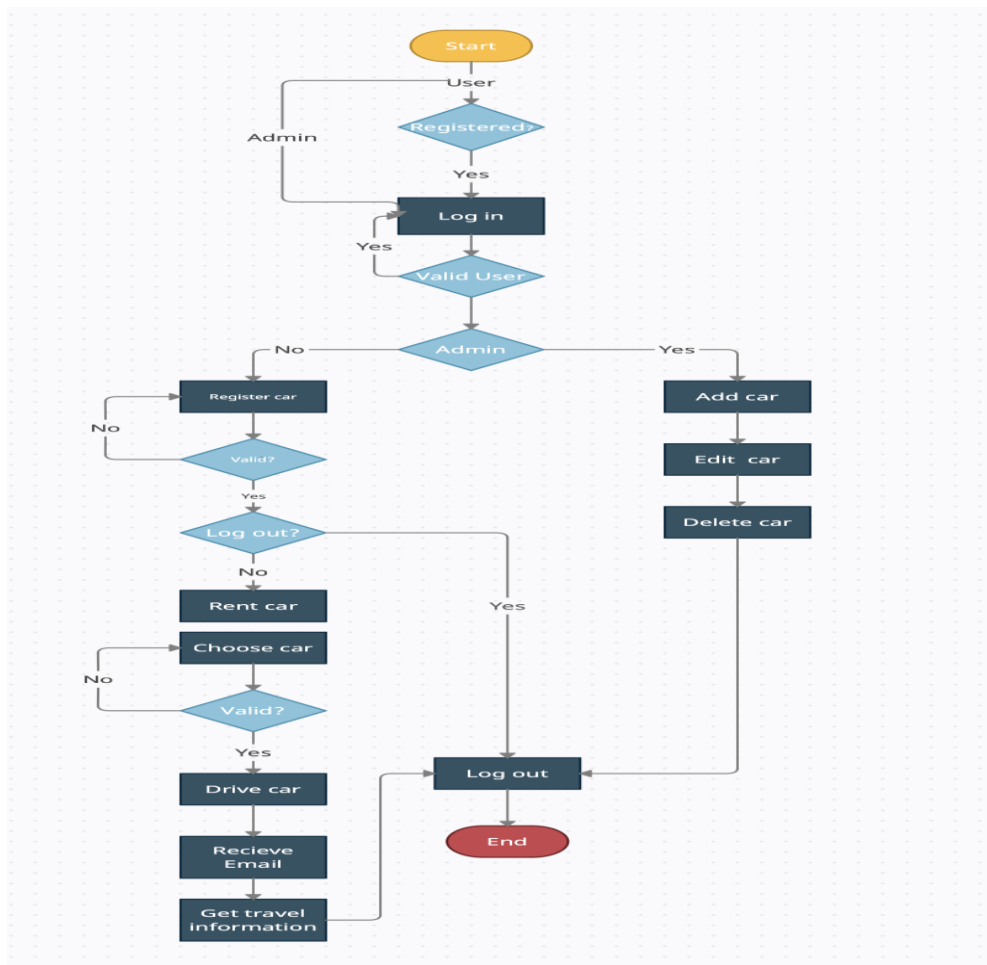
Aktivitetsdiagram figur for registrering av bruker



### Dataflyt diagram :

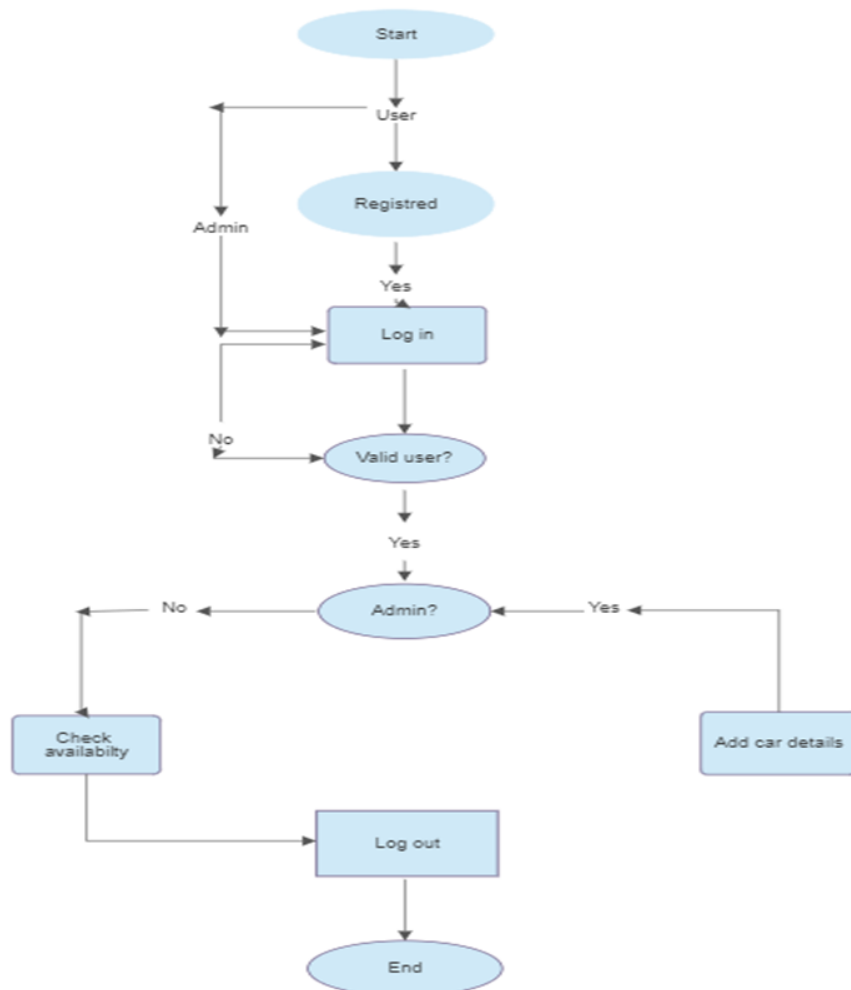
Denne figuren ser etter om en bruker er registrert i systemet, hvis ikke så må den registreres, ellers vil du kunne logge inn. Har et system som sjekker om brukernavn og passord tilsvarer det vi har i systemet, hvis ja så fortsettes det videre, eller prøv på nytt. Tilslutt har vi en som sjekker om en admin eller bruker logger inn, dersom det er en bruker vil han kunne registrere en bil, leie bil, velge bil osv.. Admin vil da kunne få endre bilene i systemet og legge til, han kan også slette den.

Brukerne må ha fylt 18år og har førerkort klasse B med gyldig legitimasjon, dette sjekkes når en bruker blir registrert.



## Dataflyt diagram 2:

Denne figuren sjekker om brukeren er registrert og om det er ledige biler tilgjengelig, også sjekker den om brukeren har en gyldig bruker i appen, vis ja kan man gå videre. I denne figuren, så får vi vite om kunden har en gyldig bruker og om de kan få sjekke om det finnes ledige biler. Først og fremst så må alle kundene være over 18 år og ha gyldig legitimasjon, dette sjekkes når de lager seg en bruker. I dette tilfellet har de enten en bruker som viser at de har førerkort og har gyldig bruker, ellers så har de ikke gyldig førerkort eller ikke gyldig bruker. Figuren viser da til slutt hvor mange biler som er tilgjengelige.



## Problemstilling og domenet:

Hvordan skal vi forhindre at bileiere bare bruker bilen sin kun sporadisk og til bare gitte tidspunkt av døgnet?

Det er ikke lønnsomt om du skal ha en bil som bare blir brukt i en kort periode for å så da selge den videre, du taper en del da bilens verdi synker. Da har vi kommet med en løsning, vi har utviklet et system som gjør det mulig for deg som har en bil kan låne ute bilen din og deg som ikke har en bil, kan få låne en bil via appen. For deg som låner, så er det lønnsomt å låne en bil når du vil, billigere vis du ikke ønsker å ha en bil eller ikke tilgang til det innimellom. Da kommer appen (Driveme) inn og kan løse problemene dine, om du skal på et møte, jobb, besøk og andre ting man ønsker. Ved noen få klikk så er du unna av å velge hvilken bil du skal kjøre i de neste timene. Det skal være enkelt for deg som bruker å få laget en bruker for å så da logge deg inn med bank id og velge bilen du ønsker deg. Via appen får du da vite de tilgjengelige bilene som er i nærheten og da velger du bilen som måtte passe deg, du åpner bilen med appen. Bilnøkkelen er da lagd ved at du åpner via Bluetooth.

Domenet, omgivelsene og miljøet er jo lagt for storbyer og folk som trenger en bil i en kort periode. Hvem vi henvender også til er folk som er over 18 år og har førerkort. Behovet rundt prosjektet, tilfredsstille behovet til kunden, ved å kunne låne en bil etter behovet deres.

## Løsning til prosjektgruppen

Det går ut på at den løser problemene i problemstilling da vi har lagd en prototype for kundene, den viser da hvordan appen skal fungere da vi har forskjellige funksjoner som er tilegnet forskjellige brukere. Som for eksempel om du har bil skal du benytte deg en knapp på skjermen din som viser låne ut bil for eksempel, eller om du ønsker å låne deg en bil. Dette kommer fram i applikasjonen og du vil også kunne velge mellom noen få forskjellige biler som du ønsker selv. Den håndterer de forskjellige elementene i problemstilling da appen løser problemene du har ved å låne en lånebil når du måtte ønske deg. Du må også være over 18 år og ha førerkort for å benytte

deg av tjenestene. Dette sjekkes via bank id. Dette skal være enkelt og greit for deg som bruker.

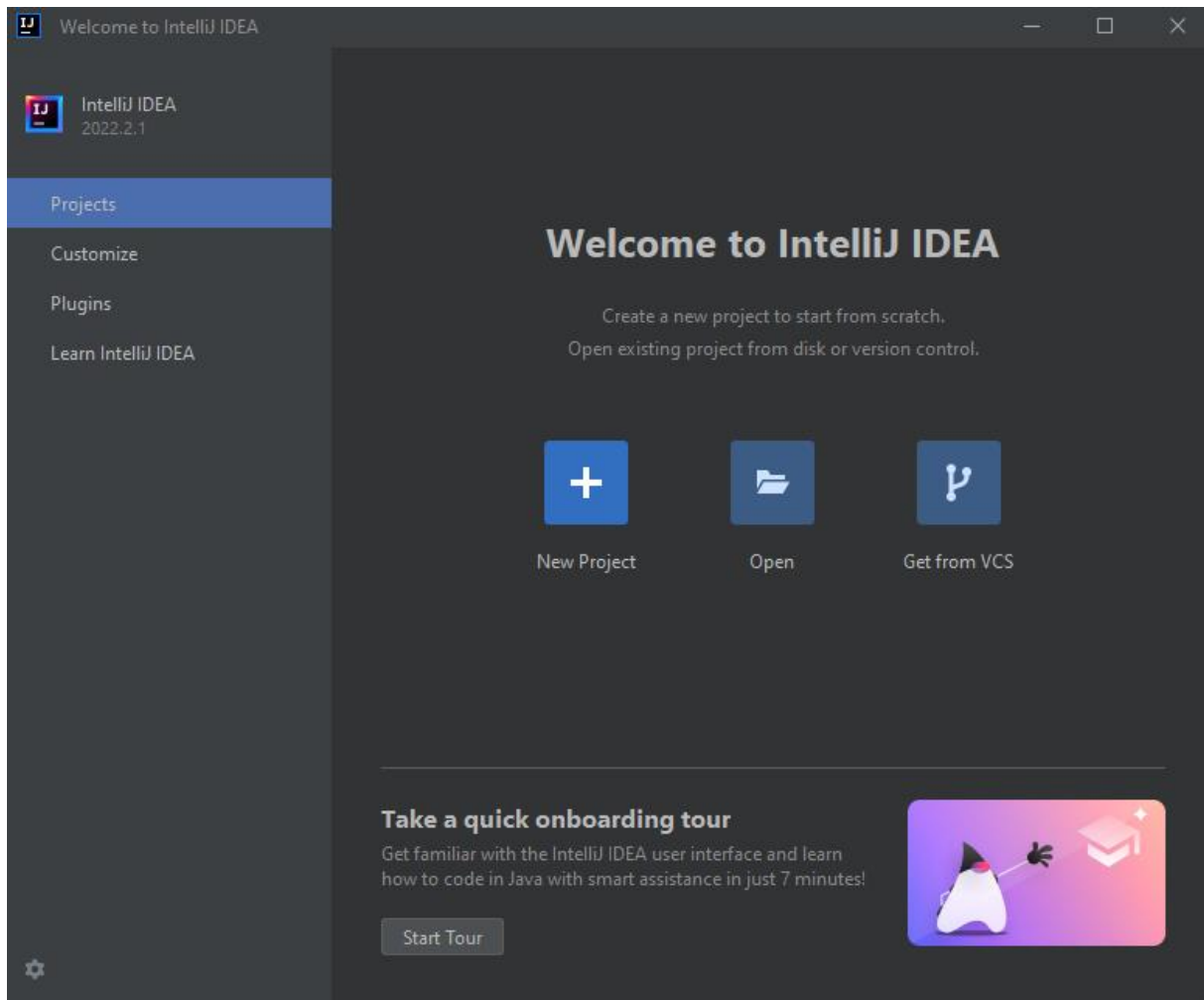
## Prototype

Protoypen vårt har blitt laget ved bruk av Java forms. Vi lagde den på plattformen IntelliJ IDEA. Java Forms gir brukeren en ganske lett GUI som man kan sette opp sitt program i. Hvis man vil jobbe videre med protoypen, eller vil bare lese kildekoden så kan man lett gjøre det ved bruk av IntelliJ. For å utføre testing så brukte vi «dependenciene»; JUnit Jupiter API (scope: test, version 5.7.2), JUnit Jupiter engine (scope: test, version 5.7.2), JUnit Jupiter params (scope: test, version 5.7.2) og til slutt maven-failsafe-plugin (versjon 2.22.2). Vi brukte også en siste dependancy kalt gson. Denne lot oss serialisere og deserialisere objekter til JSON så vi kunne skrive/lese til/fra fil. IntelliJ kan bli nedlastet her: <https://www.jetbrains.com/idea/download>.

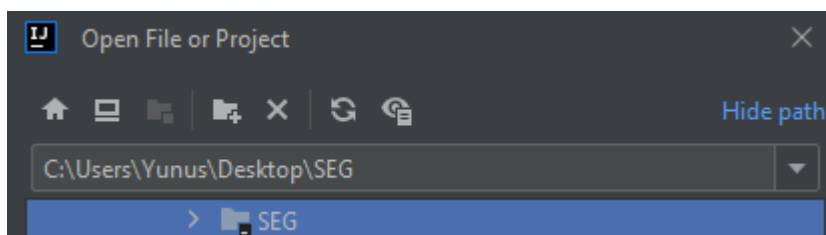
### Veiledning for å åpne prosjektet (direkte metode)

1. Trykk «Open» på IntelliJ idea:

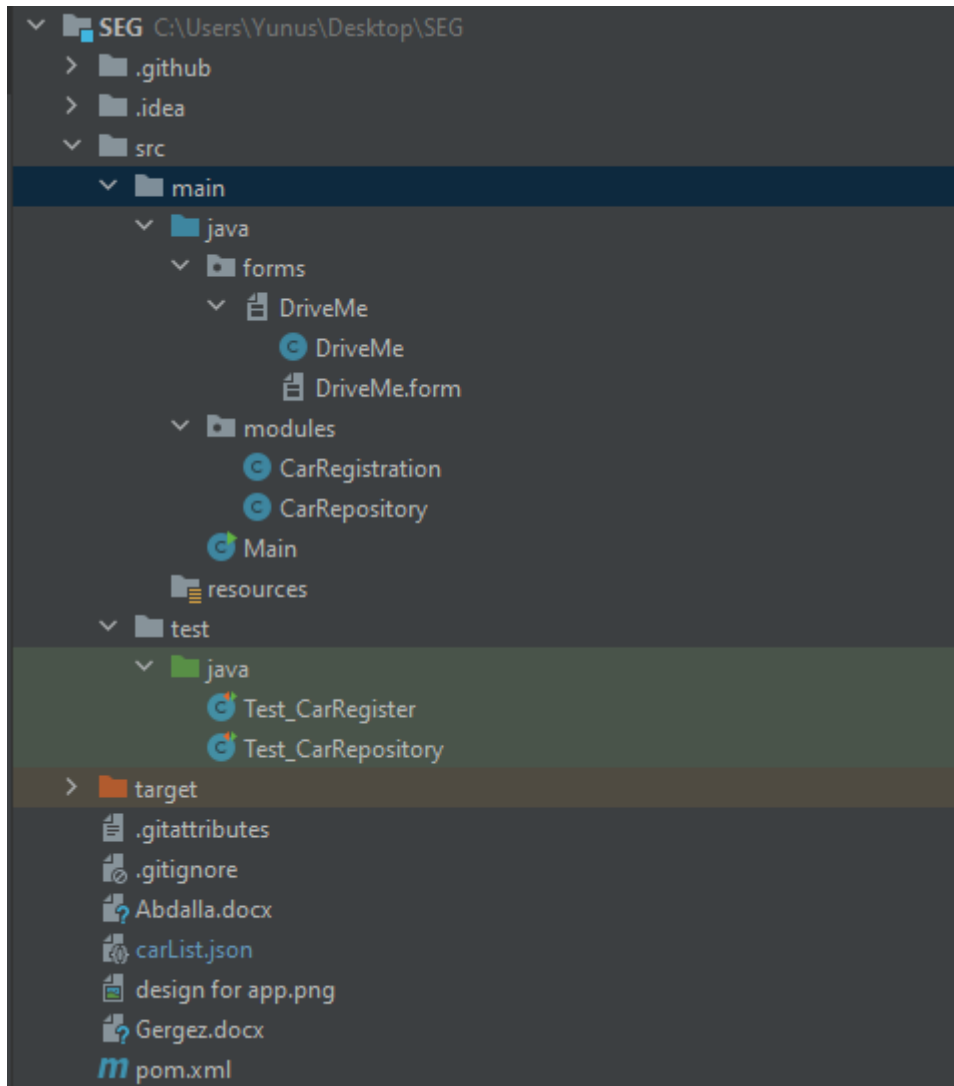




2. Finn frem prosjektmappen (kalt SEG):

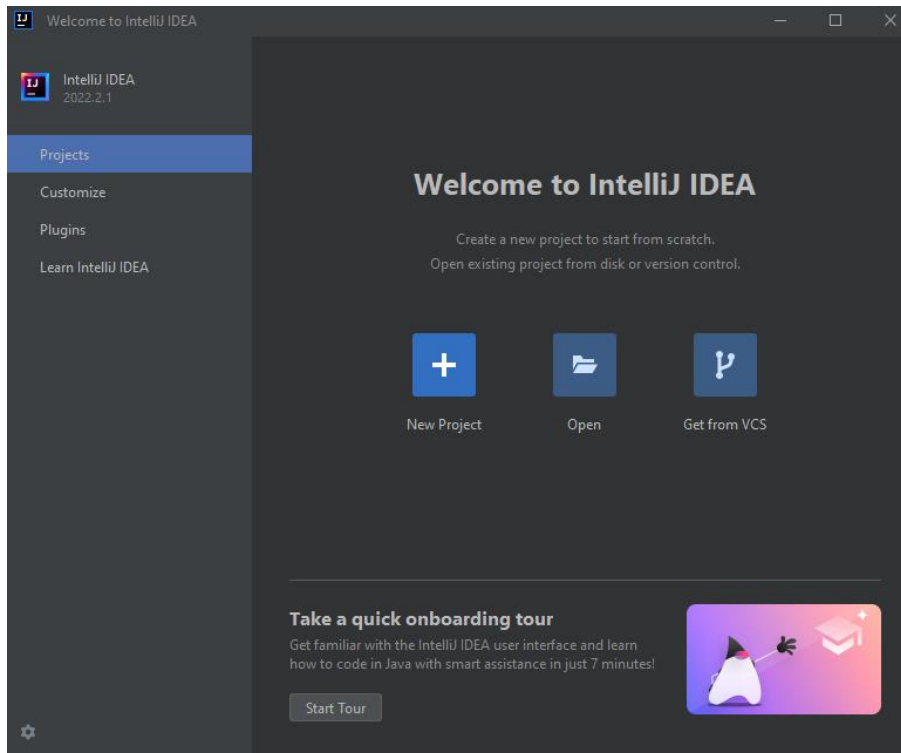


3. Så lett som det så skal det være mulig å se på kildekoden/jobbe videre med prototypen:

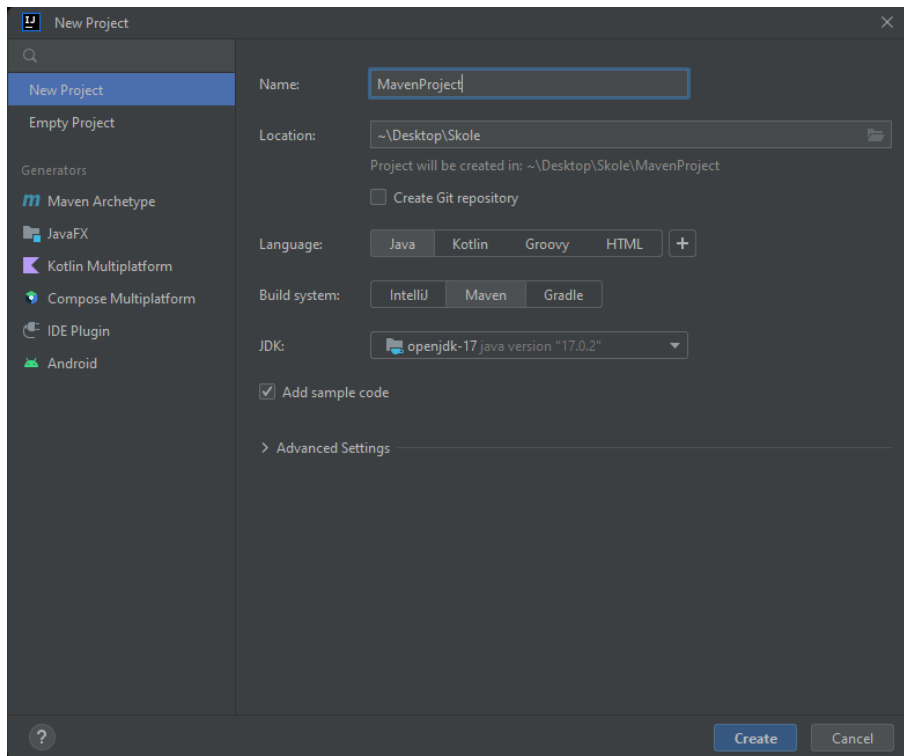


## Veiledning for å sette opp Maven (i tilfelle 7.1 ikke fungerer)

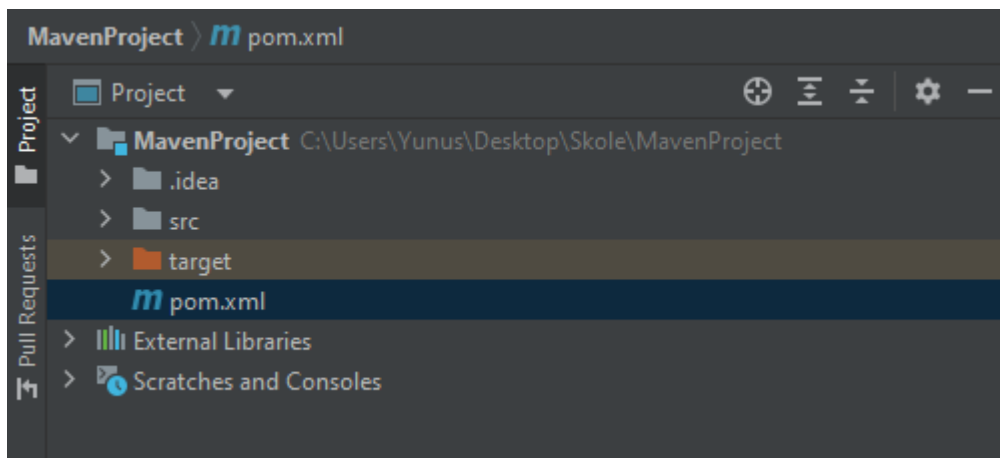
1. Trykk «New Project» på IntelliJ.



2. Her så skal du velge egendefinert navn og lokasjon. Deretter velger du Java som språk, og Maven som build system. Behold versjon 17 på openjdk. Trykk create når ferdig.

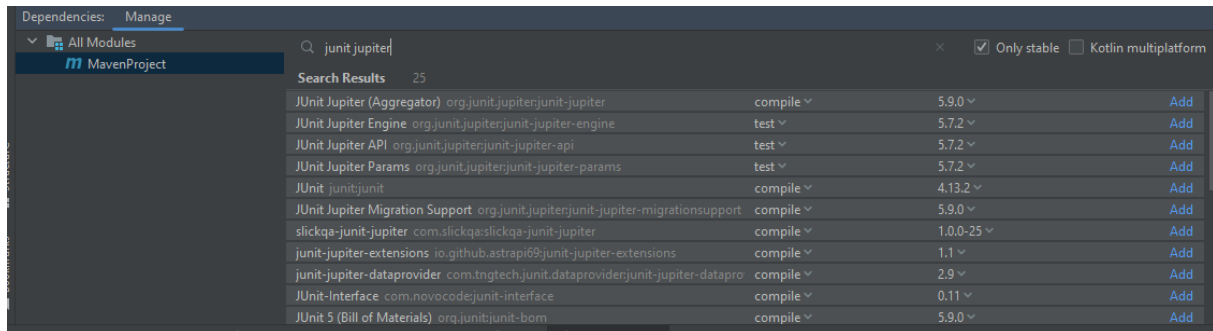


3. Nå skal du over til pom.xml som du kan finne på listen til siden.

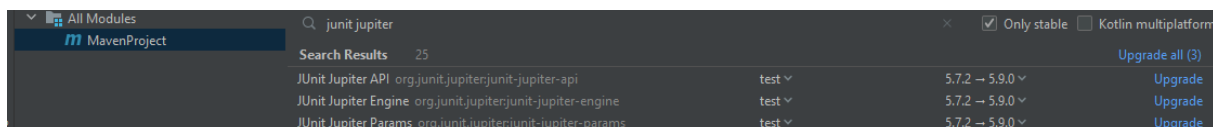


4. Du kan enten kopiere koden under og build prosjektet, eller manuelt legge til dependencies. Koden ligger på side 23. Ved å dobbel trykke «bildet» så får du

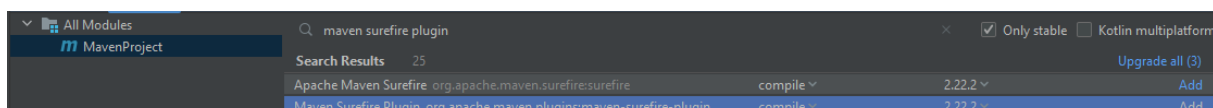
muligheten til å kopiere (usikker på hvordan funksjonaliteten fungerer på pdf). Trykk dependencies som finnes nederst i vinduet. Her søker du på «junit jupiter».



- Bytt fra compile til test på; engine, api og params. Du skal i tillegg sette alle 3 versjonene til 5.7.2. Deretter kan du trykke add på hver av dem:



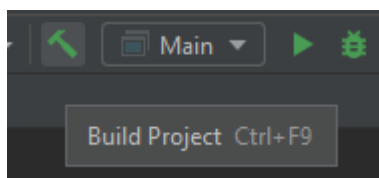
- Nå søker du etter maven surefire plugin og legger til den (denne blir brukt til github actions workflow om du vil sette om det. Noe vi gjorde i vårt repository på github):



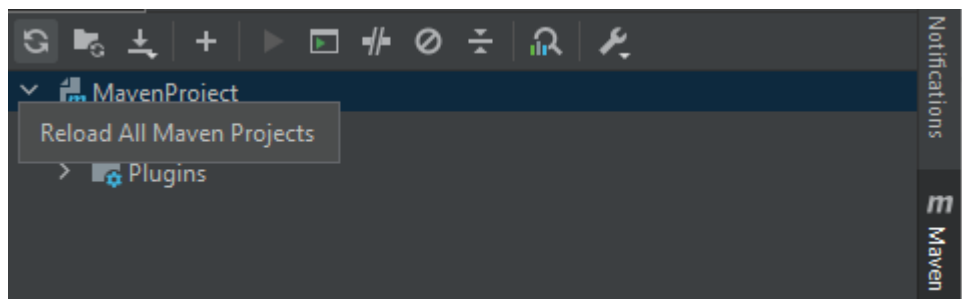
7. Til slutt så søker du etter gson og legger til den:



8. Trykk nå på “Build Project” som du kan finne øverst til høyre. Du kan også bruke CTRL + F9 for å oppnå samme resultat.

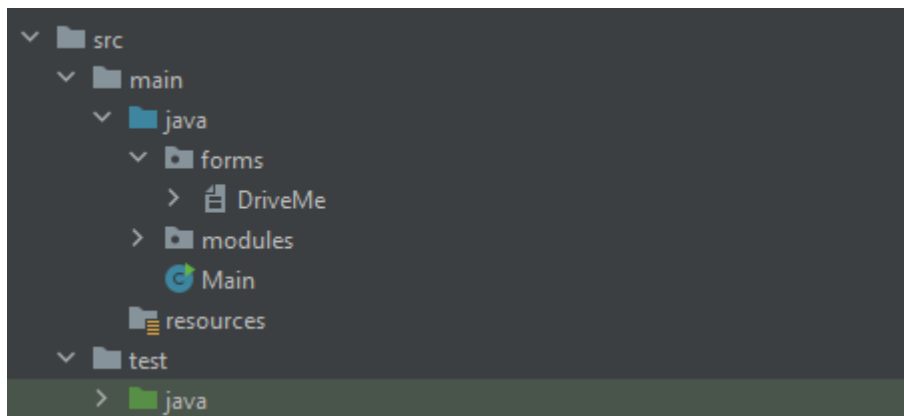


9. Nå er alt klart. Hvis json for en eller annen grunn ikke fungerer (slik som det gjorde med oss på starten), så kan du fikse det ved å gå over til «maven» helt til høyre i vinduet og trykke på «reload» knappen:



### Litt om prototypen

Protoypekoden er delt opp i 2 hovedmapper for kildekoden, og 2 egne javafiler for testing. Det var tidligere en egen mappe for Interfaces, som vi prøvde å sette opp funksjoner som vi kunne kalle til for å skrive til fil/lese fra fil. Dette fungerte dessverre ikke slik som vi ville, og har derfor blitt fjernet i sluttversjonen av protoypen. Endringene hvor vi brukte FileHandler og slikt kan finnes i git endringene. Vi fikk til eventuelt til å skrive til fil, men dessverre så fikk vi ikke til å lese fra fil. Dette kan finnes i versjon 3.4



I forms mappen så kan vi finne både forms filen som ble brukt til å designe GUI-en. I tillegg så er javafilen som kontrollerer forms i samme sted. Denne javafilen kontrollerer alt fra hva som blir vist, når det blir vist og hva som skjer når vi trykker på knapper og samhandler med protoypen.

I modules mappen så kan vi se begge klassene som har blitt brukt; CarRegistration og CarRepository. Først så lages et CarRepository som blir brukt til å lagre bilene som blir laget ved bruk av CarRegistration.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>oblig2</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>16</maven.compiler.source>
    <maven.compiler.target>16</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.7.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>5.7.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-params</artifactId>
      <version>5.7.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.10</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-failsafe-plugin</artifactId>
        <version>2.22.2</version>
      </plugin>
    </plugins>
  </build>

</project>

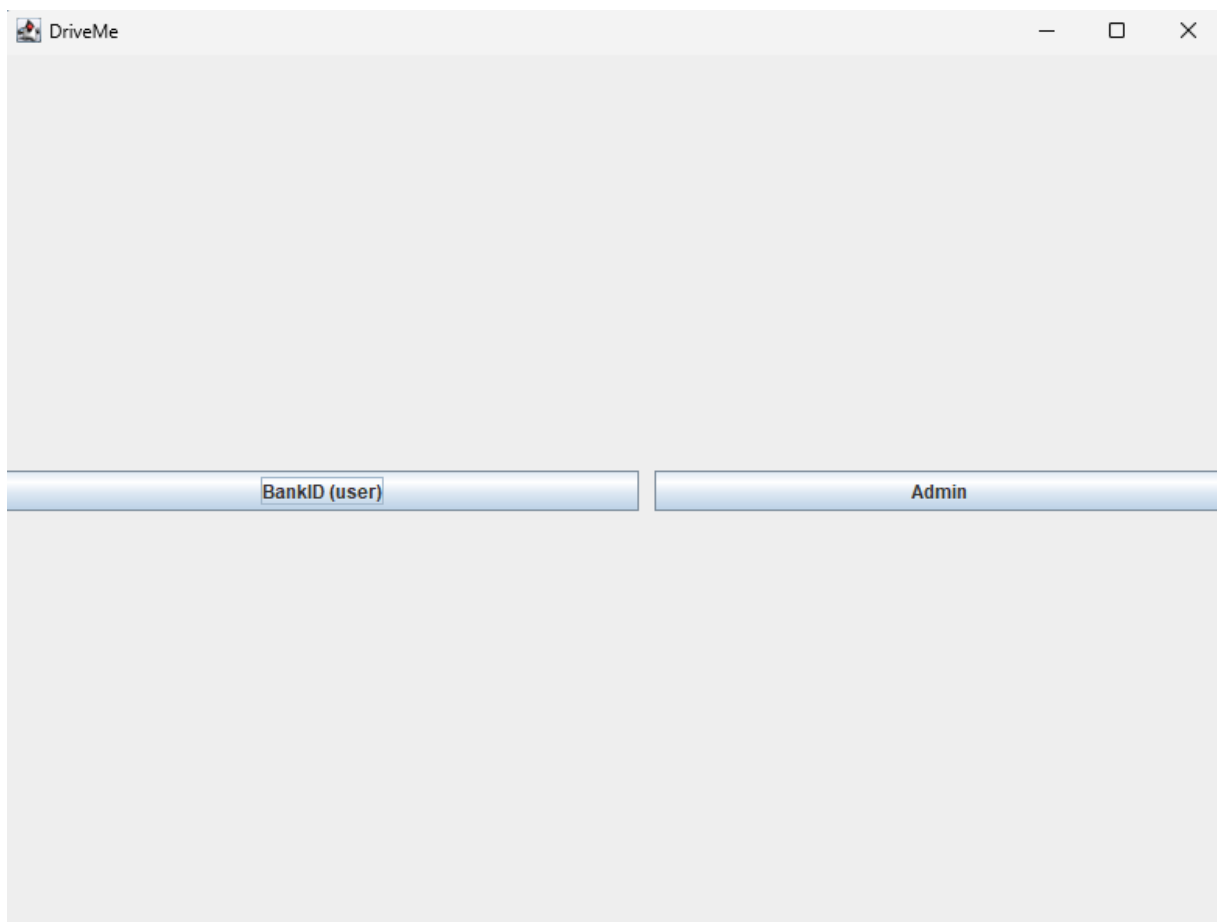
```



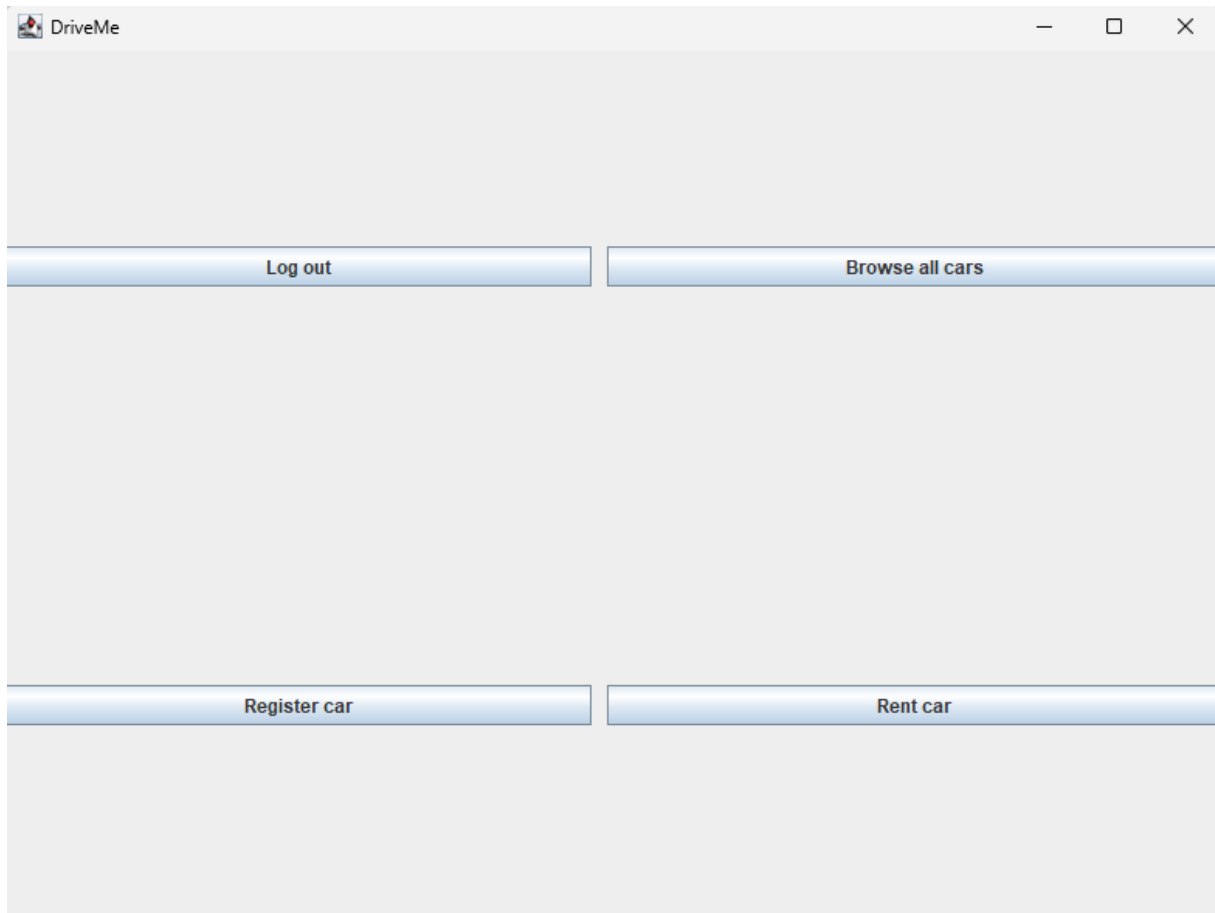
## Prototypens layout og funksjon

*Du kjører programmet ved å kjøre (CTRL+SHIFT+F10) på Main.java filen som befinner seg i src/main/java mappen.*

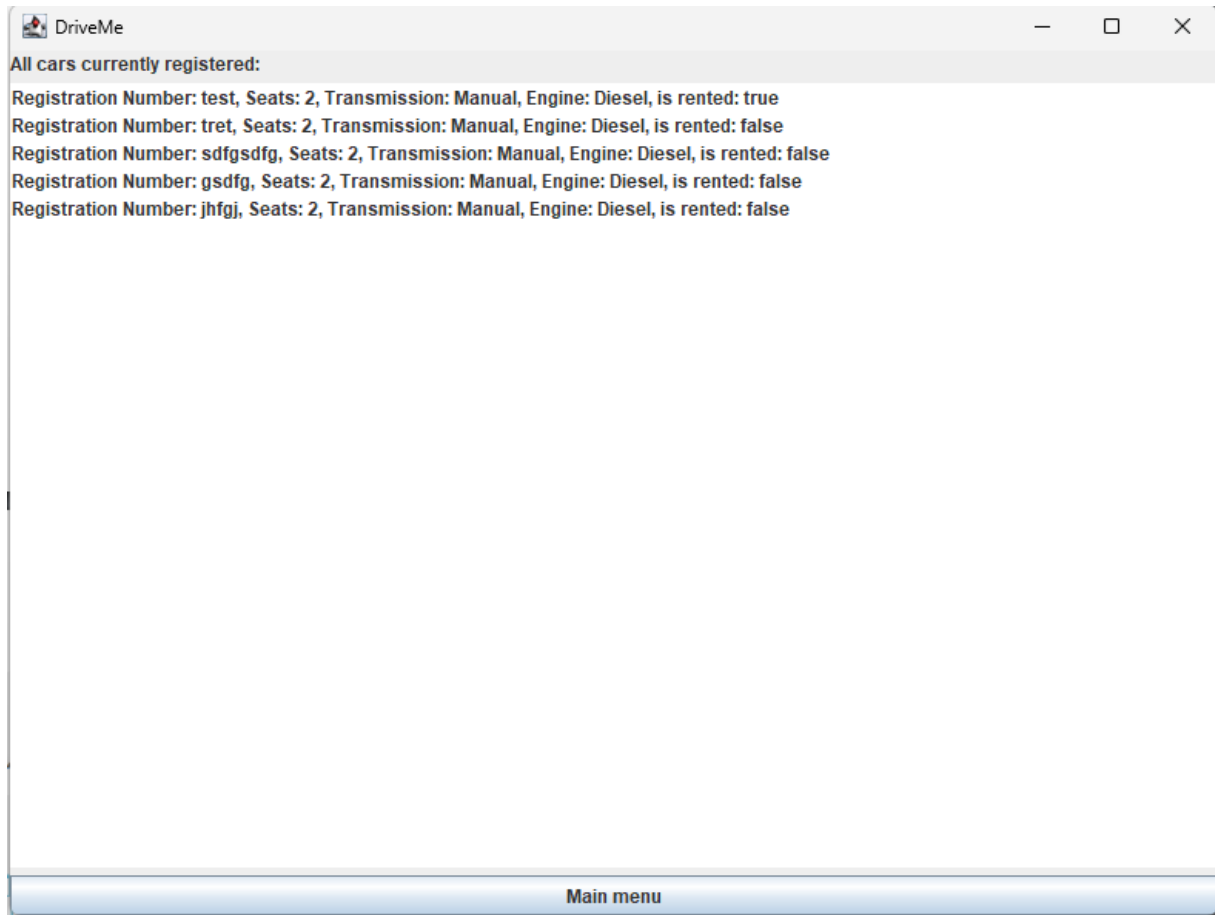
Når du først åpner prototypen så blir du velkommen av logg inn siden. Her har du muligheten til å logge inn som bruker (BankID), som gir deg muligheten til å registrer og låne bil, og admin, som gir deg muligheten til å redigere og slette biler fra listen.



Hvis du logger inn som bruker så får du muligheten til 4 funksjoner. Første er å logge ut (som tar deg tilbake til forrige siden), andre er å se alle biler som er registrert, tredje er å registrere en bil mens den siste er å kunne bestille en bil.



Vi starter først med «Browse all cars». Her så kan du se alle biler. Både de som er «rented» og de som ikke er «rented». Ved å trykke på main menu så går du alltid tilbake til innloggingssiden til brukeren.



I register car panelet så har man 5 forskjellige valg. Ut ifra de 5, så har man et begrenset antall valg på 3 av dem. Disse 3 inkluderer antall «seats» man kan velge, hva slags «transmission type» det er på bilen, og hva slags «engine type» det er på bilen. Disse har allerede blitt fast bestemt av oss på hva vi aksepterer. Vi har gjort det umulig å kunne registrere biler uten registrerings nummer, men utenom det så er alt mulig. Man kan også velge om bilen er klar til å bli lånt ut med en gang, eller om du allerede har lånt den ut.

DriveMe

Registration number

Seats: 2

Transmission type: Manual

Engine type: Diesel

☐ Rented ☐ Not rented

Register car

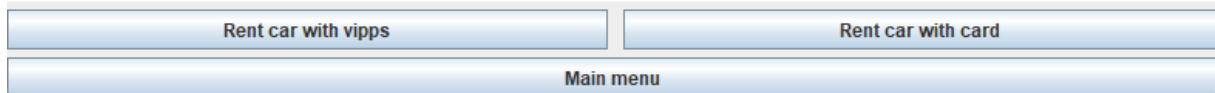
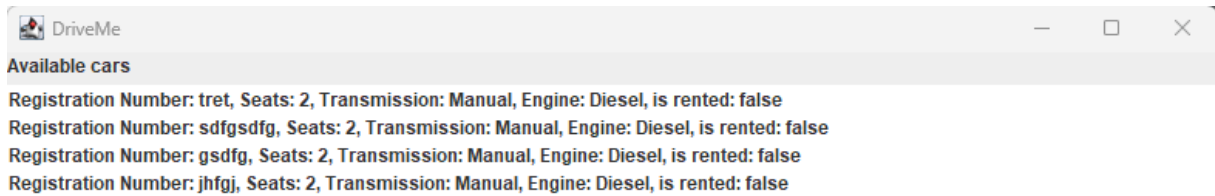
Main menu

Seats	2
Transmission type	2
	3
Engine type	4
	5
	6
	7

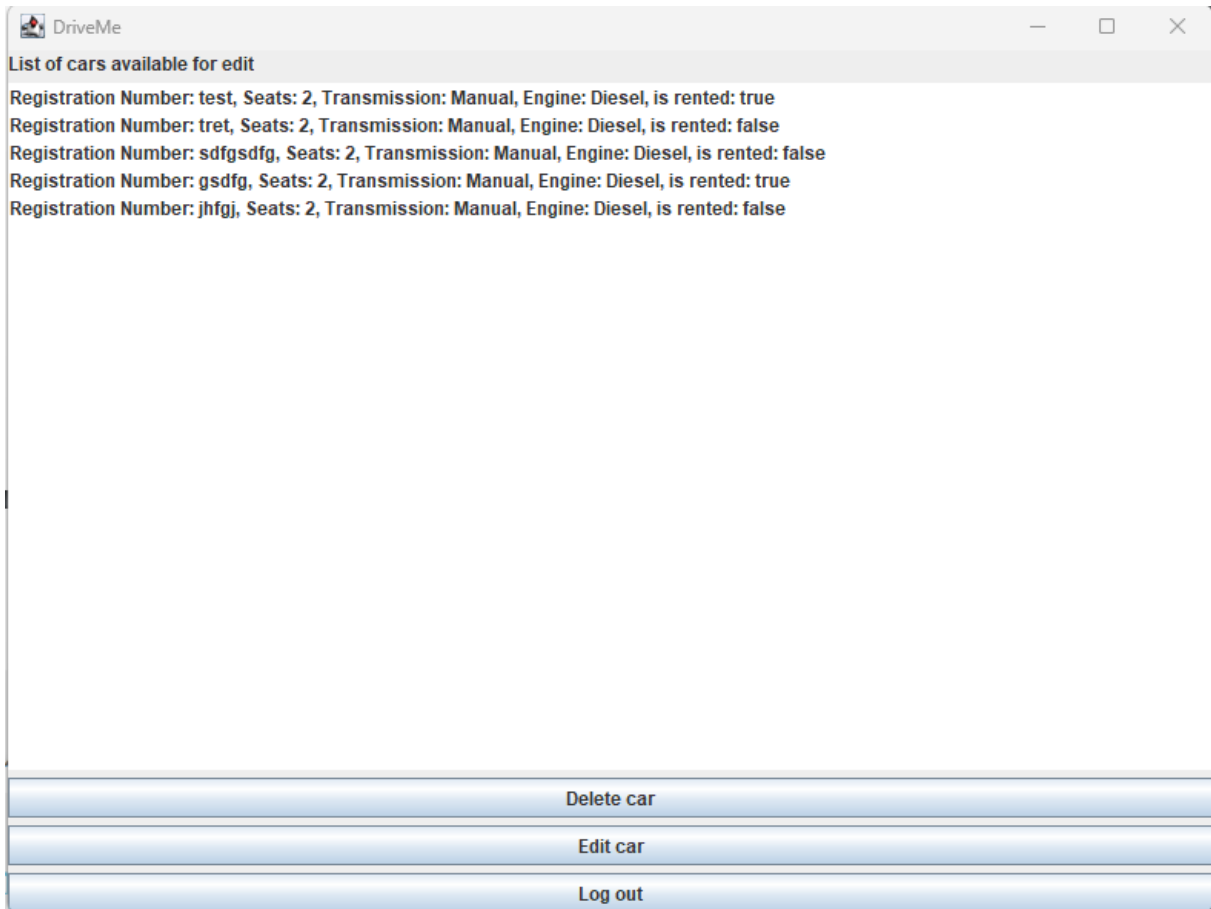
Transmission type	Manual
Engine type	Manual
	Auto

Engine type	Diesel
	Diesel
	Petrol
	Electric

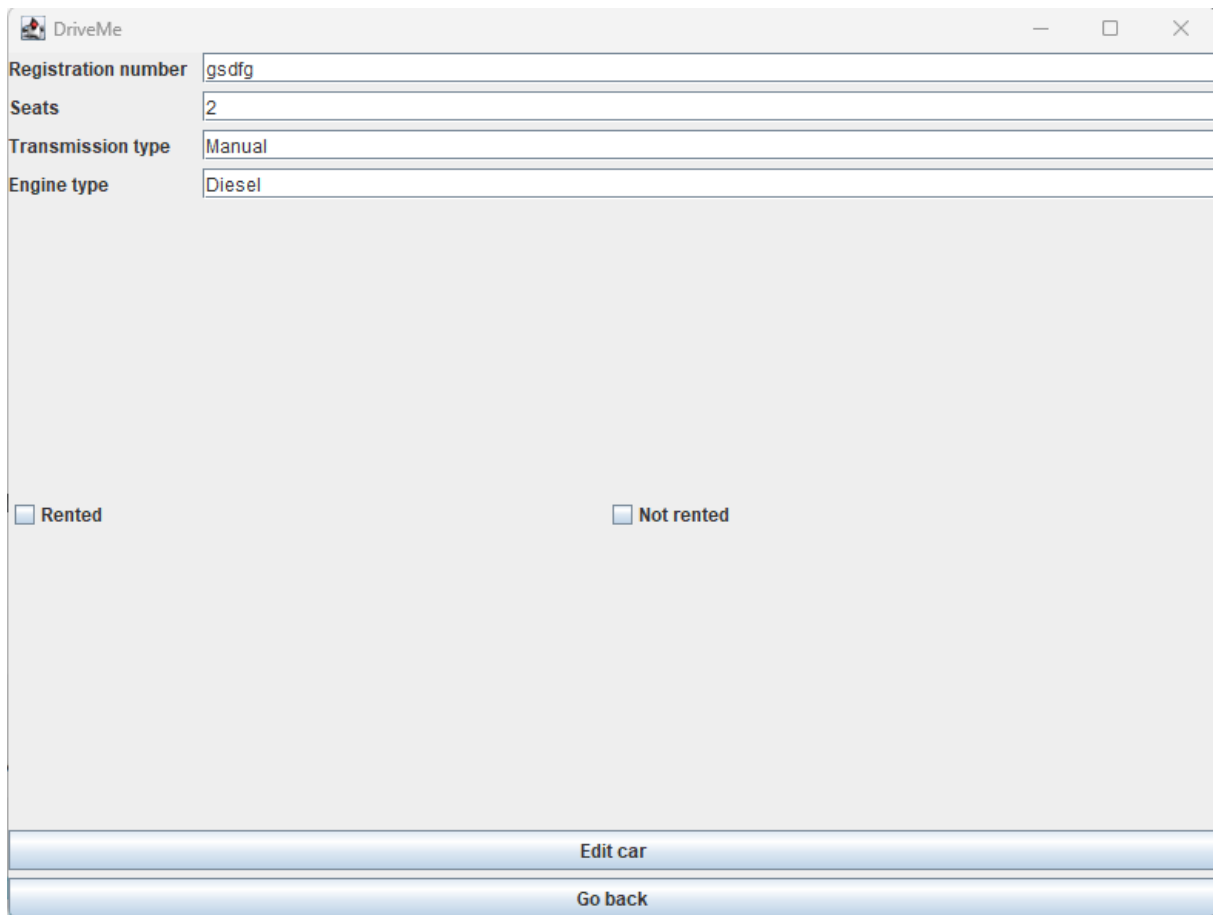
Til slutt så har vi «rent car» panelet. Her får du en liste med alle biler som er tilgjengelige. Du kan nå velge hvilken bil fra listen du vil låne bare ved å trykke på bilen, og deretter velge betalingsmåte.



Til slutt så har vi «admin» panelet. Når du først går inn i panelet så får du listen over alle biler i listen, både tilgjengelig og utilgjengelig. Her kan du lett trykke på en bil og slette den fra listen.



Du har også muligheten til å redigere biler ved å trykke på en bil og trykke «edit car». Her har du muligheten til å bytte alt informasjon slik som du ønsker ettersom at du er admin.



The screenshot shows a web application window titled "DriveMe". It contains a form for editing car details. The form has four input fields: "Registration number" with the value "gsdfg", "Seats" with the value "2", "Transmission type" with the value "Manual", and "Engine type" with the value "Diesel". Below these fields, there are two radio buttons: "Rented" (which is selected) and "Not rented". At the bottom of the form, there are two buttons: "Edit car" and "Go back".

Registration number	gsdfg
Seats	2
Transmission type	Manual
Engine type	Diesel

☒ Rented ☐ Not rented

[Edit car](#) [Go back](#)

## Testing ved bruk av Junit

*Testing kan bli utført ved å kjøre filene (CTRL+SHIFT+F10) Test\_CarRegister og Test\_CarRepository som begynner seg i src/test/java mappen.*

### Tittel:

Hvis du registrerer en bil uten å røre «rented» sjekkboksene så skal bilen automatisk være «not rented».

### Antagelser:

Brukeren glemte å trykke på «not rented».

### Test steg:

1. Bruker legger til bil uten å trykke på sjekkboksene.

### Forventet resultat:

Bilen blir automatisk lagret som not rented.

### Junit test:

```
@Test
public void creating_car_without_rented_parameter_is_not_rented_on_default() {
    CarRegistration notRentedCar = new CarRegistration("EG5345", "4",
"Manual", "Diesel");
    assertFalse(notRentedCar.isRented());
}
```



**Tittel:**

Når du låner en bil så skal det bli faktisk bli lånt ut.

**Antagelser:**

Brukeren låner bilen.

**Test steg:**

1. Bruker låner bilen.

**Forventet resultat:**

Bilen blir lånt ut og kommer nå ikke opp på «availableCarsList».

**Junit test:**

```
@Test
public void creating_car_and_renting_it_out(){
    CarRegistration rentedCar = new CarRegistration("PG46886", "2", "Auto",
"Petrol");
    rentedCar.setRented(true);
    assertTrue(rentedCar.isRented());
}
```

**Tittel:**

Når du registrer en bil som du vil legge til «repository» så blir den faktisk lagt til «repository»-en.

**Antagelser:**

Brukeren registrerte bil.

**Test steg:**

1. Bruker registrerer bil.

**Forventet resultat:**

Bilen kommer inn i listen for bil oppbevaringsstedet.

**Junit test:**

```
@Test
public void create_repository_with_cars() {
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Pet-
rol");
    testRepository.addCar(car1);
    assertTrue(testRepository.getCarArrayList().contains(car1));
}
```

**Tittel:**

Hvis admin fjerner bil fra «repository»-en så skal den bli fjernet.

**Antagelser:**

Admin vil fjerne en bil

**Test steg:**

1. Admin sletter bil fra «repository».

**Forventet resultat:**

Bilen blir slettet og befinner seg ikke lenger i «repository»

**Junit test:**

```
@Test
public void remove_car_from_repository() {
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Pet-rol");
    testRepository.addCar(car1);
    assertTrue(testRepository.getCarArrayList().contains(car1));
    testRepository.removeCar(car1);
    assertFalse(testRepository.getCarArrayList().contains(car1));
}
```

**Tittel:**

Sjekker om «repository» skrives til fil og leses opp. Sjekkes ved å se om «repository» er fortsatt samme klasse etter å ha blitt lest opp igjen

**Antagelser:**

Om fil skrivning og fil lesing fungerer riktig.

**Test steg:**

1. Blir gjort via backend.

**Forventet resultat:**

Listen blir skrevet ned og lest opp riktig, og sjekkes om listen er tom, og om listen er samme klasse som før den ble skrevet ned.

**Junit test:**

```

@Test
public void writing_and_reading_to_json_and_checking_if_still_same_class(){
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Pet-
rol");
    CarRegistration car2 = new CarRegistration("65GHDG3", "6", "Manual",
"Diesel");
    testRepository.addCar(car1);
    testRepository.addCar(car2);

    GsonBuilder gsonBuilder = new GsonBuilder().setPrettyPrinting();
    Gson gson = gsonBuilder.create();
    String json = gson.toJson(testRepository);
    testRepository.writeToJson(json);

    CarRepository testRepository2 = new CarRepository("test_reposi-
tory2.json");
    try {
        BufferedReader br = new BufferedReader(
            new FileReader("test_repository.json"));
        testRepository2 = gson.fromJson(br, CarRepository.class);
    } catch (FileNotFoundException e) {
        System.out.println("No save file to read from. Please make sure
you're trying to read an existing file.");
    }
    assertFalse(testRepository2.getCarArrayList().isEmpty());
    assertEquals(testRepository2.getCarArrayList().get(0).getClass(), CarRegis-
tration.class);
}

```

**Tittel:**

Bruker skal ikke kunne legge til 2 biler med samme registreringsnummer

**Antagelser:**

Bruker prøver å legge til 2 biler med samme registreringsnummer

**Test steg:**

1. Bruker lager 1 bil med registreringsnummer PG46886
2. Bruker prøver å registrere en bil til med registreringsnummer PG46886

**Forventet resultat:**

Bilen blir ikke lagt til og vi sjekker ved å se om bilen eksiterer i listen.

**Junit test:**

```
@Test
public void repository_does_not_accept_identical_registration_numbers() {
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Petrol");
    CarRegistration car2 = new CarRegistration("PG46886", "6", "Manual", "Diesel");
    testRepository.addCar(car1);
    testRepository.addCar(car2);
    assertFalse(testRepository.getCarArrayList().contains(car2));
}
```

### Tittel:

Bruker skal bare få opp tilgjengelige biler i listen når bruker skal låne bil

### Antagelser:

Bruker vil låne ledig bil

### Test steg:

1. Bruker åpner panelet for å låne bil og ser en ledig liste.

### Forventet resultat:

Alle bilene som kommer opp er bare ledige biler. Sjekkes ved å som opptatte biler kommer frem i «getAllAvaialableCars» listen.

### Junit test:

```
@Test
public void available_list_only_contains_available_cars() {
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Pet-
rol");
    CarRegistration car2 = new CarRegistration("65GHDG3", "6", "Manual",
"Diesel");
    car1.setRented(true);
    testRepository.addCar(car1);
    testRepository.addCar(car2);
    assertFalse(testRepository.getAllAvailableCars().contains(car1));
    assertTrue(testRepository.getAllAvailableCars().contains(car2));
}
```

**Tittel:**

Alle biler, både tilgjengelig og utilgjengelig, blir vist når det trengs.

**Antagelser:**

Bruker/admin vil se alle bilene som har blitt registrert

**Test steg:**

1. Bruker/admin åpner listen for alle biler

**Forventet resultat:**

Alle biler, både tilgjengelig og utilgjengelig, blir vist.

**Junit test:**

```
@Test
public void getCarArrayList_prints_all_cars() {
    CarRepository testRepository = new CarRepository("test_repository.json");
    CarRegistration car1 = new CarRegistration("PG46886", "2", "Auto", "Pet-
rol");
    CarRegistration car2 = new CarRegistration("65GHDG3", "6", "Manual",
"Diesel");
    CarRegistration car3 = new CarRegistration("6HFGFHF", "6", "Manual",
"Diesel");
    car1.setRented(true);
    car3.setRented(true);
    testRepository.addCar(car1);
    testRepository.addCar(car2);
    testRepository.addCar(car3);
    assertTrue(testRepository.getCarArrayList().contains(car1));
    assertTrue(testRepository.getCarArrayList().contains(car2));
    assertTrue(testRepository.getCarArrayList().contains(car3));
}
```