Karol Debik
Kenneth Langdahl Krogstad

Bacheloroppgave

Utvikling av en app som forenkler bestillingsprosessen for en drosje

Juni 2020

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Bacheloroppgave

2020

NTNU

Kunnskap for en bedre verden

Karol Debik
Kenneth Langdahl Krogstad

Utvikling av en app som forenkler bestillingsprosessen for en drosje

Bacheloroppgave
Juni 2020

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Forord

Denne bacheloroppgaven er skrevet i samarbeid med NTNU, som med et godt arbeidsmiljø og høyt kunnskapsnivå, ga oss et godt grunnlag. Vi vil derfor takke ansatte og medstudenter tilknyttet NTNU.

Vi har en interesse for å utvikle apper og ønsket å skaffe oss en bredere kunnskap innen det å utvikle et større system hvor en applikasjon er en del av systemet. Vi vil også takke MI OG MA HOLDNING AS for en spennende og utfordrende problemstilling, som har gitt oss muligheten til å lære mye om hele prosessen bak utformingen av et system. Vi håper derfor at systemet som ble utviklet kommer godt til nytte hos MI OG MA HOLDNING AS.

Å skrive denne oppgaven har vært spennende og utfordrende, men ikke minst lærerikt. Spennende fordi vi fikk muligheten til å jobbe med å utvikle et system, der vi fikk utvidet vår kunnskap innen apputvikling. Utfordrende på grunn av innsatsen oppgaven krevde, spesielt skriveprosessen. Med denne oppgaven har vi flere å takke for støtte og delt kunnskap under oppgavens livstid.

Vi vil gi en takk til vår veileder Majid Rouhani for å ha kommet med den veiledningen vi trengte for å holde stø kurs. En takk til Henrik Johnsen fra Teknisk stab på NTNU, som har brukt tid for å hjelpe oss med å sette opp servermiljøet slik at vi kunne anskaffe sertifikat og testmuligheter. Vi vil også takke Atle Årnes fra Datatilsynet som svarte utfyllende på alle spørsmålene våre knyttet til hva appen vår bør dekke i personvernområdet med hensyn på gjeldene GDPR lov.

Med den drastiske endringen i arbeidsmiljøet, på grunn av utbruddet av pandemien covid-19 som kom i løpet av oppgavens levetid, har vi måttet tilpasse oss så godt vi kunne. Vi ville ikke ha klart å få fullført denne oppgaven uten den støtten og tålmodigheten vi har hatt fått fra våre nærmeste familiemedlemmer. Det å jobbe på hjemmekontor har hatt bratt med seg flere utfordringer for oss begge. Hos Karol har kona og barna vært tålmodig og gitt god støtte for å bidra til et godt arbeidsmiljø. Kenneth har en kone og venter barn, noe som har gjort det vanskelig å bestandig prioritere oppgaven da graviditeten har vært utfordrende. Begge familiene har godtatt mye og vi takker for deres hjelp og støtte under skrivingen av bacheloroppgaven.

Signatur for oppgaven finnes i bacheloravtale (Se vedlegg D)

Oppgavetekst

Mi og Ma Holding AS er et selskap som utvikler og etablerer nye selskaper. De har fått med seg at det kommer nye regler innen drosjemarkedet og har kommet opp med en ide om et selskap de tenker å kalle Fast Track Taxi Systems AS. Systemet som er ønsket er en app som skal brukes for å bestille en drosje. Den samme appen skal brukes for drosjesjåfører for å finne disse bestillingene og utføre jobben. For kunden skal appen være superenkel å bruke: ved ett trykk skal bestillingen sendes. Denne bestillingen inkluderer nøyaktig lokasjon av enheten som blir brukt ved bestilling. Denne lokasjonen skal brukes slik at sjåføren som befinner seg nærmest kunden får et tilbud om å kjøre kunden først. Dersom det ikke er noen sjåfører i nærheten skal appen gi kunden en mulighet om å betale ekstra mot det å bli prioritert. Det skal også være begrensninger på hvilke sjåfører som får bruke appen. Denne begrensningen ble endret slik at sjåføren må igjennom en kvalitetssjekk på et kontor, hvor de så blir registrert i systemet. Appen skal ikke ha en betalingsløsning, men lagre fakturadetaljer for de turene sjåførene utfører, slik at et regnskapskontor kan generere faktura. For å få et bedre blikk på hva oppgaven var, kan man se på vedleggene visjonsdokument (Vedlegg 6.1), kravdokumentasjon (Vedlegg 6.2) og originale oppgave beskrivelsen (Vedlegg E).

Sammendrag

Denne bacheloroppgaven besvarer oppgaven gitt fra Institutt for datateknologi og informatikk, NTNU og Mi og Ma holdning AS. Hensikten med oppgaven var å utvikle en applikasjon som gjør det enklere å bestille en drosje. Med oppgaveteksten som bakgrunn, ble følgende problemstilling formulert:

> **Hvordan utvikle en app med React Native, som forenkler prosessen for bestilling av nærmeste ledige drosje?**

Denne problemstillingen ble valgt da det kom krav om at appen skulle lages både for Android og iOS. Forenklingen av bestillingsprosessen og at nærmeste drosje skulle være den som tok ordre var oppdragsiver konkret på. Resten var opp til studentene. I teorien tar vi for oss de endringene som skjer innen drosjemarkedet, gjøre rede for hvordan forretningsmodellen er i dagens løsning og hvorfor løsningen som blir laget er noe som er regjeringen setter opp til.

For å løse problemstillingen på best måte, brukte vi en agil utviklingsprosess basert på SCRUM. Dette innebærar regelmessige møter med veileder og oppdragsgiver, slik at utviklingen gikk i rett kurs. Applikasjonen er laget med fokus på at det skal være enkelt for brukerne. Applikasjonen ble utviklet med rammeverket «React Native», et rammeverk som viser seg å halvere arbeidsmengde og kostnader i direkte utvikling, læring og arbeidsmiljø.

Det fullstendige produktet leverer en enkel løsning for bestilling av drosje, som gir deg den nærmeste ledige sjåføren. Applikasjonen kan kjøres på både Android og iOS enheter, slik at den største markedsandelen av smartenheter blir nådd. Dette er en forbedring sammenlignet med forretningsmodellen med en drosjesentral og kan være løsningen på et mer konkurransedyktig drosjemarked.

Nøkkelord: React Native, Nodejs, applikasjon, Android, iOS, systemutvikling, drosjemarkedet, drosje

Summary

This bachelor thesis answers the task given from the Department of Computer Science, NTNU and MI og Ma Holdning AS. The purpose was to develop an app which makes ordering a taxi easier. Based on the received task, we formulated the following research question:

> **How to develop an app using React Native, which simplifies the process for ordering the nearest taxi.**

This research question that was chosen to meet the requirement that the app should be developed for Android and iOS. The requirements about a simplification of the ordering process and that the nearest taxi is the one to be contacted, was the employer clear about. The rest was up to the students. In the theory section of the paper the team will take on the changes coming within the taxi industry, look at the business model of today's system and the reason to why the government means the industry needs an improvement.

To reach the research question, the team chose an agile development process based on the SCRUM prosses. This entails meetings with the supervisor and the employer throughout the thesis, to keep the progress on track. The app is developed with the focus on simplicity of the use for the user. The app was developed using the «React Native» framework, a framework which reduces the time and cost needed for development, learning and creating a development environment.

The finished product delivers a simple solution for ordering a taxi, which gives you the nearest available driver. The app can be deployed to Android and iOS units, to react the biggest market share of smartphone users. The business model improves the effectivity compared to the business model with a taxi dispatcher. This solution is an attempt to make the taxi industry competitive, so that customers can get the service they deserve.

Keywords: React Native, Nodejs, application, Android, iOS, Software Engineering, taxi industry

Innholdsfortegnelse

Innhold

Figurliste

Akronymer og forkortelser

For å forstå innholdet i rapporten har vi laget en liste over ord og begreper som trenger forklaring og som setter begrepene i kontekst.

App og **applikasjon**, sammen med bøyningene, vil bety det samme.

Mobil, **smarttelefon**, **telefon** og **mobiltelefon** betyr i denne rapporten det samme.

FFT står for Fast Track Taxi og er navnet på appen vi har utviklet. Drosjekunder og drosjesjåfører er de som bruker appen.

GUI står for Graphical User Interface som på norsk betyr grafisk brukergrensesnitt. Grafisk brukergrensesnitt er det brukerne ser og samhandler med for å bruke appen.

IDE står for Integrated Development Environment, og er et program som brukes for å utvikle kildekode.

Lokasjon - med lokasjon mener vi plasseringen av en enhet ved å bruke enhetens latitude og longitude som sier hvor de er i verdenen.

GPS står for Global Positioning System er et satellittnavigasjonssystemsom som gir en nøyaktig tredimensjonal posisjon samt en svært nøyaktig tidsreferanse [1].

Løyver når vi skriver om løyve mener vi drosjeløyve, som er det man må ha for å ha lov til å ha en personbil som kan brukes til å frakte personer mot betaling.

GDPR står for General Data Protection Regulation, også kjent som personopplysningsloven. I 2018 kom det nye regler for hvordan personopplysninger skal behandles, det er disse vi refererer til når vi snakker om personopplysningsregler [2].

OTP står for One Time Password, altså et engangspassord.

MVP står for Minimum Viable Product, altså en prototype av det planlagte produktet.

Introduksjon og relevans

1.1 Bakgrunn

Sommeren 2020 kommer det nye regler for drosjenæringen. Det blir lavere krav (som løyver) som vil gi rom for flere selskaper. Dette vil medføre at et større antall drosjer og en større konkurranse om kundene. Dette vil sannsynligvis resultere i færre turer, lengre ventetid mellom turene og dermed lavere lønnsomhet for sjåførene til tross for at «drosjer i Norge kjørte i alt 282 millioner kilometer med passasjerer i fjor. Det er en økning på 2.9 prosent fra 2018. Antall kjørte kilometer minket fra 2014-2017 for igjen å stige i 2018-2019 til nesten samme nivå som i 2014.» [3] [4]. Med disse nye selskapene vil kvaliteten på service bli dårligere når det kommer til lokale geografikunnskaper samt god beherskelse av norsk. For noen er ikke pris viktigst. Disse mener at konkurranse på pris kommer til å gjøre det vanskeligere å finne sjåfører de føler seg komfortable med. De er som oftest i aldersgruppen 50 år pluss, og ønsker en mulighet for en enkel og trygg måte å bestille en drosje på. De ønsker en app hvor de trykker på en knapp, hvor de deretter blir oppringt av den nærmeste sjåføren, uten å måtte finne det selskapet man vet sjåførene er slike de ønsker. De ønsker også å ha mulighet til å betale mer for å bli prioritert hvis det mangler ledige drosjen i nærheten.

1.2 Oppgavetekst

Oppgaven var å utvikle en applikasjon for Android og iPhone som gir drosjekunder følgende 2 fordeler. Når det er ledige drosjer, får kundene den nærmeste sjåføren. Når det er kø for ledig drosje, kan kunden oppgradere bestillingen sin til en prioritert bestilling. Drosjesjåfører skal ha samme app men med annen funksjonalitet. Når en kunde ber om en tur, skal de 12 nærmeste drosjene i rekkefølge ut fra avstand til kunde få tilbud om å ta turen. Drosjesjåføren skal kunne ringe opp kunden da orden blir tatt, eller svare med SMS ut ifra hva kunden foretrekker. Appen skal ved hjelp av GPS holde orden på hvor kunder og sjåfører befinner seg og koble sammen de som er befinner seg nærmest hverandre. Det vil være noe sanntidsutfordringer, og køproblematikk ved at flere drosjer får samme tilbud, men den som trykker først, og dermed aksepterer bestillingen, får turen. Hentet ifra dokumentet som beskriver oppgaven som ble gitt til oss.

I løpet av oppgavens livstid har en del endringer oppstått. Muligheten for at kunden skal kunne velge at den vil bli kontaktet via SMS i stedet for å bli oppringt, er blitt borte. Det å velge de 12 nærmeste drosjene har blitt endret til å ikke begrense til de 12 nærmeste, men en tidsforsinkelse som bestemmer når sjåføren ser ordren. Denne tidsforsinkelsen er bestemt på avstand mellom sjåfør og ordre. Den nærmeste sjåføren får muligheten til å ta ordren før noen som befinner seg litt lengre unna. Sjåføren skal ha en liste over ordrene, ikke bare et enkelt varsel. Det kom også fram nye funksjonaliteter som oppdragsgiver ønsket for systemet. Disse består av: en web-basert løsning for administrator, en nettside med informasjon om bedriften og hvor potensielle sjåfører kan få informasjon om hvordan man kan bli

en Fast Track Taxi sjåfør, og funksjonalitet for at kunden skal ha mulighet til å vurdere sjåføren etter endt transport.

1.3 Problemstilling

I bacheloroppgaven vi ble tildelt ble vi introdusert for et marked vi selv ikke er godt kjent med, drosjemarkedet. Oppgaveteksten beskrev to problemer knyttet til drosjenæringen: At den eksisterende løsningen for bestilling av drosje er ikke enkel nok, samt at det kommer nye regler for drosjenæringen sommeren 2020, som vil gjøre det vanskeligere for kunder å finne sjåfører man føler man kan stole på. Nye regler mot drosjesentraler, skal komme, noe som gir muligheten til å lage et mer effektivt og lønnsomt alternativ for sjåførene. Vi har derfor kommet fram til problemstillingen

"Hvordan utvikle en app med React Native, som forenkler prosessen for bestilling av nærmeste ledige drosje?".

Kravet vi har fått fra oppgavestiller var å lage en app, hvor man kan bestille en drosje enkelt ved kun å trykke på en knapp. I utgangspunkt i denne problemstillingen ville vi utforske ulike selskaper for å se på hvilke eksisterende løsninger de har for bestilling, implementere løsningen vi er bedt om å lage og se om vi har forenklet prosessen.

1.4 Oppgavens struktur

Oppgaven starter med et teorikapittel hvor vi har hentet nødvendig informasjon for å kunne utvikle systemet vi skulle lage. Neste kapittel beskriver metoder og teknologier som er beskrevet i teorikapittelet. Deretter kommer resultater av arbeidet som ble gjort og diskusjon hvor disse resultatene drøftes opp mot målene. Avslutningsvis kommer en konklusjon på hvorvidt vi har klart å besvare problemstillingen og hvordan videre arbeid vil se ut. Det vil bli referert til vedlegg underveis i dokumentet.

Teori

Det første vi måtte gjøre var å skaffe informasjon for å få en oversikt over hvilke metoder og teknologi vi kan velge mellom for å angripe oppgaven. I dette kapittelet beskriver vi først en oversikt over hvilket marked vi befinner oss i, deretter hva vi var nødt til å tenke på for å utvikle et bestillingssystem. Utvikling av et bestillingssystem består av å sette seg inn i en del teori, for å kunne lage noe som vil fungere over lengre tid. Deretter beskriver vi hvilke utviklingsmetoder som finnes for å lage bestillingssystemet.

2.1 Drosjeløyve

Drosjenæringen er splittet inn i tre ulike komponenter: drosjesentral, løyvehaver og drosjesjåfør. En drosjesentral er den som mottar en bestilling fra en kunde om drosjetur og videreformidler bestillingen til sjåførene som er knyttet til denne sentralen. Løyvehaver er den som har et drosjeløyve, som er nødvendig for å kunne disponere en personbil som kan brukes for å transportere mennesker mot betaling. Drosjesjåfør er den personen som kjører bilen løyvehaver disponerer og utfører transporten av drosjeturene. I dagens system bestemmer fylkeskommunen hvor mange løyver som skal deles ut i de ulike løyvedistriktene. Disse blir bestemt ut ifra hvor stort behov det er for drosjer i området. Man trenger å fylle ett par krav før man kan anskaffe et løyve. Man må ha forretningsadresse i Norge, en vandelsattest fra politiet som bekrefter at du oppfyller vandelkravet, samt andre attester for økonomiske grunner. Man må også bestå en løyveeksamen før man kan søke og drosjeyrket må være løyvehavers hovederverv. Drosjen må være knyttet til en drosjesentral som gir tilgang til drosjekunder.

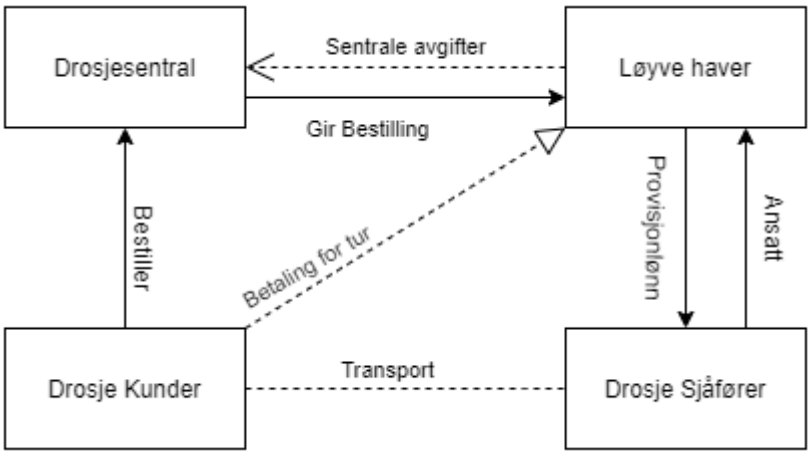
Det blir stadig gjort undersøkelser og utredelser for hvordan drosjemarkedet er i de aller fleste land rundt om i verdenen. Regjeringen har etter vurdering fastslått at drosjemarkedet ikke er konkurransedyktig. Markedet har problemer med en sunn konkurranse over kunder, som skaper dyre priser, dårlig service og derfor lite interesse i å bruke drosje som en transporttjeneste. [5]

I mars 2009 ble UberCab, en smarttelefonapp der man trykker på en knapp for å få skyss, laget i USA [6]. Etter stor suksess i USA lanserte de ideen internasjonalt. Da de kom til Norge møtte de store utfordringer. Yrkestransportloven som Norge hadde gjorde at Uber måtte avslutte sin virksomhet i Norge, samt betale store gebyr. Det ble gjort mange undersøkelser i etterkant av denne hendelsen. I en undersøkelse gjort av Norstat kom det fram at hele 75% ville vært positivt innstilt til en app som Uber. De fant også ut at 81% ville tatt drosje oftere dersom prisen hadde vært lavere, 1% dersom det var bedre tilgjengelighet, 3% dersom man kunne bestilt drosje ved bruk av en app og 5% dersom sjåførene var innstilt på en bedre service en dagens service. «Hovedbudskapet fra undersøkelsen er at nordmenn generelt er åpne for nye tjenester og større valgfrihet i markedet. Valg som kanskje tidligere har blitt begrenset av utilgjengelig prising, sier Norden-sjefen.» [7]

Regjeringen forstod at noe måtte gjøres. I juli 2020 kommer det nye løyveregler som skal forbedre drosjemarkedet. De håper Uber kommer tilbake, men forholder seg til eksisterende regler. «Vi ønsker folk ut i arbeidslivet, og at det ikke er størrelsen på lommeboken som bestemmer om man kan etablere seg i drosjemarkedet.». Kravet om bruk av drosjesentral fjernes, noe som vil si man kan finne drosjekunder drosjesentraler ikke hadde gitt enkelte sjåfører. Begrensing på antall løyver vil bli fjernet, noe som vil gi rom for tilpassing etter etterspørsel [5]. Løyvehaver trenger ikke å ha drosjeyrket som sitt hovederverv.

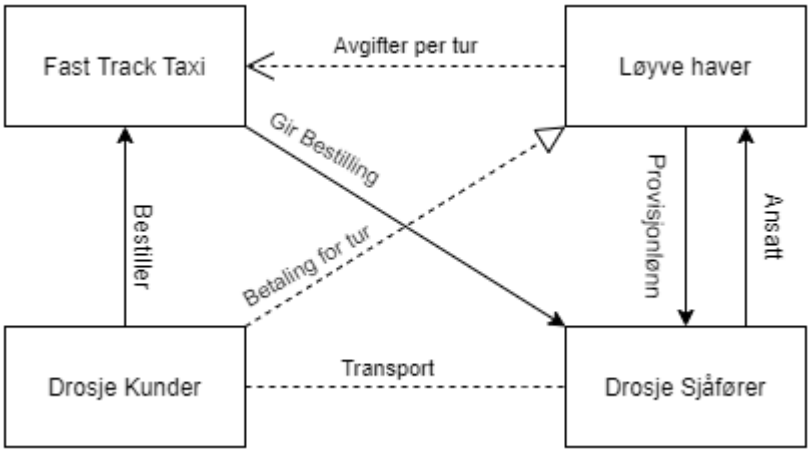
2.2 Forretningsmodellen

Her under ønsker vi å illustrere og forklare hvordan dagens drosjetjeneste ser ut, samt hvordan systemet vi utvikler endrer på dette og hvorfor denne endringen er bra for drosjemarkedet



Figur 1 Forretningsmodell for dagens drosje system

I dagens system går alle bestillinger, som ikke er gjort på gaten, gjennom en drosjesentral. Drosjesentralen gir bestilling til en av løyvehaverne som er tilknyttet de, og løyvehaver har en ansatt som utfører selve transporten av kunden. Etter at tjenesten er utført betaler kunden i bilen eller i en app til løyvehaver. Sjåføren får provisjonslønn fra løyvehaver. Løyvehaver kan ikke være tilknyttet mer enn en drosjesentral og må betale avgifter for å være tilknyttet dem. Dette betyr at for kunder som bestiller hos en drosjesentral utelukkes alle sjåfører som ikke er knyttet til denne sentralen.



Figur 2 Forretningsmodell Fast Track Taxi

Fast Track Taxi ønsker å endre på denne forretningsplanen, der den største forskjellen er at alle løyvehavere skal kunne benytte systemet, uten å måtte betale store avgifter. Systemet bruker lokasjonen til kunden og sjåføren for å gjøre det lett å finne transportmuligheten som vil bli utført raskest. I dette systemet skal en faktura bli sendt til løyvehaver, som viser antall turer som kom fra appen basert på en sum per tur, som må betales til eieren av appen.

2.3 Drosjeregulering

I Norge, samt andre økonomisk utviklede drosjenæringer, er reglene under stadig forbedring. Reglene som blir endret er hvilke krav som skal gjelde for sjåførene, bilene, adgangen til markedet og økonomisk regulering (prisregulering) når det kommer til tjenesten drosjen ufører [8].

Regjeringen mener det er for lite konkurranse i markedet og det påvirker tilbudet som er i dag. I juli 2020 kommer det nye regler som skal forbedre dagens tilbud. En av endringene er at løyvereglene endres betraktelig. Et drosjeløyve er tillatelsen for å transportere mennesker i en personbil mot betaling. Det er fylkeskommunen som bestemmer hvem som får disse løyvene. Det var begrensingerer på hvor mange som fikk løyver som gjorde det vanskelig å skape et sunt konkurransedyktig marked hvor kundene får den servicen de fortjener. Med de nye reglene fjernes begrensningen på hvor mange løyver som kan tildeles, noe som gir større mulighet for å tilpasse tilbudet etter etterspørselen. Drosjene trenger ikke lenger å være tilknyttet en drosjesentral, det blir færre regler for løyvehaver, men flere for sjåførene. En av de store endringene som blir gjort er at løyvehaver kan ha annet yrke eller arbeid i tillegg til drosjeyrket. Løyvehaver disponerer en bil som en drosjesjåfør kan bruke og med de endringene som blir kan personen ha annen jobb ved siden av drosjeyrket. På denne måten kan det komme flere måter drosjetjenesten kan bli tilbydd på, som kan skape et bedre og mer konkurransedyktig marked [9].

2.4 Personopplysninger

En personopplysning er alle former for data som kan direkte eller indirekte beskrive en enkeltperson. Typiske personopplysninger er navn, telefonnummer og e-post. I 2018 kom det nye regler, oftest kjent som GDPR, eller personopplysningsloven. Denne loven fastslo hva en personopplysning er, hvordan man skal behandle personopplysninger og hvilke rettigheter personer har for sine personopplysninger [10].

En personopplysning skal behandles på en lovlig, rettferdig og åpen måte. De skal hentes inn for spesifikke årsaker. Personen har mange rettigheter når det kommer til sine personopplysninger. Man har rett til å gi samtykke om innhenting av opplysninger, rett til å bestemme om disse kan deles og man har rett til innsyn i hvilke personopplysninger som er lagret. Før man blir en bruker av et system, må man gi samtykke til å samle personopplysningene som er beskrevet i brukervilkårene. Som bruker har man deretter rett til å endre sine opplysninger og rett til å bli informert om sine opplysninger. Når man avslutter sin tid som bruker for et system har man rett til å bli glemt, og med glemt mener vi slettet fra systemet. Dermed skal man helt fra start til slutt ha kontroll over sine opplysninger [11].

Brudd på personvernforordningen resulterer med sanksjoner. Den letteste formen for en sanksjon er en skriftlig advarsel, men den kan være i form av en bot på opptil 20 mill. euro eller mer hvis det er et foretak med veldig stor omsetning [12].

2.4.1 Personvern

Den nye GDPR loven kommer med store endringer i forhold til det siste personverndirektivet som ble innført i 1995. Det vil si at det ikke hadde blitt gjort noen store endringer i personvernloven i Norge på over 20 år. Det er viktig å nevne at GDPR loven gjelder både for offentlig og privat virksomhet.

Det som er endret er at loven gjelder også for selskaper som er registrert utenfor Europa, men som selger varer over internett. Det er ikke lov å bruke opplysninger til andre formål uten samtykke. Man har rett til å motsette seg mot profilering og profilert markedsføring. Selskaper får mer ansvar når det gjelder innsamling, registrering, lagring og behandling av disse dataene generelt. Selskaper er lovpålagt å ha innebygd personvern i sine applikasjoner. Bedrifter, som tidligere ikke hadde et personvernombud, har plikt til å opprette et slik ombud. Stort sett er arbeidet til et personvernombud uendret, men selskapene må sørge for at det blir et tilfredsstillende arbeidsmiljø for personvernombudene. En person har rett til å få innsyn i personopplysninger lagret om seg selv som kan bli levert i et lett forståelig språk. Den siste store endringen som den nye GDPR loven kom med, er at datatilsynsmyndighetene i alle land har en plikt til å samarbeide i saker som berører dem [13].

2.5 Teknologi

Hovedmålet med dette prosjektet var å lage et effektivt og enkelt system for bestilling av en drosje. For å kunne utvikle et slikt system, trengte vi kunnskap om hvilke teknologier som er tilgjengelig og hvilke metoder som bør brukes for utviklingen av systemet.

2.6 Smarttelefon

Smarttelefon er en avansert mobilenhet som kombinerer telefoni og datamaskin egenskaper. Disse enhetene er som oftest utstyrt med trådløst nettverk (både mobilt internett og WIFI), GPS-mottaker og Bluetooth. Smarttelefoner bruker apper (applikasjoner), som kan benytte seg av funksjonalitetene enheten tilbyr [14]. I dette prosjektet blir bruken av telefoni, internett og GPS de viktigste funksjonalitetene.

En applikasjon er ett program som kjører på smarttelefoner. De er utviklet for å utføre en spesifikk oppgave og kan installeres via en integrert app slik som Google Play Store eller Apples App Store. Applikasjoner kan deles inn i to hovedkategorier: native og hybrid apps.

2.6.1 Native app

En native app er den mest kjente typen av en applikasjon. Disse er skrevet for spesifikke plattformer, med egne programmeringsspråk. For iOS blir de skrevet i Swift og Object-C, mens de for Android blir skrevet med Java eller Kotlin. De bruker også sitt eget integrert utviklingsmiljø (IDE) for det utvalgte operativsystemet. Native app utvikling har mange fordeler som blant annet mye tilgjengelig dokumentasjon, er mer stabil ved endringer og er dermed bedre enn de andre løsningene [15]. Ulempen er derimot at man må bruke mye tid på å tilrettelegge miljøet når man programmerer i native. Dersom man ønsker å lage en applikasjon for både iOS og Android, trenger man to helt forskjellige miljø. Vanligvis vil et større selskap ha forskjellige avdelinger for apputvikling for de forskjellige plattformene. En programmerer velger sitt felt og går sjeldent over til den andre siden. Dette krever derfor en del mer kunnskap før man kan komme i gang med utviklingen, men gir til gjengjeld mulighet for å lage en mer pålitelig app.

2.6.2 Web app

Når man snakker om web app er det viktig å understreke forskjellen mellom en web app og en nettside. En nettside har mer informasjon enn hva en web app klarer å vise, den komprimerer derfor informasjon for å øke funksjonaliteten. En web app lastes via en nettleser, blir ikke installert på enheten og bruker derfor ikke opp lagring på enheten. En slik app kan føre til en del frustrasjon når det gjelder lastetid, små bilder og et ustabilt Network. En web app blir skrevet i språk som JavaScript, CSS og HTML5. Dette gjør utviklingen av en web app ganske kjapp og billig i forhold til native app, men det kommer på bekostning av en mer avansert funksjonalitet.

2.6.3 Hybrid app

En hybrid app fungerer på flere plattformer og oppfører seg som en native app. En hybrid er en kombinasjon av en native app og web app. Den blir lastet ned slik som en native app, men oppfører seg veldig lik en web app. Den skrives nesten likt som en web app, og bruker HTML, CSS og JavaScript. Her ligger appen som en native app i bakdelen, men laster ned og viser koden slik en web app ville gjort. De store fordelene med hybrid app er at man kan bruke enhetens interne API'er, men blir skrevet med ett språk. Dette er en veldig god løsning for mindre bedrifter som ikke har mulighet til å ha flere avdelinger for å nå alle plattformer. De to store innen hybrid er Xamarin og React Native. React Native har den styrken at man kan lage autentiske native apper, ved bruk av en kombinasjon av React og JavaScript. Disse har tatt hybrid utvikling til ett høyere nivå og mange mener de ikke tilfører hybrid klassen noe, men kun er veldig gode løsninger som får de til å se ut som en native app. En annen fordel med hybrid app er at selve appen kan henge seg opp, uten at enheten henger seg opp også. «In 2018, React Native had the 2nd highest number of contributors for any repository in GitHub." [16]

2.6.4 React Native

I Norge har vi to store konkurrenter når vi snakker om apper, Android og iOS. Appene i disse aktørene er skrevet på to forskjellige språk, iOS blir skrevet med Swift og Android skrives med en versjon av Java. Det å skrive swift på noen annen plattform enn iOS er praktisk talt umulig, hvis man ønsker det må man uansett teste og kjøre kompilering via en iOS plattform. I motsetning til iOS er android en open-source platform.

React Native er et rammeverk som tillater bruk av React biblioteket. React er et bibliotek for å lage brukergrensesnitt i JavaScript. React ble lansert i 2013 og siden det har en økende popularitet [17]. React Native bygger videre og kan bli brukt for å lage web løsninger og apper for Android og iOS. Noe som gjør rammeverket ganske populært i programmeringsmiljøet. På Github i 2018 var React Native nummer to i antall personer som bidrar til å utvide rammeverket. React Native lar deg bryte ned applikasjonen ned i deler, på denne måten kan man fokusere på enkelte komponenter uten å måtte ha hele applikasjonen i hode [18].

2.7 Lokasjon

Det er to forskjellige måter lokasjon blir brukt på i en smartenhet. Den ene bruker telefonens Cell Site location Information (CSLI), den andre bruker global positioning system (GPS). CSLI bruker det nærmeste telefontårnet for å finne en tilnærmet lokasjon. Man kan også bruke noe kalt triangulering, da brukes flere tårn for å få en mer nøyaktig lokasjon [19]. GPS derimot bruker satellitter for å finne lokasjonen, og denne metoden fungerer over hele planeten. Den er svært nøyaktig.

2.8 Database

Når vi snakker om lagring av data, blir som oftest databaser brukt. Databaser deles opp i to ulike typer, relasjonsdatabase (SQL) eller ikke-relasjonell database (NoSQL). De er begge gode til lagring av data, men har sine fordeler og ulemper. Det er derfor viktig å være bevist på hva man velger før man tar ett valg.

2.8.1 SQL database

En SQL-database bruker er et strukturert spørrespråk for å manipulere data. Det er en av de mest populære valgene vi har i dag for databaser og er derfor veldig godt dokumentert. Det et veldig godt språk for store spørringer, men den har sine svakheter. En av begrensningene er at den krever at all data holder samme form [20]. Hvis man ønsker en fast struktur, er SQL et godt valg. Data blir lagret i tabeller, og gjør det dermed lett å se relasjoner, noe som gjør SQL et godt alternativ for relasjonsdata.

2.8.2 NoSQL database

Ikke relasjoner spørrespråk (NoSql) har et forbestemt skjema, men man kan bruke det hvordan man selv vil. Man kan lagre data på mange forskjellige måter, som for eksempel i kolonner, dokument-orientert, grafbasert eller organisert med nøkler. I NoSql blir tabeller lagret som ett dokument og før man manipulerer data har den sin egen unike struktur. Man kan legge til felt uten å måtte tenke på andre dokumenter før man gjør endringer. I SQL måtte man ha utført flere endringer for å få tilsvarende endringer i tabellen [20].

2.9 Server

Server, tjener eller nettverksserver er ett og samme. Det er betingelsen for en datamaskin som er dedikert for å utføre tjenester for brukere innen et gitt nettverk [21]. Når man snakker om server mener man både maskinplattformen og programvareplattformen som datamaskinen benytter. Maskinplattformen er selve hardware for datamaskinen, mens programvareplattform er programvaren som kjøres på hardwaren. Det finnes mange programvareplattformer, men de siste 20 årene har det kommet fram en favoritt, Linux server. Hele 60% av alle servere ute på nettet er nettopp Linux servere, dette på grunn av dets mange fordeler. Linux er både gratis og har åpen kildekode. Åpen kildekode betyr at man kan bygge videre på koden for å tilpasse programvaren til sine behov. Det betyr også at det er mange som kommer med forslag på problemløsning hvis en feil eller et sikkerhetsbrudd blir oppdaget. Dette gir stor fleksibilitet og sikkerhet. Linux servere er også kjente for å være gode til å skalere. Samtidig er den kjent for å være litt vanskeligere å komme i gang med. Et annet alternativ er Windows servere, som er en av de største klient operativsystemene og er svært brukervennlige. Man kommer lett i gang, men ulempen er det er vanskeligere å tilpasse systemet og å skalere.

2.10 Sikkerhet

Når vi snakker om klienter og servere kan det være svært viktig med sikkerhet. Man ønsker at begge parter er hvem de sier de er, at det som formidles mellom disse ikke er tuklet med og at den ene ikke misbruker den andre. I denne delen skal vi skrive litt om de forskjellige måtene man kan bruke for å oppnå den nødvendige graden av sikkerhet innen IT verdenen.

2.10.1 Autentisering

Det å autentisere noe er å bevise at noe er ekte. Det er mange måter å oppnå dette på, men generelt sier vi at det er tre faktorer for å oppnå autentisering [22]. De tre faktorene er: Noe du vet, noe du har og noe du er. Noe du vet kan være et passord, men et passord har sine svakheter og derfor kan ofte være lurt å inkludere en faktor til. Dette kalles to-faktors autentisering. Da benyttes også noe man har sammen med noe man vet. Noe man har kan variere mye, det kan for eksempel være et passord eller en PIN fra en liste. Dette blir som oftest kalt en «Token». Den vanligste måten er å ha en enhet som generer dette. Mange banker brukte en egen kodebrikke som generer denne koden, men det har blitt stadig mer populært å bruke smarttelefoner som en generator. Mange mener at to-veis autentisering er tilstrekkelig, men i noen situasjoner kan det være ønskelig å øke til tre-veis. Tre-veis autentisering bruker også noe man er, noe man er blir oftest sett på biometri. Den mest kjente biometriske autentiseringen vi har er et fingertrykk, som er unik for hver person og er vanskelig å stjele.

2.10.2 SSL

Secure Sockets Layer (SSL) er en protokoll som gir muligheten for autentisert og krypter forbindelse mellom en klient og en tjener [23]. Det er mange grunner til at man ønsker en autentisert og kryptert kobling mellom to enheter, når man sender eller mottar informasjon. Man ønsker som oftest ikke at andre kan få innsyn i informasjonen eller at den er blitt tuklet med på en eller annen måte. Hvis man er tilkoblet et offentlig nettverk, som for eksempel på en restaurant, kan ruteren være satt opp slik at nettsider ikke går dit de skal, men blir videreført til en kopi av nettsiden. Gjelder dette en bankside kan konsekvensene være katastrofale, da informasjonen som blir oppgitt for å logge inn, blir videresendt til eieren av det trådløse nettverket. Man kan se forskjellen mellom en usikker nettside og en sikker en, ved å se på adressen til nettsiden. En usikker nettside har adresse som starter med «http://» mens den er «https://» for en sikker nettside. For å oppnå denne sikkerhetsstatusen trenger man et sertifikat. Det er mange måter å anskaffe et sertifikat på. Dersom man ønsker at sertifikatet skal stoles på, bør man skaffe det fra en rotleverandør også kjent som «Root Certificate Authority (CA)». Disse CA er selskaper som jobber for å bygge tillit og stiller seg som ansvarlig for å bekreftte at nettsiden med dette sertifikatet er autentisk.

2.10.3 SQL-Injection

Dersom man bruker SQL databaser er det lett for brukere å misbruke systemet dersom man ikke passer på. En standard spørring kan se slik ut “SELECT id FROM users WHERE username=’” + uname + “’ AND password=’” + passwd + “’”. Her tar vi inn et brukernavn og et passord for å logge inn en bruker.

Ettersom denne koden ikke har noen beskyttelse mot angrep kan en angriper gjøre alt han vil med kun få linjer. Det eneste man trenger å gjøre er å skrive «password' OR 1=1» og man blir logget inn uansett, selv om passordet er feil [24]. Detter er fordi det nye spørsmålet blir «hent bruker id hvor bruker er lik et brukernavn bruker og passord lik rett passord eller 1 lik 1 (som er sant) og man blir logget inn. Man kan deretter fortsette med flere SQL setninger, hvor man kan man gjøre hva man selv vil. Vil man slette tabellen eller endre alle passord til å være 123 er det ingen stopp for dette i denne spørringen. For å forhindre at dette skal være mulig kan man validere og sette begrensninger på det man tar imot, men man bør også bruke noe som heter «prepared statement». Prepared statement betyr at data som blir mottatt fra brukere blir behandlet slik at de ikke kan endre på spørringen som er planlagt. De kan alså ikke legge inn andre SQL setninger og kan derfor ikke misbruke systemet.

2.11 Utviklingsmetoder

Når man skal utvikle et system eller et produkt, bør man tidlig begynne å tenke på hvilken utviklingsmetode man tenker å bruke. Den tradisjonelle metoden å utvikle på er vannfallsmodellen. Det finnes mange varianter av vannfallsmodellen, men felles for de er at de vanligvis starter med en detaljert planleggingsfase, hvor produktet blir planlagt, designet og dokumentert i detalj [25]. Oppgaver som er nødvendige for å utvikle produktet blir planlagt, og blir lagt inn i ett gantt diagram eller en annen lignende metode for planlegging. Når denne planen er på plass, begynner utviklerne å utvikle produktet. Denne prosessen har sine styrker og svakheter. Alle oppgavene blir utført etter hverandre, men planlegging og utførelse er kjapt og kostnadseffektivt. Når man har startet prosessen jobber man til man er ferdig før man kommuniserer med kunden igjen. Da dette kan føre til feil, feiltolkinger eller nye ønsker som kommer underveis, passer denne prosessen best for korte og presise prosjekter [26]. Da denne prosessen ikke tar i betrakting at kunden ikke bestandig vet hva den vil ha, eller hva som er mulig, har iterative (agile) modeller blitt utviklet. I iterative modeller blir alt gjort stegvis og kan bli gjentatt flere ganger for å gå framover. Disse stegene blir satt i faste intervaller eller opp mot noe er ferdig, slik at de kan bli vurdert. Man kan da fange opp feil, rette feiltolkninger eller endre framtidige planer, slik at man ikke bruker mye tid på noe som ikke kommer til å bli brukt og dermed får de resultatene som er ønsket. [27]. Tilbakemeldinger er viktige i den agile modellen, man venter ikke til leveransedatoen før man måler resultatet.

Valg av teknologi og metode

I dette kapitlet vil vi beskrive hvilke teknologier og metoder som er blitt brukt i utviklingen av applikasjonen. Vi går først gjennom personvern, deretter tar vi for oss hvilke valg innen systemutvikling vi har tatt. Deretter beskriver vi hvilken organisering som er blitt brukt for at å forsikre at teamet oppnår gode resultater.

3.1 Personvern

Med tanke på personvern måtte vi sette opp systemet for å følge alle reglene som er skrevet i personopplysningsloven. «Personopplysninger skal behandles på en lovlig, rettferdig og åpen måte med hensyn til den registrerte («lovlighet, rettferdighet og åpenhet»), samles inn for spesifikke, uttrykkelig angitte og berettigede formål og ikke videre behandles på en måte som er uforenlig med disse formålene» [28]. For å følge disse reglene begynner vi når man starter applikasjonen. Da man startet appen kommer man til en side hvor man har muligheten til å lese om hvilke personopplysninger som blir lagret, hvordan disse blir behandlet samt hva som skjer dersom man ønsker å avslutte sitt kundeforhold i appen.

Først må vi finne ut hvilke opplysninger vi skal bruke. Personopplysninger vi trenger for kunden er telefon nummer, enhetens unike id og lokasjon. Telefon nummer og enhetens id blir lagret i databasen, lokasjon blir lagret for hver ordre som kunden bestiller. Dersom en kunde sier opp sin tid som kunde, må vi vente til faktura er sendt, deretter kan kunden fjernes fra systemet. Når det kommer til sjåføren er vi nødt til å ta inn ett par ekstra opplysninger siden bedriften sjåføren kommer ifra skal faktureres for bruken. Hos sjåføren må vi ta inn telefon nummer og enhetens id slik som kunden, men vi trenger også informasjon som kan brukes for å opprette faktura. Faktura detaljer som blir hentet inn er navn på sjåføren, taxi nummer bilen sjåføren bruker også bedriftens navn, organisasjonsnummer og faktura adresse. Det samme gjelder sjåføren dersom den ønsker å avslutte sin tid i systemet. Brukeren skal bli slettet da den siste faktura er generert og ferdig.

3.2 Systemutvikling

Det eneste oppgavestiller hadde stort ønske om var at applikasjonen skulle fungere på både Android og iOS. Ettersom vi ikke hadde nokk arbeidskraft, tid, kunnskap eller arbeids miljø til å klare å utvikle en applikasjon for begge plattformene, måtte vi finne en løsning som ga oss muligheten for å utvikle i ett IDE. Ettersom begge studentene ikke hadde mye erfaring med iOS, så tok det en tid før vi fant løsningen. Da vi fant ut om React Native fant vi fort ut at nodejs var løsningen for server vi kom til å bruke. Databasen systemet vi hadde tilgjengelig fra før hadde vi heller ikke noe problem med å koble sammen med serveren.

3.2.1 React Native og Expo

Det vi så for oss når vi søkte etter et rammeverk for utvikling av mobile applikasjoner var muligheten til å skrive en kode for både Android og iOS. Etter å ha gjennomgått mulighetene, ble React Native valgt fordi det var det mest populære rammeverket som egnet seg godt til prosjektet vårt. Vi tenkte også at god dokumentasjon var tilgjengelig og at det skulle være lett å finne svar ved eventuelle spørsmål, på grunn av den store popularitet. Ved en tilfeldighet fant vi et videokurs på edX platformen som hjalp oss med å lære oss de grunnleggende byggesteinene ved utvikling av en app for mobile enheter ved bruk av React Native [29]. Rammeverket var helt ukjent for oss, men vi valgte det fordi et av kravene var å utvikle en app for både Android og iOS platformene. Etter at NTNU hadde stengt lokalene sine pga. covid-19 bestemte vi oss for å bruke Expo rammeverket [30] fordi det tillot oss og kjøre kode på kun en PC og samtidig starte applikasjonen på flere mobile enheter. Dette var mulig ved å lage en konto på Expo platformen og logge seg på med denne kontoen på hver mobil.

3.2.2 React Navigation

React Navigation er et bibliotek som ble brukt til å kunne navigere mellom komponentene i appen lett. Det var en foreslått løsning på sidene til React Native dokumentasjon [31].

3.2.3 Redux

Det var behov for å lagre en del informasjon lokalt på mobilen. Biblioteket som egnet seg godt til det var Redux biblioteket som vi fant ved hjelp av dokumentasjonen for React Navigation [32].

3.2.4 WebStorm

For å utvikle appen for både Android og iOS ved bruk av React Native gikk vi for WebStorm IDE. Valget vårt var begrunnet med det at vi hadde en studentkonto opprettet hos JetBrains fra før og vi var vant til IDEene levert av dem [33].

3.2.5 Server med Nodejs og mysql

Siden React Native bruker JavaScirpt når man utvikler er det lurt å bruke ett ramme verk for server som også bruker javaScript. Nodejs og React Native bruker samme pakke manager kalt npm (Node Package Manager). Med npm kan man installere og holde kontroll på alle API som både nodejs og React Native

bruker. Man kan derfor programmere både front-end og back-end i samme miljø. Det er derfor store fordeler å bruke Nodejs sammen med React Native.

Nodejs kan brukes på både Windows og Linux, er enkelt å sette opp og veldig lett å bruke. Vi brukte Windows da vi utviklet serveren lokalt, men møtte på problemer med å få serveren ute på internett. Etter hjelp fra IT avdeling på NTNU fikk tildelt en virtuell Windows server 2019. Med denne kom vi oss til slutt ute på internett med nodejs serveren. Men da vi ønsket å gå fra å kjøre server fra port 80 (http) til port 443(https) møtte vi problemer. Ettersom vi ønsket å benytte ett gratis alternativ for anskaffelse av ett sertifikat, kom vi fram til at vi skulle bruke Let's Encrypt [34]. Vi klarte å anskaffe sertifikat for selve windows serveren, med ikke web serveren fra nodejs som skulle kjøres på denne. Etter en stund i å prøve og feile kom vi fram til at dokumentasjonen ikke var tilstrekkelig og måtte gi oss. Vi fikk så tildelt en ubuntu 18.01, vi møtte så problemer med brannmurer. Da It avdelingen hjalp oss til et annet miljø prøvde vi på anskaffelse av sertifikat igjen. Dette viste seg å være enkelt og godt dokumentert for linux distribusjoner.

3.2.6 Gitlab

For prosjektet vårt har vi valgt Gitlab som lagringsplass for utviklingsprosessen fordi den har et versjonskontrollsystem og fordi vi har studentkontoer tilknyttet NTNU. Det er også innebygd kontinuerlig integrasjon som blir kjørt hver gang noen laster opp endringene sine til en branch i repoen vår og på den måten hjalp oss med å holde orden på en fungerende kode.

3.2.7 Jest

Vi valgte jest som testrammeverket fordi i dokumentasjonen for React Native var testeksemplene skrevet ved hjelp av Jest og vi så for oss at dette rammeverket egner seg best for prosjektet vårt [35].

3.2.8 JSDoc

For å dokumentere klassene og funksjonaliteten til metodene i kodene brukte vi JSDoc fordi den er tilgjengelig i Webstorm som en plugin [36].

3.2.9 Prettier

For å ha en kode som blir bestandig formattert på samme måte brukte vi brukte Prettier som vi kunne tilpasse til behovene våre og etter det vi tenkte at koden skal se bra ut. I tillegg var det mulig i IDEen vår å ha den slått på hele tiden [37].

3.3 Sikkerhet

Når man lager et system er det mye man må tenke på, sikkerhet er en av de store. Systemet må beskytte seg mot de som benytter systemet, men systemet må også beskytte brukeren. Systemet må hindre andre fra å ta kundens informasjon, forfalske en front slik at kunden tror de er vårt system, men er på rett system de snakker med. Siden applikasjonen gir ikke noe valg til å velge hvilket domene som blir brukt er det ikke mulig for kunden å rote seg til feil system, men dersom man ikke bruker https er tilkoblingen mellom enheten og severen ukryptert. Dette er noe iOS ikke aksepterer, noe som betyr at https er ett krav for vårt system. Man kan lage egne sertifikat og signere de selv, men da er det ingen andre programmer eller enheter. Som nevnt i kapitlet over bruker vi Let's Encrypt for å anskaffe sertifikat som vil bli godkjent av de fleste enheter. Da vi fikk på plass sertifikat begynte vi å tenke på å hvordan vi skal beskytte oss mot brukere som ønsker å misbruke systemet. Serveren er en web server med endepunkter som kan nåes på mange måter, vi er derfor nødt til å begrense hvem som kan bruke de endepunktene til serveren. Vi endte opp med å nedprioritere sikkerhet når det gjelder kontroll på om kunden er hvem de sier de er, samt token som er på ett godkjent nivå. Server slik den er nå bruker en felles hemmelighet hos serveren og applikasjonen. Da denne stemmer får brukeren en godkjent token, som gir muligheten til å bruke funksjonaliteten som finnes. I applikasjonen er det veldig få plasser vi tar inn noe kunden putter inn, men de plassene som finnes blir beskyttet med begrensinger på hvilken type data og hvor stor denne dataen er.

3.4 Testing

For serveren brukte vi applikasjonen som testingmediet. For databasen blir det laget enkle sql setninger for å teste, disse blir eksekvert i databasens webløsning.

For å teste applikasjonen valgte vi Jest som testrammeverket.

For mer detaljert og oppdatert informasjon om testene kan man lese systemdokumentasjonen (Se vedlegg C).

3.5 Organisering

I dette underkapitlet skriver vi om utviklings metode, arbeids- og rollefordeling, samhandlingsverktøy og kommunikasjon. Hvordan teamet er satt sammen, hvordan teamet jobber sammen og hvilken utviklings metode de brukte for å få de resultatene de gjorde.

3.5.1 Oversikt

For å holde god oversikt over arbeidet som trengs å gjøre har teamet bestem seg for å bruke Trello, en digital versjon av den tradisjonelle fysiske tavlen. Teamet er vant til digitale løsninger og gjør Trello til rette valg. I Trello har man det lett for å opprette ett gantt diagram som teamet kan bruke for å holde

kontroll om de er i rute eller ikke. Alle arbeidsoppgaver blir delt opp i mindre deler slik at man lett kan se hva som treng å gjøres og ikke. Hver arbeidsoppgave kan deles ut til enkelte medlemmer.

3.5.2 Utviklings metode

Teamet benyttet en agil lik utviklings metode. De jobbet i intervaller på omtrent 2 uker. Hvor vært intervall hadde møter med veileder og oppgavestiller. De begynte med å få på plass ett visjons dokument (Se vedlegg A) og laget ett gantt diagram (Se vedlegg D). Da dette var på plass begynte de å lage ett kravdokument (Se vedlegg B), dette dokumentet ble framvist en gang og godkjent av oppdragsgiver. Da vi skal ha møter omtrent hver andre uke. Dersom andre krav eller funksjoner ble oppdaget ble dette dokumentet ble oppdatert underveis. Da systemet var ferdig ble det laget ett systemdokument (Se vedlegg C)

3.5.3 Arbeids- og rollefordeling

Teamet består av to utviklere, som har nesten lik kompetanse nivå. Karol litt mer erfaring med front-end (applikasjon) utvikling, mens Kenneth har mer kompetanse på back-end (server) enn det han har i front-end. Derfor har Karol hoved ansvar for applikasjonen og Kenneth har hoved ansvaret for serverne. Det er ingen leder, men begge ble enige om hvilke områder de har ansvar for slik at man vet at ting blir gjort uten å måtte spørre den andre. Karol var møteleder for møtene, Kenneth referent. Karol hadde ansvar for innsending av dokumenter og innkallelse til møter, mens Kenneth hadde ansvar for generering av timeliste/status rapport. Da det oppstår uenigheter får den som har ansvar ta valget, med mindre det er stor uenighet, da blir det tatt opp på møte med veileder eller oppgavestiller.

3.5.4 Samhandlingsverktøy

Det er flere verktøy som er nødvendig for å ha en lett og oppdatert utvikling. Alle dokumenter blir produsert og lagres blir gjort i Teams, Teams er ett filhåndterings system fra Microsoft. Siden Teams er fra Microsoft har de gode løsninger for samhandling i å skrive med programmer som Words og Excel. Med disse har alle tilgang til de oppdaterte filene, og jobbe sammen samtidig uten store problemer. For selve programmeringen blir koden lagret med git. Git-repository'et som blir benyttet er gitLab, da denne er lett å bruke, sikker og oversiktlig.

3.5.5 Kommunikasjon

Kommunikasjon mellom medlemmene skjer med systemet Slack, da dette er en oversiktlig og enkel måte å sende meldinger på. Teamet begynte med å jobbe sammen på NTNU sine lokaler, men da jobb på hjemmekontor ble innført på grunn av cov-19. Med hjemmekontor begynte teamet og ta videomøter via Zoom. Meldinger og fil deling med veileder skjer via e-post, møter skjedde omtrent hver andre uke. Disse møtene ble gjort med Zoom da hjemmekontor ble implementert. Kommunikasjon med oppgavestiller også gjort med e-post og møter hver andre uke, videomøter etter hjemmekontor.

Resultater

I dette kapittelet blir først vitenskapelige resultater beskrevet, deretter ingeniørfaglige resultater og til slutt skal de administrative resultatene. Disse resultatene skal brukes i neste kapittel, diskusjon, hvor vi skal diskutere resultatene opp mot problemstillingen.

4.1 Vitenskapelige resultater

Systemet som er blitt utviklet i denne bacheloroppgaven består av en komponent i front-end og to komponenter i back-end. Front-end komponenten består av en applikasjon som fungerer likt på Android og iOS smarttelefoner, mens back-end består av en web server som kommuniserer med en database. For at front-end skal ha funksjonaliteten som er ønsket, er back-end nødt til å fungere også. Disse kommuniserer med hverandre for å oppnå det ønskede resultatet. Systemdokumentet (Se vedlegg C) beskriver hvordan systemet er bygget opp og hvilke funksjonaliteter som eksisterer.

4.1.1 Applikasjon

Applikasjonen er hvor vi får frem resultatet og hvor vi best måler serveren. Applikasjonen bruker serveren for å utføre forskjellige oppgaver. Det er her kunden og sjåføren blir koblet sammen slik at sjåføren får utført tjenesten som er ønsket av drosjekunder. Først skal vi beskrive hva systemets MVP er og hvordan applikasjonen ser ut. Det er viktig å huske at det er fysiske forskjeller mellom Android og iPhone. Alle Android enheter har en fysisk bakk knapp, mens iPhone ikke har den.

MVP

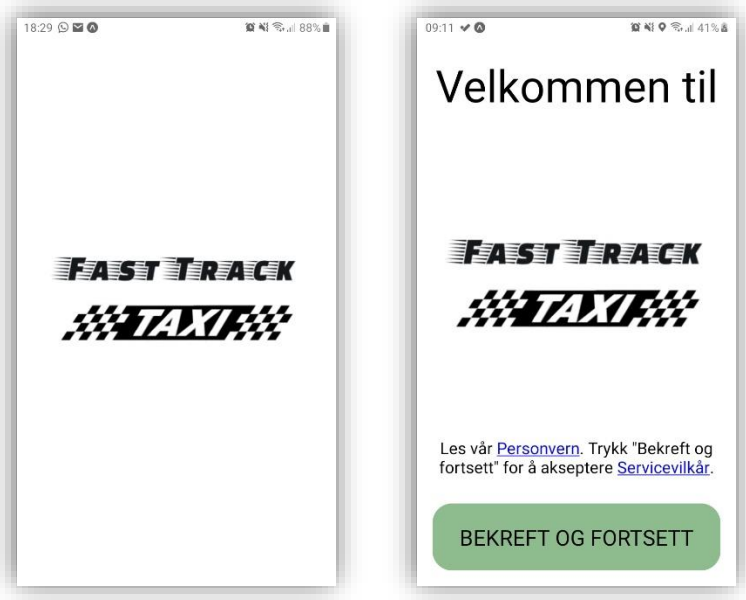
MVP er en betegnelse på det minste produktet man må ha for å få testet sin ide eller teori. I et MVP er mye av applikasjonen laget, men en oppkobling opp mot serveren mangler. Etter at teamet oppdaget mangel på tid mot hva teamet hadde planlagt å utvikle måtte de revurdere hva som skulle implementeres. Teamet satte seg ned og kom fram til nå en MVP for systemet. Det teamet trengte for å komme med resultater var: En kunde bestiller en taxi, en drosjesjåfør ser denne bestillingen og kan kontakte kunden slik at de kan møtes for transporten. Andre funksjonaliteter ble nedprioritert og implementert dersom det ikke var store problemer for å oppnå et bedre resultat.

Flyten i applikasjonen

Resultatene vises best med å vise flyten i applikasjonen. Vi skal vise hvordan kunden bestiller, og hva sjåføren må gjøre for å kunne ta denne ordren.

Begge brukernes flyt i applikasjonen

De aller fleste skjermene er like for Android og iPhone, vi bruker derfor Android for å vise der de er like og viser noen eksempler på hvordan det ser ut for iPhone der de er forskjellige. Den naturlige flyten for kunden er: laste ned applikasjonen, akseptere personvern/servicevilkår, registrere sitt telefonnummer, skrive inn et engangspassord (OTP), bestille taxi (hovedskjerm), se at appen leter etter sjåfør, få bekreftesle på at sjåføren er på vei, vurdere den utførte tjenesten og gå tilbake til hovedskjermen slik at den kan bestille en ny taxi. Sjåføren har samme start som kunden, men etter å ha skrevet inn OTP går den til hovedskjermen for sjåføren isteden. På sjåførens hovedskjerm har sjåføren denne flyten: sette seg som tilgjengelig og få vist liste over tilgjengelige ordre, trykke på en ordre, få telefonnummerert til kunden, avbestille turen og gå tilbake til hovedskjermen for sjåføren. Da man har fullført registreringen vil ikke denne komme neste gang man starter appen.



Figur 3 Bruker laster inn applikasjonen

Figur 4 Bruker har mulighet til å lese "personvern", "Service vilkår" og bekrefte disse

Når man trykker på app-ikonet på enheten, er det første som kommer opp en «loading screen» (Figur 3). Mens denne skjermen vises hentes variabler inn fra enheten fra tidligere kjøringer av appen. Den skulle ha kjørt en «fetch request» til serveren for å logge inn brukeren eller starte registreringsprosessen, men vi endte opp med å nedprioritere registreringsprosessen. Da «loading screen'en» er ferdig kommer vi

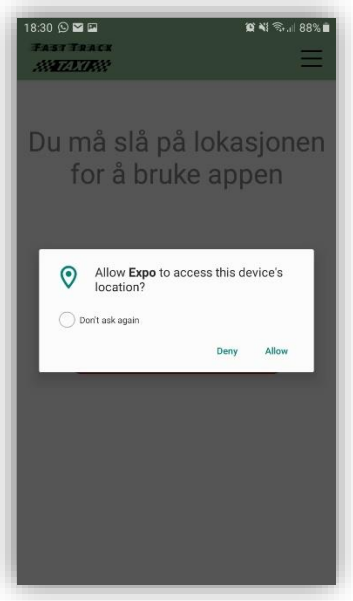
automatisk til siden hvor man kan lese om hvilke personopplysninger som blir brukt og servicevilkår, slik at man kan akseptere disse og gå videre (Figur 4). Neste skjerm er hvor man skriver inn sitt telefonnummer (Figur 5)

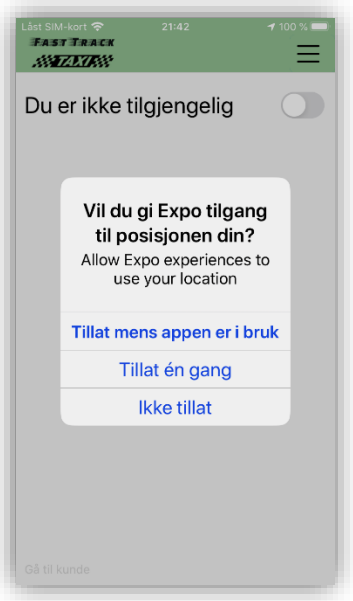
Figur 5 Starten på registreringsprosessen

Figur 6 Kunden trykket "Neste" i registreringsprosessen: bekreftet å ha skrevet inn rett nummer

Figur 7 Bruker venter på OTP

Når brukeren har akseptert person/servicevilkårene kommer den til siden hvor man kan skrive inn telefonnummeret sitt, for å få tilsendt OTP. Da man trykker på «Neste» etter å ha skrevet inn telefonnummeret, blir man spurt om nummeret man skrev inn stemmer (Figur 6). Dersom det stemmer trykker man ok, hvis ikke, trykker man «Endre» og går tilbake for å skrive inn nummeret på nytt. Da man har trykket på «ok» kommer man videre til neste skjerm. På denne skjermen venter brukeren på et OTP fra serveren (Figur 7). Da denne funksjonaliteten ble ikke implementert, skriver man inn seks siffer og trykker bekreft for å gå videre. I neste skjerm blir man spurt om å godta at appen bruker enhetens lokasjon (**Error! Reference source not found.** og **Error! Reference source not found.**).





Figur 8 Applikasjonen spør om tillatelse til å bruke lokasjon, Android

Figur 9 Applikasjonen spør om tillatelse til å bruke lokasjon, iPhone

Figur 10 Bruker trykket "Ikke tillat"

Figur 11 Tillatelsen ikke gitt, kun Android

Dersom brukeren (sjåfør eller kunde) ikke har gitt appen tillatelse til å bruke lokasjon tidligere, ber den om tillatelse for å bruke denne. Denne skjermen dukker opp bare på Android (Figur 11). På iPhone må man gå til innstillinger for å gi tillatelse. Lokasjon trengs for at en kunde skal kunne bestille en drosje da sjåføren bruker sin lokasjon for å måle avstanden mellom seg og kunden. Applikasjonen sorterer deretter aktive ordre, slik at de med minst avstand vises øverst. Da denne tillatelsen for å bruke lokasjonen blir godtatt blir brukeren sendt videre til hovedskjermen.

Kundens flyt i applikasjonen

Figur 12 Kundens startskjerm, Android/IPhone

Figur 13 Kunden trykket enhetens tilbake funksjon uten å bestille taxi, Android

Den første skjermen vi ser her (Figur 12) er hovedskjermen til kunden. Dette er hva kunden skal se først ved åpning av applikasjonen, etter at den har fullført registreringen. Hovedfunksjonen for denne siden er å bestille taxi. Øverst ser vi ikonet til appen samt en menyknapp. Menyknappen fungerer dessverre ikke. Vi støttet på en del problemer da vi ønsket å implementere menyknappen og siden dette tok mer tid enn det vi hadde planlagt måtte dette nedprioriteres for å ha tid til å implementere de viktigste delene. Midt på skjermen får vi opplyst adressen man befinner seg på, dersom kunden ikke vet hvor den er kan denne være nyttig for å kommunisere hvor man befinner seg. Man kan se en litt usynlig tekst, «Gå til sjåfør», denne knappen er tilstede slik at vi kunne gå til sjåførens hovedskjerm (Figur 25). Denne er implementert da vi ikke fikk tid til å implementere en sjekk på om brukeren er en kunde eller en sjåfør og skal ikke eksistere da appen har implementert en måte å logge bruker inn på rett profil. Deretter ser vi en stor grønn knapp hvor det står «Bestill taxi» i stor skrift og «(30kr)» i en mindre størrelse med mindre synlighet. Med teksten ønsker vi å tydeliggjøre funksjonaliteten denne knappen gjør. Dersom man trykker knappen bestiller man en drosje og at det medfører en tilleggspris. Man kommer deretter til neste skjerm (Figur 14). Dersom kunden trykker på enhetens tilbake knapp, blir man spurt om man ønsker å lukke appen eller fortsette å bruke den (Figur 13).

Figur 14 Kunde har trykket på knappen "bestill taxi", Android/iPhone

Figur 15 Kunde har trykket på «avbestill taxi», Android

Figur 16 Kunde har trykket på «avbestill taxi», iPhone

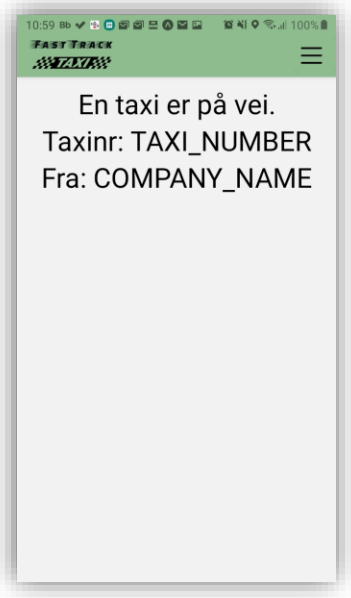
Dette er skjermen kunden ser dersom den trykker på knappen «bestill taxi» (Figur 14). Når kunden ser denne skjermen får den opplyst at applikasjonen «Leter etter en ledig taxi», i bakgrunnen sender appen en etterspørsel til serveren for å finne ut om det er en sjåfør som har tatt ordren. Da en sjåfør tar orden skal kunden bli oppringt og skal avtale hentested. Deretter skal appen navigere kunden til neste skjerm (Figur 20), hvor kunden blir informert om taxi nummer og hvilket firma sjåføren kommer ifra. Før en sjåfør har tatt på seg denne ordren har kunden muligheten til å angre seg. Dersom man ønsker å avbestille taxien trykker man på «avbestill taxi» (Figur 15 og Figur 16). Denne vil først spørre om man mente å trykke på knappen. Da man trykker på ja blir en melding sendt til serveren, serveren behandler denne meldingen og oppdaterer databasen slik at en sjåføren ikke lenger har muligheten til å ringe kunden. Kunden blir også navigert tilbake til kundens hovedskjerm. Dersom man trykker på nei blir varslet borte og den fortsetter å søke om det er en ledig sjåfør. Dersom det tar lang tid før en sjåfør tar orden får kunden muligheten til å oppgradere til en prioritert bestilling (Figur 17). Denne funksjonalitet ble ikke implementert, men man kan nå denne siden ved å trykke på «leter etter en ledig taxi».

Figur 17 Kunden får muligheten til å oppgradere til en prioritert bestilling

Figur 18 Kunde trykket på "kjøp deg ut av køen"

Figur 19 Kunden trykket avbestill taxi, prioritert ordre

Funksjonaliteten om å bestille en prioritert ordre ble ikke implementert da teamet måtte prioritere andre funksjonaliteter. Meningen med disse skjermene er å gi kunden mulighet til å oppgradere til prioritert bestilling (Figur 17) hvis det er lang ventetid. Dersom man trykker på knappen «Kjøp deg ut av køen», blir man navigert til skjermen som sier at appen leter etter prioritert taxi (Figur 18). Kunden har fortsatt mulighet til å avbestille taxien. Da en sjåfør tar ordren kommer vi til siden hvor kunden får opplyst taxi nummer og firma navn for sjåføren (Figur 20).



Figur 20 Kunden har en taxi på vei

Figur 21 Kunden trykket på enhetens tilbake funksjon

Når kunden har vært i kontakt med sjåføren kommer vi til siden hvor man blir informert om sjåførens taxi nummer og firma navn. Da man er på denne siden har man ikke mulighet til å gå tilbake (Figur 21). Kunden er nødt til å vente til enten sjåføren fullfører turen i appen eller at det har gått lang tid. Da skal appen gi kunden mulighet til å vurdere turen (Figur 22) og komme tilbake til hovedskjermen sin. Denne funksjonaliteten er ikke implementert og kunden blir sendt til å vurdere sjåføren etter kort tid, slik at vi kunne gå videre.

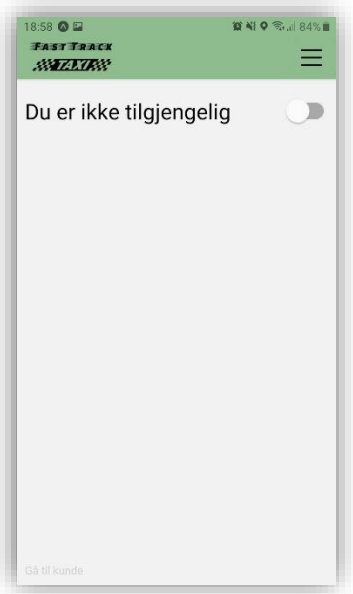
Figur 22 Kunde har sjåfør, mulighet til å vurdere sjåføren

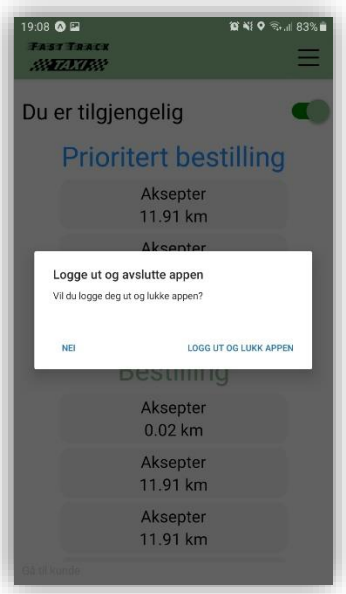
Figur 23 Kunden vurderer tur

Figur 24 Kunde trykket telefonens tilbake funksjon

Når kunden får muligheten til å vurdere turen har den valget om å gi en 1-5 stjerner eller å la være å vurdere turen (Figur 22). Ingen av funksjonalitetene på denne siden snakker med serveren, men lar deg gå tilbake til kundens hovedskjerm (Figur 12) og utføre samme prosess dersom man ønsker det. Dersom man trykker på en stjerne blir man spurt om antall stjerner stemmer med det man mente (Figur 23). Da man trykker «Gi vurdering» blir man sendt til hovedskjermen, hvis man trykker endre vurdering kan man velge en annen verdi. Man kan også trykke på «Nei, takk» for å navigere til hovedskjermen. Dersom man bruker enhetens tilbake funksjon (Figur 24) får man muligheten til å gå til hovedskjermen eller lukke meldingen slik at man kan vurdere turen.

Sjåførens flyt i applikasjonen





Figur 25 Sjåførens hovedskjerm, er ikke tilgjengelig

Figur 26 Sjåføren er tilgjengelig, ser ordre

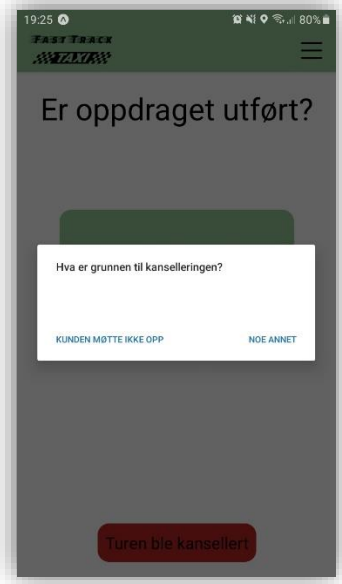
Figur 27 Sjåføren trykket enhetens tilbake funksjonen før ordre er tatt

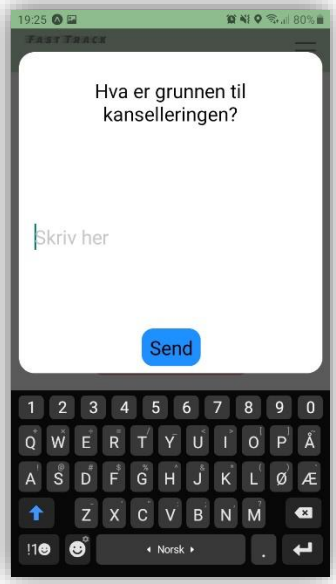
Da sjåføren er registrert kommer han til sin hovedskjerm (Figur 25). Som standard har sjåføren status som ikke tilgjengelig, og kan skru denne på til tilgjengeling. Da sender appen en melding til serveren om å få ledige ordene. Ordrene blir splittet opp i to lister, prioritert bestilling og vanlig bestilling (Figur 26). I hver liste blir ordrene lagt inn sortert etter avstand mellom sjåføren og ordrene. Hver ordre kan bli trykket på. Når man bekrefter en ordre sendes en melding til serveren som prøver å ta ordren. Hvis ordren fortsatt er ledig blir sjåføren lagt inn i databasen og får tilbake telefonnummeret til kunden på denne ordren (Figur 28). Hvis sjåføren trykker på enhetens tilbake funksjon får den et varsel (Figur 27). Sjåføren får da muligheten til å lukke appen eller å gå tilbake til ordrene.

Figur 28 Sjåføren tok en ordre, ringer kunden

Figur 29 Sjåfør har vært i kontakt med kunde, oppdrag utføre/ tur ble kansellert

Når sjåføren har trykket på en ordre får den telefonnummeret dersom ordren ikke allerede er tatt av en annen sjåfør (Figur 28). Sjåføren skal deretter se en skjerm som gir mulighet for å gi beskjed om at oppdraget er blitt utført eller kansellert (Figur 29). Denne siden kommer vi ikke til, ettersom ingen av funksjonene på denne siden snakker med serveren. Dersom man trykket på «Oppdrag utført», skal man sende melding til serveren om at oppdraget er fullført og serveren arkiverer den i databasen. Man blir også navigert til sjåførens hovedskjerm (Figur 25). Vi rakk ikke å implementere at den går tilbake til hovedskjermen sin med «Du er tilgjengelig» statusen og funksjonaliteten. Hvis man trykker på «Turen ble kansellert» får man opp en melding hvor man må skrive inn grunnen for kanselleringen (Figur 30).





Figur 30 Sjåføren trykket på «Turen ble kansellert»

Figur 31 Sjåføren gir grunn til kansellering, beskrivende

Figur 32 Sjåføren trykker enhetens tilbake funksjon etter å ha tatt ordre

Dersom sjåføren har vært i kontakt med kunden og har trykket på «Turen ble kansellert» må den begrunne hvorfor kanselleringen skjer (Figur 31). Den vanligste grunnen er at kunden ikke møtte opp da sjåføren kom for å hente kunden, derfor er det et valg. Dersom det er en annen grunn trykker man på «Noe annet», og man blir bedt om å skrive inn en grunn til at kanselleringen skjer (Figur 31). Når man trykker «send» kommer sjåføren tilbake til sin hovedskjerm (Figur 25). Dersom sjåføren trykket på enhetens tilbake funksjon da den er i en ordre, får man opp en melding (Figur 32Figur 30 Sjåføren trykket på «Turen ble kansellert»

Figur 31 Sjåføren gir grunn til kansellering, beskrivende

Figur 32). Her har man muligheten til å lukke meldingen hvis man trykker nei. Dersom man trykker på «Utført» kommer man til sjåførens hovedskjerm.

4.2 Ingeniørfaglige resultater

Det første teamet gjorde da de kom i gang med bacheloroppgaven var å beskrive målet for systemet i et visjonsdokument (se vedlegg A). Da en av medlemmene har sluttet og det ble trangt av tid ble mange funksjoner nedprioritert. Derfor er kun det som er blitt implementert og det er resultater på med her. For å vise i hvor stor grad målene er oppnådd brukes en skala fra 1-6, hvor 6 er oppnådd og 1 er ikke oppnådd.

4.2.1 Mål

1 Registrere seg

Da man åpner appen for første gang møter man en velkomstside, hvor personvern og servicevilkår ligger. Man må akseptere disse før man kan gå videre. Deretter kommer man til en side hvor man må skrive inn sitt mobilnummer og trykke «send engangskode». Da kommer man til neste side, hvor man skal skrive inn koden man får tilsendt i en melding. Da dette blir godkjent er man registrert og kommer til kundesiden dersom man ikke er registrert som sjåfør via et fysisk kontor.

Oppnådd grad: 3

2 Godkjenne lokasjons tillatelser

For å kunne bruke appen må man akseptere at appen benytter telefonens lokasjon. Det dukker opp et spørsmål opp om å gi tillatelse første gang kunden registrerer seg. For en sjåfør kommer dette spørsmålet opp når den vil gjøre seg «tilgjengelig» for første gang. Dersom man gir tillatelse, blir det husket i appens hukommelse.

Oppnådd grad: 6

3 Enkelt å bestille en drosje.

Dersom man har fullført registreringen som en bruker så kommer man i skjermen kunden skal komme til ved oppstart av appen. Hvis man ikke har akseptert at lokasjon blir brukt må denne aksepteres før man kan fortsette. Når dette er gjort møter man en skjerm med en stor knapp der det står «Bestill drosje» trykker man den, så legges orden inn i systemet og man blir navigert til en side som forteller deg at det letes etter en sjåfør.

Oppnådd grad: 6

4 Lett å forstå at en sjåfør er på vei

Når man har bestilt en drosje og sjåføren har oppringt kunden, vil skjermen i appen etter kort tid endres til informasjon om hvilket drosjenummer bilen har og hvilket firma sjåføren kommer ifra.

Oppnådd grad: 5

5 Vurdere sjåfør etter tur

Da sjåføren avslutter ordren etter en viss tid dukker det opp et vurderingsskjema i appen hvor man trykker på antall stjerner (1-5) og trykker vurder. Med mindre man ikke ønsker å vurdere så trykker man «ønsker ikke å vurdere sjåføren».

Oppnådd grad: 3

6 Se ordre for kunder som ønsker drosje

Når man er registrert som sjåfør og har skrudd «er tilgjengelig» på, kommer alle ordrene i sortert liste ut ifra avstand mellom seg og kundene.

Oppnådd grad: 6

7 Se lett forskjellen mellom en vanlig ordre og en prioritert ordre

Når man har fått ordrene så ligger prioriterte ordre i en egen liste som vises øverst, mens de vanlige ordrene ligger i den nederste listen.

Oppnådd grad: 6

8 Lett å ta en ordre

Da det finnes ordre ser sjåføren den i listen, man trykker på selve ordre man ønsker å ta. Ordre blir oppdatert og sjåføren får tilbake mobilnummer som blir åpnet i telefonens ringe app.

Oppnådd grad: 5

4.3 Administrative resultater

Her vil vi beskrive hvordan de administrative resultatene. Først blir timeregnskap beskrevet, deeretter kommer utviklingsprosessen. Med disse kan man se hvordan teamet organiserte seg.

Vedlegg som blir brukt under dette kapittelet: Timelister og statusrapport, Møteinnkallinger og referat, Gantt diagram (Vedlegg F)

4.3.1 Timeregnskap

Teamet lagde timelister som ble sendt inn til veileder på en ukentlig basis. Timelistene har også status rapport og beskrivelse på hva som ble gjort og hvem som gjorde det.

4.3.2 Utviklingsprosess

Utviklings prosessen teamet brukte var SCRUM basert. Prosjektet inneholdt 8 sprint. Under kommer en liten oppsummering av hver sprint. Oppsummeringen inneholder mål, avhengigheter og forutsetninger, til slutt resultater på slutten av sprinten. Timelistene inneholder beskrivelse som gi mer informasjon om hva som ble utørt i hver sprint.

Sprint 1

Sprint 1 ble gjennomført i perioden 05.01.2020 - 26.01.2020 (Uke 2-4).

Mål

Hovedmålet for sprint 1 var å skaffe en fullkommen forståelse av oppgaven. Det skulle utarbeides ett visjonsdokument, slik at teamet var enige om hva oppgaven var. Teamet skulle organisere seg slik at alle viste hvilke roller de har, samt hvilke regler som gjelder. Hvilke verktøy de skal bruke for samhandling og oversikt av prosjektet skal beskrives slik at det ikke kommer uenigheter om hva som skal brukes.

Avhengigheter og forutsetninger

For å komme bra i gang vil det bli utført ett møte med alle studentene, oppgavegiver og veileder. Slik at de involverte fikk en felles forståelse av domenet. Dette blir en viktig start hvor teamet spesifiserer hvor ofte møter skal skje og andre tekniske spesifikasjoner om hvordan organiseringen skal utføres.

Resultat

Under sprint 1 ble visjonsdokumentet laget, strukturen og organisering kom på plass. Det oppstod problemer mellom teamet. Kevin hadde kun mulighet til å møtes på ettermiddagen, da han hadde noen andre arbeidsoppgaver på starten av året. Etter de to første ukene bestemte han seg å forlate gruppen, da han følte han ikke var en del av teamet. Det ble startet på noen prototyper og «user-stories» for å anskaffe en sterkere forståelse av hva produktet var.

Sprint 2

Sprint 2 ble gjennomført i perioden 27.01.2020 - 09.02.2020 (Uke 5-6).

Mål

I sprint 2 skulle visjonsdokument revideres, første utkast av kravdokument skal produseres. Ett gantt diagram blir laget og sprints blir spesifisert. Kravspesifikasjoner skal utarbeides og fram stilles på i starten av neste sprint.

Avhengigheter og forutsetninger

Møte med veileder skal utført slik at visjonsdokumentet kunne revideres, samt møte med oppdragsgiver slik at kravspesifikasjoner kan justeres. På slutten av sprinten skal første utgave av kravdokument være klar for å blir framvist i starten på neste sprint.

Resultat

I sprint 2 ble visjonsdokument revidert etter møte med veileder, slik at versjon 1.0 ble ferdig. Prototyper blir revidert etter møte med oppdragsgiver, slik at kravdokumentet kan godkjennes på starten av neste sprint. Ganntt diagram blir opprettet og sprints blir spesifisert.

Sprint 3

Sprint 3 ble gjennomført i perioden 10.02.2020 - 03.03.2020 (Uke 7- 10).

Mål

I sprint 3 fremstilles av kravdokument og forstudie skal utført slik at teamet kunne starte utvikling i neste sprint. Møte med veileder skal utføres slik at struktur på kravdokument er god, møte med oppdragsgiver slik at alle krav var på plass og versjon 1.0 av kravdokument ble ferdigstilt.

Avhengigheter og forutsetninger

For å starte utviklingen i neste sprint, må alle kravspesifikasjoner være godkjente og organisering av hvordan utviklingen skal skje må planlegges. Møte med veileder skal utføres slik at kravdokument er strukturert rett og møte med oppdragsgiver slik at ingen misforståelser kommer under utviklingen.

Resultat

I starten av sprintet blir det tatt en pause, litt forstudie blir gjort. Deretter blir det jobbet med kravdokument. Det blir utført møte med veileder slik at kravdokument er strukturert korrekt. Det blir også utført ett møte med oppdragsgiver. Etter møtene blir kravdokument revidert og utvikling kan starte i neste sprint. På slutten av sprinten begynnes det også med hjemmekontor på grunn av NTNU sine retningslinjer for håndtering av COV-19 viruset.

Sprint 4

Sprint 4 ble gjennomført i perioden 04.03.2020 - 29.03.2020 (Uke 10-13).

Mål

Oppstart av hovedrapport og utviklings perioden starter. Det å komme i gang med hovedrapporten er viktig slik at rapporten ikke er forskjellig fra hva som blir utviklet. Finne ett miljø hvor serveren skal kjøres er noe som må komme på plass. Få struktur på hvordan hjemmekontor skal utføres blir ett viktig mål.

Avhengigheter og forutsetninger

Med hjemmekontor blir det utfordringer på hvordan kommunikasjonen mellom studentene blir, og hvor effektiv utviklingen blir framover. Det utføres ett møte med veileder mot slutten av spriten for å holde teamet på rett kurs.

Resultat

Utviklingen starter, hovedrapport blir satt opp slik at den kan skrives på gjennom prosjektets livstid. I starten på utviklings perioden blir det utført forstudie, slik at teamet fikk kunnskap om hvordan utvikling i react native fungerer. System dokumentasjon blir også startet på, slik at det blir mindre arbeid senere. En av studentene endte opp med å bli syk i nesten to uker, noe som stoppet utviklingen av applikasjonen en stund. Node.js server på lokal pc blir laget, og ett miljø der den kan kontaktes fra internett er nødvendig. Det ble kontaktet NTNU sin IT-avdeling slik at teamet kunne fått en server som kunne brukes.

Sprint 5

Sprint 5 ble gjennomført i perioden 30.03.2020 – 13.04.2020 (Uke 14-15).

Mål

Utvikling fortsetter, fokus på å anskaffe sertifikat for server. Hovedrapport skal holdes oppdatert,

Avhengigheter og forutsetninger

Det skal utføres møter med veileder og oppdragsgiver for å holde teamet på rett kurs. Anskaffelse av en server som kan nåes over internett blir viktig. Server ute på internett er nødvendig for testing av prototype og anskaffelse av sertifikat for å oppnå høyere sikkerhet. Med server ute på internett vil sikkerhet også bli ett viktig punkt å tenke på.

Resultat

Problemer med å holde fokus i hjemmekontor begynner å bli tydelig. GUI av applikasjonen begynner å ta form. Serveren blir oppdatert med beskyttelse mot sql-injection. Teamet har problemer med å anskaffe sertifikat for server de har fått tildelt. Token blir implementert for serveren slik at serveren er sikker da den blir lagt ute på internett. Møter med oppdragsgiver og veileder blir holdt for å holde teamet på rett kurs når det gjelder og oppdatering på hvordan helsen er på studenten som ble syk.

Sprint 6

Sprint 6 ble gjennomført i perioden 14.03.2020 – 10.05.2020 (Uke 16-19).

Mål

Hovedmålet i sprint 6 blir revisjon av gantt diagram. Redefinering av sprints blir viktig, da det må justeres etter uforutsatte endringer. Sertifikat må komme på plass. Kapitel 2 i hovedrapporten må nærme seg ferdig. Slik at den kan senders inn for tilbakemelding fra veileder.

Avhengigheter og forutsetninger

En revurdering av hva teamet måtte utvikle for å komme med resultater som kan måles opp mot problemstillingen for rapporten. Innsending av kapitel 2 til veileder blir viktig, slik at veileder kom komme med tilbakemeldinger.

Resultat

Det ble jobbet mye med å fullføre kapittel 2 på hovedrapporten, jobbet for å få sertifikat på serveren vi fikk tildelt fra NTNU. Det tok tid og var ikke før slutten på sprinten før sertifikat var på plass. Sever var da klar for å stå gående slik at appen kan testes ved alle endringer. Det ble definert hva som var nødvendig for å oppnå de resultatene som var nødvendig. GUI for applikasjonen er ferdig, og jest tester blir laget. Jest testingen er ikke så omfattende som vi har sett for oss, men vi ville teste noe av appen for å lære oss å bruke Jest som testrammeverket og for å se om appen genererer skjermer på en riktig måte. CI/CD blir satt opp på gitlab. Revisjon av gantt diagram og sprint for de siste to sprintene blir revidert for å komme i mål for den nye fristen som ble gitt.

Sprint 7

Sprint 7 ble gjennomført i perioden 11.05.2020 - 24.05.2020 (Uke 20-21).

Mål

Kapittel 2-3-4 i hovedrapporten skal ferdig stilles, testing og opprydding av applikasjons kode. Det blir viktig å bli ferdig med utviklingen, slik at systemdokumentasjon kan bli laget og resultater kan måles.

Avhengigheter og forutsetninger

For å få avsluttet utviklingen er det viktig at MVP blir ferdig. Slik at resultater blir klart, og kan brukes i rapporten. Møter med oppdragsgiver og veileder blir nødvendig for å justere planen slik at teamet blir ferdig i tide.

Resultat

Mange kritiske endringer ble gjort i sprint 7. Kapittel 2 og kapittel 3 blir ferdigstilt. Systemdokumentasjon for back-end blir ferdig, og venter på applikasjonen blir ferdig slik at det kan dokumenteres. Opprydding av feilmeldinger, knapper blir fikset for å hindre krasjer og andre problemer, slik at kildekode kan levers. gitLab blir ryddet og CI/CD blir oppdatert med flere tester. Hovedrapporten trenger utviklingen til å stoppe før kapittel 4, 5 og 6 kan skrives. Det ble gjort justeringer etter tilbakemelding fra veileder blir gjort på hovedrapporten. Kapitel 3 blir ferdig stilt, venter så på produktet for å beskrive riktige resultater i kapittel 4, 5 og 6. Problemstilling ble fullstendig formulert og beskrevet, slik at resultatene har en god formulert problemstilling.

Sprint 8

Sprint 8 ble gjennomført i perioden 25.05.2020 - 05.06.2020 (Uke 22-23).

Mål

Siste sprint blir å få alle dokumenter ferdige, med fokus på hovedrapporten.

Avhengigheter og forutsetninger

For å få en bra slutt kreves det at alle tidligere sprint gikk som planlagt og ble justert da det ble nødvendig, slik at prosjektet endte blir ett godt resultat og kan levere det som skal leveres innen firsten.

Resultat

Siste sprint, sprinten hvor siste endringer ble gjort, opprydding og ferdigstilling av systemdokument samt hovedrapporten. Sprinten startet med å gjøre noen få endringer på server og applikasjons kode, som løftet resultater for systemet på en høyere skale. Veileder ga tilbakemelding på systemdokumentasjon, og dokumentet ble ferdigstilt. Kapittel 4, 5 og 6 i hovedrapporten blir ferdig stilt, alle kapitler ble revidert slik at beste resultater blir levert da prosjektet blir levert.

Diskusjon

I dette kapittelet blir resultatene som ble beskrevet i forrige kapittel diskutert. Avvik og mangler blir også beskrevet, samt hvilke erfaringer teamet har anskaffet. Etiske problemstillinger blir tatt opp og diskutert.

5.1 Drøfting av vitenskapelige resultater

Problemstillingen for denne oppgaven var:

Hvordan utvikle en app med React Native, som forenkler prosessen for bestilling av nærmeste ledige drosje?

Punkt 1 omfatter utvikling av applikasjonen for Android og iOS. Når kravet om at applikasjonen skal lages for Androig og iOS, ble det kritisk å finne løsninger for hvordan dette kunne gjennomføres. Den valgt løsningen react native var noe som fungerte overaskende godt. Det å kunne utvikle en app for begge enhetene er med et programmeringsverktøy var banebrytende. Det minimerer arbeidsmengden, reduserer behovet for to utviklingsmiljø og kunnskapen som er nødvendig. React native er et hybrid rammeverk, som er gjør kildekode om til native kode for de enkelte enhetene. Det skjer uten å komprimere kvaliteten på native kode som blir kjørt på applikasjonen.

Punkt 2 består innebærer at det skal være enkelt for kunder i drosjemarkedet å bestille en drosje. Det ble ikke gjort noen vitenskapelig dokumentert etterforskning, men teamet gjennomførte en rask etterforskning for å se om løsningen oppdragsgiver mente ikke eksisterte, fantes. Den eksakte løsningen hvor kunden kun trykker en knapp, og nærmeste sjåfør ringer opp kunden for å arrangere transporten fantes ikke. Mange hadde løsninger hvor man kunne bestille med en knapp, men alle appene var upersonlige og tok ikke kontakt med kunden. Teamet fikk ikke tid for bruker tester. Men prosessen som var ønsket var på plass. Kunden bestiller med den eneste store knappen i sin startskjem, sjåføren ser denne ordren, sjåføren ringer så kunden for å arrangere transporten.

Punkt 3 tar for seg hvordan nærmeste sjåfør er den som utfører transporten for kunden. Nærmeste sjåfør blir den nærmeste sjåføren som er tilgjengelig i systemet. Dersom alle sjåfører bruker appen, har systemet mulighet til å gi ordren kun til den nærmeste sjåføren. Deretter sende den til flere hvis den ikke blir tatt. Kunden ønsker ikke å vente lenge og sjåføren ønsker å ha oppdrag man ikke trenger å kjøre langt for før man starter takstmeteret. Resultatene som er blit framstilt har ingen stopper for sjåføren som er lengre fra kunden sammenlignet med en sjåfør som er nærmere. I resultatene får sjåføren mulighet til å ta all ordre, uansett om det kanskje finnes andre sjåfører som er nærmere. Web socket ville løftet denne løsningen til det nivået som var planlagt.

5.2 Drøfting av ingeniørfaglige resultater

I starten viste ikke studentene hvordan de skulle utvikle en app for både Android og iPhone. Når teamet ble et medlem mindre ble det tydelig at det ikke kunne bli tilfellet. Under en workshop med problemstillingen for prosjektet, ble vi introdusert til react native. React native har gitt oss mulighet til å bruke arbeidsmengde nødvendig for utvikling innen en av plattformene, men der resultatet er en applikasjon for begge. Teamet hadde også ikke enheter for å utvikle native kode for iOS. I økonomiske- og tidsperspektiv var react native et valg uten negative sider.

Oppgaven var klar fra starten av, men fremgangsmåten for å løse oppgaven ble endret noen ganger underveis. Målet med at det skulle være lett å bestille en drosje var det ikke problemer med å oppnå, men selve køteorien ble komplisert og endret til en enklere form. Det at sjåføren får alle ordre er ikke en optimal løsning, men var den valgte løsningen da teamet ikke fikk implementert web socket. Med web socket kunne vi sendt kun ordre som var i den utvalgte distansen mellom kunden og sjåføren, og derfor fått varsler når en ny ordre oppsto.

Alle administrative funksjonaliteter ble nedprioritert, men react native er en utvidelse fra react. Et rammeverk for utvikling av bruk GUI. Hvis det var mulig ville det blitt laget en nettside hvor man kunne registrert nye sjåfører, sett vurderinger på sjåførene og hente ut fakturadetaljer slik at bedriften kan ta betalingen som var planlagt.

Det var noen funksjonaliteter som ikke ble implementert i appen, da det ble trangt om tid. Funksjoner som at sjåføren bekrefter at turen er fullført slik at man ikke kan ta flere ordrer på samme tid, avlyse ordre dersom kunden ikke møtte opp da sjåføren kom for å bli transportert. Dette var noe som trengtes for å ikke belaste sjåføren for en drosjetur som ikke ble utført. GUI for disse funksjonene er laget men mangler back-end. Vurdering av sjåføren etter endt tur mangler også tilknytting mellom GUI og back-end.

Sikkerheten i systemet ble nedprioritert, og det er tydelig at det ikke var planlagt implementering av god sikkerhet. Det ble lest opp på sikkerhet for å prøve å implementere et godt og sikkert system. Tiden det ville tatt, ville tatt mye tid ifra det å dokumentere prosjektet.

I løpet av prosjektet kom det endringer som endret hvordan sjåføren skulle bli en sjåfør i appen. Oppdragsgiver var usikker på hva slags løsning systemet skulle ha. Det å kvalitetskontrollere sjåførene ble flyttet ut av systemet teamet skulle lage, og erstattet med at administrator skulle ha mulighet til å registrere de, etter gjennomført kontroll. En nettside med informasjon om bedriften samt informasjon om hvordan man søkte for å bli en sjåfør ble den nye funksjonaliteten som ble ønsket. Dette ble ikke prioritert da vi hadde problemer med å få ferdig en prototype som kunne brukes som en MVP.

En meny knapp i applikasjonen ble ikke implementert, da det oppsto mangle problemer med å få dette på plass. I meny funksjonalisten skulle brukerne ha blant annet informasjon om hvordan de kan slette sin brukerkonto, lese personvern og servicevilkårene.

Resultatene i 4.2 viser at kjernestrukturen for systemet er på plass. Det er den som ble prioritert selv om ønsket var å implementere alle funksjonaliteter man kunne komme på. Oppdragsgiver var forstående over hvilke funksjoner som ble prioritert. Teamet har fått tilbud om å fortsette utvikling etter endt prosjekt. Dette er noe teamet og oppdragsgiver må ta opp etter levert rapport.

Utviklingsprosessen teamet hadde var bestandig agilt, men strukturen var ikke fastslått hardt nok til å komme med gode resultater på om prosessen var god eller ikke. De siste to sprintene gikk ikke etter planen, men fikk gjort det som måttes for å komme med gode resultater for applikasjonen, samt dokumentasjonen som skulle leveres. Det som muligens gjorde det vanskelig å holde seg til planene som ble lagt, var mangelen på medlemmer. Oppgaven var planlagt for en gruppe på størrelse mellom 3 til 4 studenter. Vi startet med 3 medlemmer og fikk inntrykk av at oppgaven var overkommelig innen tiden vi hadde. Da teamet ble 2, tok vi kanskje ikke god nok vurdering over hvor mye som skulle bli gjort, og prioritering av visse funksjonaliteter.

Hjemmekontor gjorde en stor del av alle avvik fra planene som originalt ble laget. Ikke minst var et medlem svært syk. Men alt begge medlemmer kunne gjøre, ble gjort for å fullføre oppgaven innen den nye tidsfristen etter utsettelse på grunn av sykdom. Hjemmekontor med familie er ikke hvordan man kan effektivisere tiden man bruker. Men begge medlemmene jobbet seg igjennom det.

Utviklingsteamet hadde god kommunikasjon med oppdragsgiver gjennom hele prosjektet. Det var ingen hindringer med å ta kontakt for spørsmål om hvordan funksjonaliteter skulle fremstilles, eller hvilke funksjonaliteter som skulle implementeres. Oppdragsgiver var enkel å forstå, beskrivende og tydelig på hva som var ønsket. Tok seg tid til å være tilgjengelig for møter på kort varsel. Lånte teamet en iOS enhet slik at utviklingsteamet kunne teste applikasjonen på iOS.

5.3 Drøfting av administrative resultater

Her vil vi diskutere hvordan de administrative resultatene. Først blir timeregnskap diskutert, deretter kommer utviklingsprosessen. Målet med dette er å få fram det som gikk etter planen, og eventuelt hva og hvorfor det ikke gikk som planlagt. Prosjekthåndboken er vedlegget som inneholder mye informasjon av det som blir diskutert.

5.3.1 Timeregnskap

Timer brukt per person ved endt prosjekt:

- **Karol:** 655 Timer
- **Kenneth:** 530 timer

Teamet arbeidet forskjellig i to forskjellige perioder. I første periode skulle teamet møtes hver skoledag mellom 08:00 til 15:30, så godt det kunne gjøres. I andre periode ble det hjemmekontor som ikke hadde faste tider hvor arbeid skulle gjøres. Kravet for arbeidstimer for prosjektet var 500 timer per person. Timelistene i prosjekthåndboken beskriver hvordan disse time blir brukt.

5.3.2 Utviklingsprosess

I tilknytning til utviklingsprosessen ble det laget en plan for hver sprint, se kapittel 4.3. Den inkluderer mål, avhengigheter og forutsetninger for hver sprint. Under ser man på hvilke erfaringer og avviket mellom planlagte mål og faktiske resultater for hver sprint.

Sprint 1

Avvik

Samarbeidet mellom alle medlemmene var ikke på plass, og endte med å miste et medlem. Ingen store avvik fra å fullføre målene satt for denne spriten.

Erfaring

Teamet erfarte at noen ikke er like gode til å samarbeide som andre. Det å få fullstendig forståelse av hvordan oppgaven skulle løses viste seg å ha litt utfordringer. Samarbeidet mellom studentene og oppdragsgiver ble testet og vist at det fungerte fint. Forståelsen var den samme, men hvordan oppgaven skulle utføres var litt vanskeligere. Kø teori med avstand gjorde ting vanskelig.

Sprint 2

Avvik

Teamet blir oppmerksom på att gant diagram ikke har blitt laget og sprints blir lagt inn i planen. Kenneth har gravid kone som gjorde at noen dager blir kortere enn planlagt.

Erfaring

Kenneth får erfart hvordan det er å starte familie på samme tid som å skrive en bacheloroppgave. Teamet har fått struktur og tilpasser seg etter hverandres timeplaner.

Sprint 3

Avvik

Vinterferie kom, uten at var en del av planen. Planene ble derfor endret litt og justert slik at teamet ikke hadde store avvik fremover. Kravdokument ble derfor ikke fastslått før på slutten av sprinten, og revisjonen utvidet sprinten ett par dager. Covid-19 pandemien kom, og vi fikk introdusert hjemmekontor mot slutten av sprinten.

Erfaring

Ferier og heildager må planlegges også da man oppretter en plan. Det var bra møtet med oppdragsgiver ble utført før hjemmekontor ble introdusert. Det ble viktig med en god gjennomgang av kravdokument og funksjonaliteter som skulle implementeres. Starten på utvikling startet neste sprint og god kommunikasjon med oppdragsgiver viser seg å være lettere enn forventet.

Sprint 4

Avvik

Hjemmekontor kommer med mange avvik. Sprinten startet godt, men effektiviteten faller kjapt. Karol ender opp med å bli syk, og setter sprinten på en midlertidig pause.

Erfaring

Hjemmekontor med familie er vanskelig. Det å ha familietid, samtidig som man skal utføre en jobb, er ikke lett. Når en er syk, mister andre deltakere motivasjonen til å jobbe for fult.

Sprint 5

Avvik

Dette sprintet blir nesten å gjenoppta forrige sprint, slik at vi kom på bane igjen. Selv om vi fikk mye hjelp med å få en server på ett miljø som kan nåes fra internett, var det mye problemer med å anskaffe sertifikat for node.js rammeverket.

Erfaring

Når man er syk er det likeså greit å lære nye ting, eller gjøre ting som kreves mye fokus. Kunnskap lært i sprint 4 måtte læres på nytt. Hjemmekontor begynner å få større problemer.

Sprint 6

Avvik

Sprintene og gantt måtte revideres etter sykdom. Oppstart av hovedrapport går tregt. Sertifikat for server hadde fortsatt ikke kommet på plass, en ny server ble tildelt og sertifikat ble anskaffet på slutten av spriten. Tester ble ikke gjort slik at de fungerte optimalt.

Erfaring

Det kan lønne seg å gå for en linux basert server, da windows har mindre frihet og dokumentasjon. Eneste som tok tid for anskaffelse for sertifikat på ubuntu serveren vi ble tildelt, var brannmurer som vi ikke hadde kontroll over. Da den ble flyttet, fikk vi sertifikat uten å måtte gjøre endringer på hvordan sertifikatet ble laget. Å lage tester for react native er vanskeligere enn planlagt.

Sprint 7

Avvik

Kapittel 1-2-3-4 skulle vært ferdig, men kun kapittel 1, 2 og 3 er de som nærmer seg ferdig. Uvikling av applikasjon og sever foregår fortsatt. Database må gjennomgå en oppdatering etter endringer i hvordan funksjonalitet blir utført. Noe som krever oppdatering av server kode, Samt applikasjon. GUI navigering i react native viste seg å være vanskeligere enn planlagt. Var nære gode resultater og klarte ikke å stoppe. Ettersom det fortsatt utvikles er det ikke mulig å skrive om resultater, noe som stopper mye dokumentasjon.

Erfaring

GUI navigering i react native tar mye tid. Noe som tar tid fra andre oppgaver. En grundig planlagt database viser seg å være viktig, da det kan endre mye struktur i serverkode.

Sprint 8

Avvik

Utvikling skjedde i første del av siste sprint. Systemdokumentasjon ble ikke ferdig før siste del. Planene ble ikke fulgt, men slutt resultatet ble bra. For å få alle ting på plass måtte arbeidstidene overstige de som var planlagt.

Erfaring

Det å ikke følge planen kan lønne seg. De siste endringene som ble implementert løftet resultater opp på ett høyere nivå. Det kom på bekostning av at mange timer måtte skje på en kort periode.

5.4 Etiske problemstillinger

Når man utvikler applikasjoner for små eller store systemer, er det mye å tenke på når det gjelder etikk. Graden av etiske problemstillinger varier mye, både store og små systemer må ta høyede for disse. Når man utvikler systemer som har stor kundedatabase, som også inneholder sensitive personopplysninger, er det spesielt viktig å holde kontroll over hvordan noe blir håndtert. Man må derfor ta for seg de profesjonsetiske spørsmålene, se på hvilke konsekvenser systemet kan ha i en samfunnsmessig, økonomisk- og miljømessig sammenheng.

5.4.1 Profesjonsetiske spørsmål

Sporing av brukere blir gjort for å få den ønskede funksjonaliteten. Dette må behandles og minimeres så godt det lar seg gjøre. Det blir også lagret en del informasjon av sjåfør brukerne. Alt dette blir brukerne informert om ved starten av appen. Hvor lenge dette blir tatt vare på og hvordan prosessen for hvordan man fjerner seg fra systemet blir gjort. Det er viktig at ingen andre kan ta denne informasjonen ut ifra kommunikasjonen mellom de forskjellige enhetene.

5.4.2 Økonomiske konsekvenser

Økonomiske verdier har ikke blitt utregnet, men ønsket med systemet er å gjøre drosjemarkedet til et konkurransedyktig marked. Kundene i systemet blir belastet en ekstra sum, mot det å få en kjappere og bedre service innen lokalgeografiske- og norsk kunnskaper. Det skal generere en større markedsandel for de sjåførene som fortjener det. Valget av nærmeste sjåfør minske transportutgifter hos sjåførene.

5.4.3 Miljømessige konsekvenser

Med å ta den nærmeste sjåføren til en kunde kan man teoretisk sett minske klimautgifter. Det har kommet frem at mengden datatrafikk begynner å bli en større post i negative miljøkonsekvenser. Det er ikke gjort mye etterforskning på om det å søke i en søkemotor for så å finne et drosjeselskap man ønsker å benytte, for så å bestille drosje, opp mot det å laste ned en app for så å sende en etterspørsel via applikasjonen for så å bli kontaktet for transport, er mest miljøvennlig. Men systemet vi lager minimerer antall enheter etterspørselen går igjennom før sjåføren får muligheten til å ta oppdraget, og håpet er at en mer direkte kontakt mellom kunden og sjåføren er en løsning som kan minimere datatrafikk som blir brukt.

5.5 Drøfting av gruppearbeidet

Teamet bestod av tre studenter i starten, men endte opp som et team på to. Teamet har jobbet godt sammen gjennom hele prosjektet, med noen få uenigheter. Disse uenighetene ble tatt opp med veileder eller oppdragsgiver og løst fort. Begge har familier hjemme og har møtt store problemer med å holde fokus med hjemmekontor, og det var en felles forståelse om hvilke utfordringer som oppstod. Teamet hadde videomøter under hjemmekontor da i tilfeller da det var nødvendig med en større diskusjon rundt et emne. Kommunikasjonen mellom studentene har vært åpen og begge har følt de kunne komme

med sine ider og innspill. Selv om studentene var delt opp i front-end og back-end, var det fortsatt et godt samarbeid og gjensidig støtte da problemer eller usikkerheter oppstod.

Konklusjon og videre arbeid

I dette kapittelet konkluderer teamet på hvor godt problemstillingen fra kapittel 1 blir nådd, med bakgrunn i resultatene og diskusjonen. Siste del av kapittelet tar for seg hvordan videre utvikling ville blitt utført, forslag til nye funksjonaliteter og forbedringer.

6.1 Konklusjon

Utviklingsteamet har konkludert med fullføring av oppgaven, i tråd med den formulerte problemstillingen, i stor grad er ansett som vellykket. Det var nære på at resultatene ikke kom på et nivå hvor de kunne komme til denne konklusjonen. Oppdragsgiver var fornøyd med endt resultat, tatt i betraktning utfordringene som oppstod underveis.

Deler av oppgaven som ble tildelt ble endret underveis, og en del av kravene ble fjernet. Det ble ikke brukt noen tid for utvikling av disse funksjonene, men det tok litt tid i første fase i å få forståelse av hva oppgaven faktisk bestod av. Kjernefunksjonalitetene var bestandig de samme og ble oppnådd mot slutten av prosjektet.

Problemstillingen var som følger:

Hvordan utvikle en app med React Native, som forenkler prosessen for bestilling av nærmeste ledige drosje?

Ut ifra flyten i applikasjonen i 4.2, ser vi at funksjonaliteten som utførte forenklingen av bestillingsprosessen, er på plass. Muligheten for at nærmeste sjåfør tar orden og utfører transporten er der, selv om noen som er lengre unna i prinsippet kan ta ordren. Disse to punktene er godt oppfylt. Utvikling med rammeverket react native er dokumentert i systemdokumentasjonen, jsDoc, og kan sees på som en stor suksess. Android og iOS applikasjonen fungerer likt, har samme GUI og gir samme resultat. Dette uten å programmere noen kildekode på en apple enhet. Ansvaret for å ta oppdrag er gitt til sjåføren, slik at de som er effektive får økt sin lønnsomhet.

6.2 Videre arbeid

Det er mye videre arbied for at dette systemet skal bli slik som planlagt, også oppgraderinger som oppdragsgiver ikke tenkte på i første omgang. Det er lagt til rette for at videre utvikling av applikasjonen er mulig. Alle funksjoner som var planlagt men ikke implementert er dokumentert, GUI for de fleste skjermer er laget. Det er lett å flytte kildekode over på ett nytt nivå, uten store problemer. Den største mangelen for systemet er god sikkerhet. Kryptering mellom databasen og server ble ikke satt opp, men skulle blitt gjort. Alle mangler som er beskrevet i 5.2 er klare for videreutvikling. Med endring av oppkoblingen mellom server og applikasjonen til web sokket, ville man minimert mengden data som blir

sendt. Man får en kjappere responstid på oppdateringer, og man kunne lagt opp for flere funksjonaliteter som ville forbedret applikasjonen.

Referanser

[1] B. &. K. N. Forssell, “Store Norske Leksikon,” 2020. [Online]. Available: https://snl.no/GPS. [Accessed 17 April 2020].

[2] T. Logemann, “intersoft consulting,” 2018. [Online]. Available: https://gdpr-info.eu/. [Accessed 20 April 2020].

[3] “NRK - Spalte med siste info på toppen av siden,” [Online]. Available: https://www.nrk.no/. [Accessed 05 06 2020].

[4] “Drosjetransport - årlig - SSB,” [Online]. Available: https://www.ssb.no/transport-og-reiseliv/statistikker/drosje. [Accessed 05 06 2020].

[5] S. Stephansen, “Regjeringen,” 2020. [Online]. Available: https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/. [Accessed 23 April 2020].

[6] Uber, “Uber Newsroom,” 2020. [Online]. Available: https://www.uber.com/nb-NO/newsroom/historikk/. [Accessed 18 May 2020].

[7] D. Bach, “David BachDavid Bach,” 2019. [Online]. Available: https://e24.no/naeringsliv/i/AdkBEq/ubers-norden-sjef-varsler-comeback-i-norge. [Accessed 23 April 2020].

[8] A. R. Sverre, “Erfaringer med deregulering av drosjemarkedet i,” Oslo Economics, Oslo, 2019.

[9] S. S. Tor Midtbø, “Regjeringen,” 2020. [Online]. Available: https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/. [Accessed 22 April 2020].

[10] B. E. Thon, “Datatilsynet,” 2019. [Online]. Available: https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/. [Accessed 24 April 2020].

[11] O. Storm-Paulsen, “Lovdata, Personopplysningsloven,” 15 Juni 2018. [Online]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38. [Accessed 22 April 2020].

[12] “Lov om behandling av personopplysninger (personopplysningsloven),” [Online]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38/gdpr/ARTIKKEL_83#gdpr/ARTIKKEL_83. [Accessed 03 06 2020].

[13] “Hva er nytt med personvernforordningen?,” [Online]. Available: https://www.datatilsynet.no/regelverk-og-verktoy/lover-og-regler/hva-er-nytt/. [Accessed 03 06 2020].

[14] H. Øverby, “smarttelefon i Store norske leksikon,” 2018. [Online]. Available: https://snl.no/smarttelefon. [Accessed 27 April 2020].

[15] A. Dossey, “Clearbridge mobile,” 2019. [Online]. Available: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/. [Accessed 27 April 2020].

[16] R. Native, “React Native,” 2019. [Online]. Available: https://reactnative.dev/. [Accessed 27 April 2020].

[17] “Web framework ranking,” [Online]. Available: https://hotframeworks.com/. [Accessed 04 06 2020].

[18] T. Occhino, “Facebook Engineering,” 2015. [Online]. Available: https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/. [Accessed 22 April 2020].

[19] R. G. J. R. Bhairav Acharya, “Cell Phone Location Tracking,” Samuelson Law, Technology & Public Policy Clinic, UC Berkeley, 2016.

[20] M. Smallcombe, “Xplenty,” 2019. [Online]. Available: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/. [Accessed 24 April 2020].

[21] K. Sander, “estudie,” 2019. [Online]. Available: https://estudie.no/nettverksserver/. [Accessed 28 April 2020].

[22] D. d. Borde, “Two-factor authentication,” Siemens Insight Consulting , UK, 2007.

[23] K. Sander, “estudie,” 2019. [Online]. Available: https://estudie.no/ssl/. [Accessed 29 April 2020].

[24] Acuentix, “Acuentix,” 2020. [Online]. Available: https://www.acunetix.com/websitesecurity/sql-injection/. [Accessed 29 April 2020].

[25] G. B. C. L. B. V. Pete Deemer, “THE SCRUM PRIMER,” 2009. [Online]. Available: https://docplayer.net/15545261-The-scrum-primer-by-pete-deemer-gabrielle-benefield-craig-larman-bas-vodde-version-1-1-certified-scrum-training-worldwide-www-scrumti.html. [Accessed 25 Mai 2020].

[26] J. K, “Acodez,” 01 Juni 2018. [Online]. Available: https://acodez.in/12-best-software-development-methodologies-pros-cons/. [Accessed 25 Mai 2020].

[27] flexify, “flexify,” 2019. [Online]. Available: https://www.flexify.no/blogg/agile-metoder. [Accessed 25 Mai 2020].

[28] O. Storm-Paulsen, “Lovdata,” 15 Juni 2018. [Online]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38. [Accessed 22 April 2020].

[29] “CS50's Mobile App Development with React Native,” [Online]. Available: https://courses.edx.org/courses/course-v1:HarvardX+CS50M+Mobile/course/. [Accessed 2020].

[30] “Expo Documentation,” [Online]. Available: https://docs.expo.io/. [Accessed 2020].

[31] “React Navigation,” [Online]. Available: https://reactnavigation.org/. [Accessed 2020].

[32] “A Predictable State Container for JavaScript Apps,” [Online]. Available: https://redux.js.org/. [Accessed 2020].

[33] “WebStorm,” [Online]. Available: https://www.jetbrains.com/webstorm/. [Accessed 2020].

[34] “Let's Encrypt - Free SSL/TLS Certificates,” [Online]. Available: https://letsencrypt.org/. [Accessed 2020].

[35] “Jest | Delightful JavaScript testing,” [Online]. Available: https://jestjs.io/.

[36] “JSDoc documentation,” [Online]. Available: https://jsdoc.app/. [Accessed 05 2020].

[37] “Prettier - Opinionated Code Formatter,” [Online]. Available: https://prettier.io/. [Accessed 05 2020].

[38] G. B. C. L. B. V. Pete Deemer, “THE SCRUM PRIMER,” 2012. [Online]. Available: https://scrumprimer.org/scrumprimer20.pdf. [Accessed 24 Mai 2020].

Vedlegg

6.1 Visjonsdokument

6.2 Kravdokument

6.3 Systemdokument

6.4 Bacheloravtale

6.5 Original oppgavetekst

Fast Track Taxi
Visjonsdokument

Versjon 1.1

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
20.02.2020	1.0	Første utgave.	Karol Debik, Kenneth L Krogstad, Kevin Andre Helgeland
28.01.2020	1.1	Endret utydelige formuleringer slik at de passet bedre.	Karol Debik, Kenneth L Krogstad

Innholdsfortegnelse

1. Innledning

Dette dokumentet ble laget for å gi et klart bilde over hva problemet er med dagens system når det gjelder drosjebransjen, hvilket produkt som blir laget og hvordan dette produktet løser problemet.

Sommeren 2020 blir det innført nytt drosjeregelverk. Det sannsynligvis kommer mange nye drosjeselskaper som skal konkurrere om kunder ved å tilby billigere tilbud. Det vil komme på bekostning av service.

Produktet skal gi adgang til sjåfører som blir kvalitetssikret innen norsk- og lokal geografi og kunder som ønsker seg denne kvalitetssikringen.

Det skal lages en Android og iOs applikasjon som skal ha funksjonalitet for både kunder og sjåfører.

1.1 Referanser

Stephanse S, 2019, Spørsmål og svar om nytt drosjeregelverk, *Regjeringen.no,* Tilgjengelig fra:

https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Problem med	en del drosjesjåfører som ikke behersker norsk og mangler lokale geografikunnskaper
berører	alle drosjekunder
som resultatet av dette	oppstår det misforståelser når det gjelder kundens destinasjon
en vellykket løsning vil	gi en plattform der norsk- og lokal geografikunnskapene er kvalitetssikret og misforståelser blir redusert.

Problem med	stort antall drosje selskaper

berører	drosjesjåfører
som resultatet av dette	reduseres antall turer og lønnen
en vellykket løsning vil	øke antall turer til sjåfører som bruker dette produktet.

2.2 Produktsammendrag

For	kunden
som	opplever misforståelser med drosjesjåføren om kundens destinasjon
produktet	Fast Track Taxi
som	gir kunden tilgang til drosjesjåfører som behersker norsk og lokal geografi
I motsetning til	dagens system hvor kunden må velge ett drosjeselskap, i stedet for nærmeste drosje
Har vårt produkt	kvalitetssikrede drosjesjåfører, og finner nærmeste drosje uavhengig av drosjeselskap.

For	drosjesjåfører
som	behersker norsk og lokal geografi og ønsker en større andel av drosje turer
produktet	Fast Track Taxi
som	har tilgang til kunder som setter pris på å snakke med sjåføren og at sjåføren vet hvor den skal kjøre.
I motsetning til	dagens system hvor sjåførene kun får tilbud om turer fra drosjeselskapet de tilhører, eller ved å finne kunder på gaten
Har vårt produkt	en andel av drosjekunder, som ønsker sjåfører med god norsk og lokal geografi kunnskaper.

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen

Mi og Ma holding AS	Arne Pukstad Juliussen, Oppgavestiller	Tar avgjørelser om hvordan funksjonaliteter som skal være i produktet og produktets design.
Student	Karol Debik	Designer, dokumentasjon, møteleder, tester.
Student	Kenneth Langdahl Krogstad	Designer, dokumentasjon, referent, programmerer.

3.2 Oppsummmering brukere

Navn	**Utdypende beskrivelse**	**Rolle under utviklingen**	**Representert av**
Kunde	Den som ønsker å bestille en drosje	Tester systemet	Av seg selv
Sjåfør	Den som tar oppdrag om henting av kunde	Tester systemet	Av seg selv
Admin	Administrerer sjåfører, priser og henter lister for faktura	Tester systemet	Av seg selv

3.3 Brukermiljøet

Fast Track Taxi skal være en app som skal hovedsakelig strekke seg mot kunder som søker ekstra kvalitet i drosjebransjen i form av beherskelse i norsk og lokale geografikunnskaper til sjåførene. Reglene på standarden til drosjesjåfører blir mindre, så vil det bli lettere for nye drosjeselskaper å starte opp med billigere drosjetilbud og mindre kontroll på kvalitet på sjåføren.

3.4 Sammendrag av brukernes behov

Behov	**Prioritet**	**Påvikrer**	**Dagens løsning**	**Foreslått løsning**
En enkel måte å bli hentet av nærmeste drosje	Høy	Kunde	På bedrifts basis	En ny plattform som finner den nærmeste

				drosjen for kunden uavhengig av selskap
Redusere tid brukt i kø	Middels	Kunde	Ingen	Bli prioritert i køen
Se egen historikk	Lav	Kunde	Ingen	Vise historikk
Skille mellom dårlige og gode sjåfører	Høy	Sjåfør	På bedrifts basis	En ny platform for å finne gode sjåfører
Finne flere kunder	Middels	Sjåfør	Stå på rett plass, bedriftens system	Systemet finner flere kunder enn enkelte bedrifter har
Se egen historikk	Middels	Sjåfør	Ingen	Vise historikk
Se historikk for sjåfører	Høy	Admin	Ingen	Vise historikk av sjåfører
Legge til en drosjesjåfør i system	Middels	Admin	Ingen	En egen løsning for å legge til en ny drosjesjåfør

3.5 Alternativer til vårt produkt

Det finnes andre apper som gjør omtrent det samme, men de er begrenset til hver enkel bedrift.

4. Produktoversikt

4.1 Produktets rolle i brukermiljøet

Denne appen har tre hovedmål som man skal oppnå for kunden:

1. **Kvalitet på drosjesjåføren**
2. **Enkelt å bestille**
3. **Hurtig bestilling**

Samtidig, så ønsker man å skape en ekstra nisje innen drosjemarkedet som er rettet mot kvalitet på drosjesjåføren.

Kunden kan velge bedre kvalitet i stedet for lavere pris.

4.2 Forutsetninger og avhengigheter

Denne appen forutsetter at det ikke er noe som hindrer en sjåfør å ta ekstra betalt utenfor den ordinære pris per kilometer, eller generelt noe samarbeid med tredjeparts bedrifter.

Appen er avhengig av å få tillatelser til lokasjon på enheten.

5. Produktets funksjonelle egenskaper

Egenskap	**For**	**Prioritet**
Se pris for påslag ved bestilling av drosje	Alle	Mid
Se pris for prioritering i kø	Alle	Høy
Se at noen ønskes å bli hentet	Sjåfør	Høy
Se at kunde er prioritert kunde	Sjåfør	Høy
Akseptere ett oppdrag	Sjåfør	Høy
Kansellere ett oppdrag	Sjåfør, Kunde	Mid
Få nummeret til kunden	Sjåfør	Høy
Ringe/sende melding til kunden	Sjåfør	Høy
Vurdering av en kunde	Sjåfør	Lav
Laste ned sjåfør funksjonalitet	Sjåfør	Høy
Sette seg som inaktiv	Sjåfør	Mid
Endre informasjon om seg selv	Sjåfør	Lav
Logge seg inn	Admin	Høy
Lage en ny admin	Admin	Lav

Endre pris for påslag ved bestilling av drosje	Admin	Lav
Endre pris for prioritering i kø	Admin	Lav
Verifisere en sjåfør slik at brukeren blir aktiv	Admin	Mid
Se vurdering på en sjåfør	Admin	Lav
Se vurdering på kunde	Admin	Lav
Blokkere ett nummer	Admin	Lav
Logge seg ut	Admin	Mid
Sende in vurdering av sjåfør etter tur	Kunde	Mid
Sette ønske om melding eller ringing ved henting	Kunde	Lav
Sende ut oppdrag om henting	Kunde	Høy
Registrere seg	Kunde, Sjåfør	Høy
Endre mobil nummer	Kunde	Lav
Se om det er ledige drosje i nærheten	Kunde	Mid
Se at du er prioritert	Kunde	Mid
Verifisere seg med engangskode, fått på melding	Kunde, Sjåfør	Mid
Sende oppdrag til de nærmeste ledige sjåførene innen en viss distanse.	System	Høy
Sjekke om en kunde og en sjåfør flytter samme vei	System	Lav

Ved kanselering av oppdrag sjekke tillatelse til lokasjon hvis ikke rapportere det	System	Lav
Få lokasjon til brukere	System	Høy

6. Ikke-funksjonelle egenskaper og andre krav

Brukbarhet:

Denne appen skal være enkel å bruke, slik at kunden kan laste den ned og mestre den innen kort tid.

Det er en liten tutorial, der du får informasjon om funksjonalitetene i appen.

Vi skal følge WCAG 2.0 standarden for å designe Interface.

Pålitelighet:

Gjennomsnittlig skal man kun forvente en feil i året per bruker. Det skal være høyt fokus på pålitelighet, så det blir en lav feil toleranse siden denne appen skal være så enkel som mulig for brukeren, siden det er et luksusprodukt.

Man skal forvente minst 70% kode dekning av tester i koden.

Ytelse:

I første omgang skal man kunne forvente at man skal håndtere 100 brukere om gangen, der appen i seg selv skal kunne yte på 8ms.

Støtte:

Eieren av produktet tar seg av videreutvikling og vedlikehold av produktet.

Implementasjon:

Dette er en app som skal lanseres til både Android produkter (Play store) og Apple produkter (App store).

Interface:

Det vil være en Kunde og Sjåfør Interface på Android og Apple, mens Admin for systemet blir på en web-side.

Operasjon;

Eieren av produktet tar seg av videre drift av systemet.

Pakke løsning:

På Admin siden er det en installasjon av en server og en database.

Bruker Interface blir lastet ned som standard ved nedlastning fra app butikker.

Lover:

Vi må ha samsvar med de reglene i GDPR for lagring av data for brukerne.

7. Referanser

Stephanse S, 2019, Spørsmål og svar om nytt drosjeregelverk, *Regjeringen.no,* Tilgjengelig fra:

https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/

Prosjektnr 040

040_Fast Track Taxi

Kravdokumentasjon

Versjon 0.2

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
28.02.2020	0.1	Utkast 1	Karol Debik, Kenneth L Krogstad
	0.2	Reviderer user stories, sekvensdiagram og wireframes etter møte med arne	Karol Debik, Kenneth L Krogstad

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet ble laget for å få en oversikt over hvilke krav som skal gjelde for produktet "Fast Track Taxi". I dokumentasjonen vil du finne User stories som beskriver hva en bruker vil oppleve ved bruk av systemet. Det skal også dokumenteres sekvensdiagram slik at man ser hva systemet skal gjøre, samt domenemodell og databasemodell slik at man kan lage systemet fra å lese dette dokumentet. Det ble laget en digital Wireframe hvor man kan navigere seg igjennom produktet som en bruker, slik at man forstår hvordan systemet skal settes opp. Skjermbildene fra den digitale Wireframe ble overført til dette dokumentet og beskrevet.

2. User Stories

2.1 User Story 1

Som en bruker

Ønsker jeg å akseptere betingelsene angående personopplysninger

Slik at jeg kan bruke appen.

Scenario: Akseptere betingelsene angående personopplysninger

Gitt at det er første gang jeg starter appen

Når appen har lastet inn

Så kommer det opplysninger om hvilke personopplysninger som blir lagret om meg

Og jeg må trykke på aksepter knappen før jeg kommer videre i appen.

2.2 User story 2

Som en bruker
Ønsker jeg å registrere meg
Slik at jeg kan bruke appens funksjoner.

Scenario: ved første start av appen
Gitt at jeg er en ny bruker
Når jeg åpner appen

Så ser jeg informasjon om personvern

Og jeg trykker “akseptere” for å fortsette.

Scenario: Verifisere mobilnummer

Gitt at jeg har akseptert personvern innstilligene jeg har skrevet inn nummeret mitt
Så skal jeg trykke neste

Og bli logget inn automatisk av en SMS med engangskode.

Scenario: Verifisert som kunde

Gitt at jeg ble godkjent med engangskoden og ikke er sjåfør i systemet

Så blir jeg logget inn som en kunde

Og jeg kan bruke alle appens funksjoner som en kunde

Scenario: Verifisert som sjåfør

Gitt jeg ble godkjent med engangskoden og jeg er en godkjent sjåfør

Så blir jeg logget inn som en sjåfør

Og jeg kan bruke alle appens funksjoner som en sjåfør

2.3 User story 3

Som en bruker

Ønsker jeg gi appen tillatelse til

Slik at jeg kan bruke appens funksjoner.

Scenario: Appen trenger lokasjon

Gitt at har blitt verifisert med engangskode

Når jeg starter appen

Så skal det komme opp en beskjed om hvilke tillatelser jeg må akseptere før appen fungerer

Og jeg må trykke "akseptere” før appen fungerer i sin helhet

2.4 User story 4

Som en bruker

Ønsker jeg å lukke appen

Slik at jeg kan bruke resten av enheten.

Scenario: lukke appen

Gitt at jeg er logget inn

Når jeg trykker på “meny”

Så kommer det en liste, hvor nederst det står “lukk app”

Og når jeg trykker på denne, lukkes appen.

2.5 User story 5

Som en bruker

Ønsker jeg å lese “om appen”

Slik at jeg kan bli mer informert om appen.

Scenario: om appen

Gitt at jeg er logget inn

Når jeg trykker på “meny”

Så kommer det en liste, hvor nest nederst sår det “om appen”

Og når jeg trykker på denne kommer jeg til en side med informasjon om appen.

2.6 User story 7

Som en kunde

Ønsker jeg å bestille en drosje

Slik at en drosje kan hente meg.

Scenario: Bestille drosje

Gitt at jeg en kunde

Når jeg trykker på “bestill taxi”

Så blir det laget ett oppdrag på serveren

Og jeg må vente til en sjåfør tar oppdraget.

2.7 User story 8

Som en kunde

Ønsker jeg å bli prioritert oppdrag

Slik at jeg blir prioritert først og slipper å vente så lenge for å komme hjem.

Scenario: Bli prioritert

Gitt at jeg er kunde som står i kø for henting av taxi

Når det kommer opp beskjed om at det ikke er noen ledig taxi for øyeblikket

Så kommer det en knapp “bli prioritert” som jeg trykker på

Og viser da for sjåførene i appen at jeg er en prioritert kunde.

2.8 User story 9

Som en kunde

Ønsker jeg å avbestille turen

Slik at jeg slipper å betale for noe jeg ikke trenger lengre.

Scenario: avbestille turen som kunde

Gitt at jeg er en kunde som venter å bli hentet av en taxi

Når jeg trykker på knappen “kanseller ordre”

Så spør den om jeg er sikker

Og når jeg trykker ja, kommer jeg hjem siden for en kunde.

2.9 User story 10

Som en kunde

Ønsker jeg å få vite at sjåføren har kansellert oppdraget

Slik at jeg kan sette meg opp for en ny ordre

Scenario: sjåfør kansellerte hentingen

Gitt at sjåføren har ringt kunden for henting

Når sjåføren har kansellert

Så ringer telefonen min og sjåføren sier noe har skjedd

Og jeg må sette meg opp for en ny taxi tur.

2.10 User story 11

Som en kunde

Ønsker jeg å vurdere sjåføren etter endt tur

Slik at jeg bidrar til å luke ut dårlige sjåfører.

Scenario: Vurdere sjåføren etter endt tur

Gitt at jeg er ute av taxien

Når sjåføren har trykket på “fullført tur”

Så får jeg en notifikasjon som sender meg til en side i appen

Og jeg kan nå sette 1-5 stjerner på sjåføren.

Scenario: Vurdere sjåføren etter endt tur

Gitt at jeg er ute av taxien

Når det har gått 10 minutter siden turen ble akseptert

Så har jeg muligheten til å vurdere sjåføren

Og jeg kommer til hoved skjermen.

2.11 User story 12

Som en kunde

Ønsker jeg å vite hvor mye det koster for bestilling av et oppdrag, og hva det koster å bli prioritert

Slik at jeg ikke ender opp i situasjoner med sjåføren angående penger.

Scenario: Vite priser

Gitt at jeg en kunde

Når jeg ser på knappene om bestilling eller å bli prioritert

Så står prisene på knappene

Og jeg blir ikke overrasket over prisene i taxien.

2.12 User story 13

Som en kunde

Ønsker jeg å få notifikasjon om at det ikke var noen ledig taxi

Slik at jeg kan sette meg opp for å bli prioritert.

Scenario: notifikasjon prioritert kunde

Gitt at jeg har bestilt taxi

Når serveren ikke har funnet en taxi

Så skal jeg få en notifikasjon som informerer meg om dette

Og jeg kommer til siden for å sette meg som prioritert hvis jeg trykker på notifikasjonen.

2.13 User story 14

Som en sjåfør

Ønsker jeg å velge at jeg er en sjåfør

Slik at jeg kan bruke appens funksjoner som en sjåfør.

Scenario: valg av bruker som sjåfør
Gitt at jeg er verifisert og kommer til siden av valg av bruker

Når jeg trykker på sjåfør
Så skal jeg skrive inn navn, taxinummer, taxieier, faktura adresse og organisasjonsnummer
Og trykker “bekreft”.

Scenario: Vente på godkjenning som sjåfør
Gitt at jeg har sendt inn person opplysningene

Når jeg trykker “bekreft”
Så kommer jeg til en side hvor jeg får opplysninger om søknaden min blir behandlet

og må vente på å få bli bekreftet før jeg får bruke appen.

2.14 User story 15

Som en sjåfør

Ønsker jeg å kansellere oppdraget

Slik at kunden kan finne en ny taxi.

Scenario: kansellere oppdraget

Gitt at jeg er en sjåfør med et oppdrag

Når jeg ser at kunden ikke møtte opp

Så skal jeg trykke på “kansellere oppdraget”

Og jeg kommer ut av oppdraget.

Scenario: kansellere oppdraget annen grunn

Gitt at jeg er en sjåfør med et oppdrag

Når det oppstår ett problem slik at jeg ikke kan hente kunden

Så skal jeg trykke på “kansellere oppdraget”, og velger noe annet, og beskriver hva som gikk feil

Og jeg ringer kunden for å informere den.

2.15 User story 16

Som en sjåfør

Ønsker jeg å ta ett oppdrag

Slik at jeg kan kjøre en kunde.

Scenario: Ta ett oppdrag

Gitt at jeg er en sjåfør som er aktiv

Når jeg har ett oppdrag i listen av oppdrag

Så trykker jeg akseptere

Og jeg vil automatisk ringe kunden.

2.16 User story 17

Som en sjåfør

Ønsker jeg å se at ett oppdrag er prioritert

Slik at jeg kan tjene mest mulig per tur.

Scenario: prioritert oppdrag

Gitt at det er ett oppdrag i oppdrags listen som er prioritert oppdrag

Når jeg ser den i listen, står det tekst om at den er prioritert og er i annen farge slik at jeg ser at den er ett prioritert oppdrag

Så jeg trykker aksepter

Og jeg vil automatisk ringe kunden.

2.17 User story 18

Som en sjåfør

Ønsker jeg å fullføre ett oppdrag

Slik at jeg kan ta ett nytt oppdrag.

Scenario: Fullføre ett oppdrag

Gitt at jeg har akseptert ett oppdrag

Når jeg er ferdig med oppdraget

Så skal jeg trykke på fullført tur

Og jeg vil komme tilbake til startsiden.

2.18 User story 19

Som en sjåfør

Ønsker jeg å se historikk

Slik at jeg kan sjekke at detaljene stemmer med faktura.

Scenario: Historikk

Gitt at jeg er en sjåfør

Når jeg trykker på “meny”

Så kommer det en liste med flere valg, hvor en av de er “historikk”

Og når jeg trykker på denne blir jeg ført videre til en liste med alle kjøringene jeg har gjort med appen.

2.19 User story 20

Som en sjåfør

Ønsker jeg å sette meg selv som tilgjengelig

Slik at jeg kan få oppdrag.

Scenario: tilgjengelighet

Gitt at jeg er en sjåfør og står som utilgjengelig

Når jeg er på hjemmesiden

Så ser jeg at jeg er utilgjengelig

Og jeg trykker på knappen slik at det står jeg er tilgjengelig.

2.20 User story 21

Som en admin

Ønsker jeg å få faktura detaljer for alle sjåfører

Slik at jeg kan sende faktura.

Scenario: faktura detaljer

Gitt at en e-post er i systemet

Når det er en ny måned

Så blir det sendt epost til e-posten som er i systemet med faktura detaljer for alle sjåførene

Og jeg kan laste ned dette fra innboksen.

2.21 User story 22

Som en admin

Ønsker jeg å se vurdering på en sjåfør

Slik at vet om jeg bør slette en sjåfør.

Scenario: Se vurdering

Gitt at jeg er logget inn i databasen

Når jeg søker med parameter sjåfør under 2 stjerner

Så får jeg en liste over alle sjåfører under 2 stjerner

Og jeg kan skrive slett “sjåfør x” fra databasen.

2.22 User story 23

Som en admin

Ønsker jeg å verifisere en sjåfør

Slik at sjåføren kan begynne å ta oppdrag.

Scenario: Verifisere en sjåfør

Gitt at jeg er logget inn i databasen

Når jeg søker etter sjåførens navn

Så kan jeg endre parameteren “verifisert” til aktiv

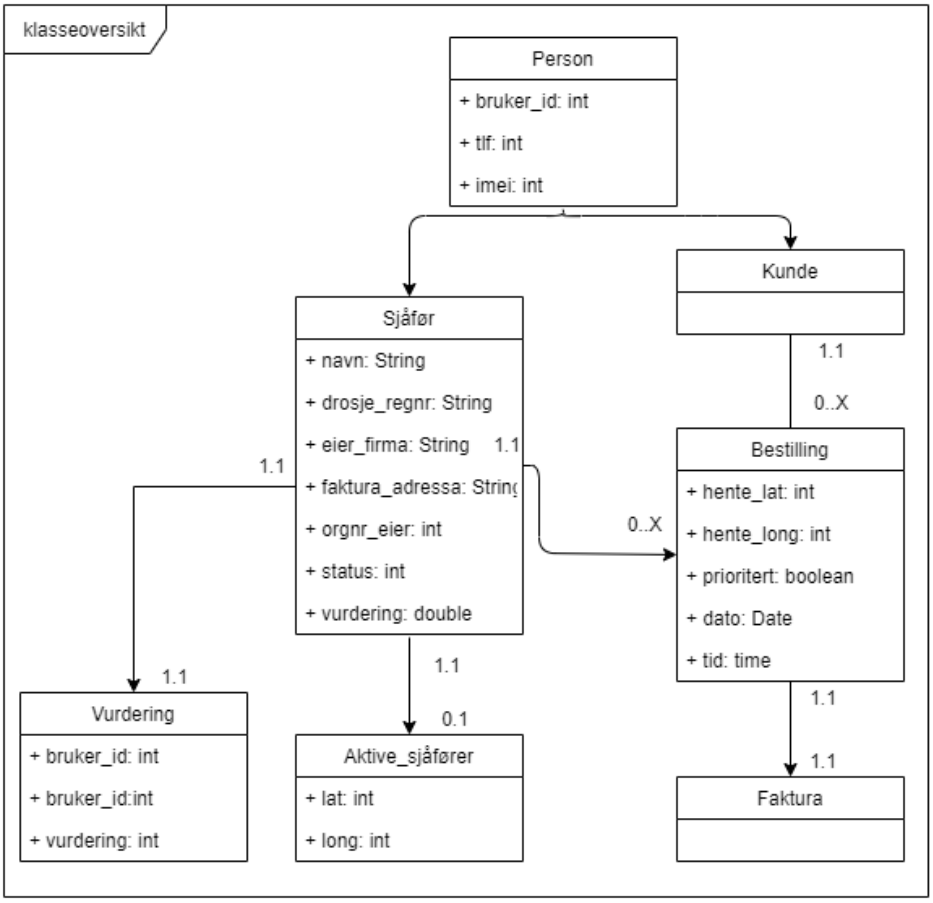
Og sjåføren får beskjed om dette.

3. Domenemodell

Her blir systemet beskrevet med klasseoversikt, databasemodell og sekvensdiagram.

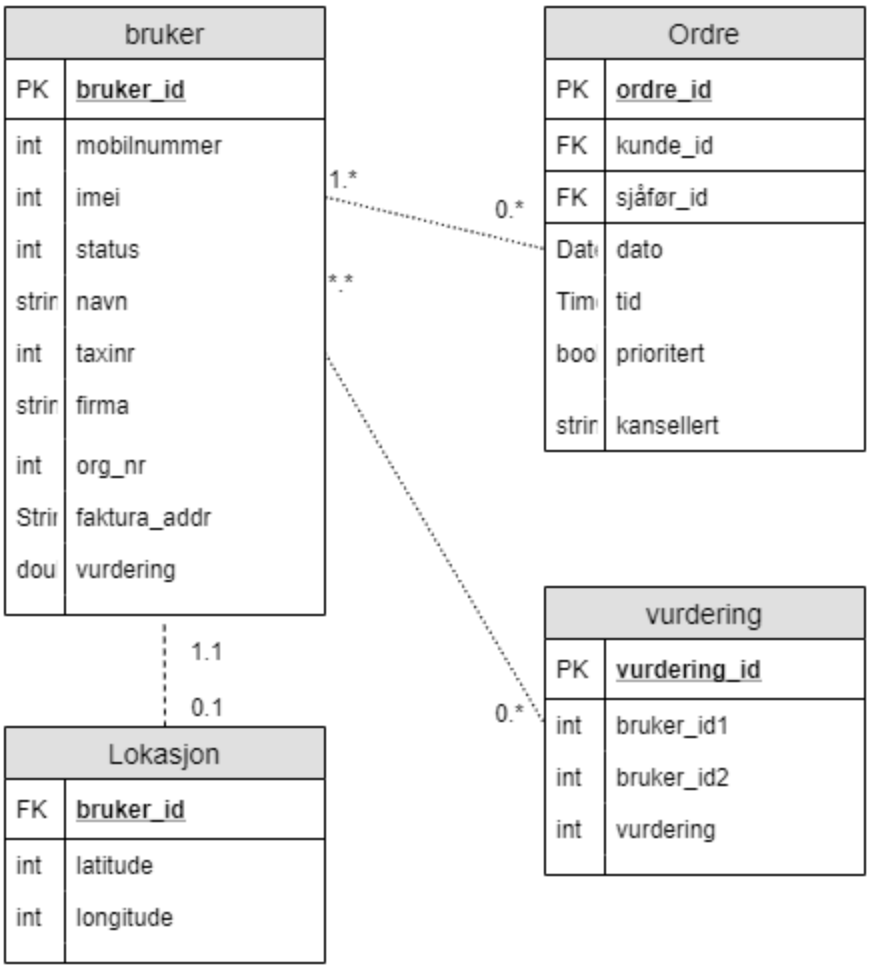
3.1 Klasseoversikt

Slik skal klassene i systemet se ut.



3.2 Databasemodell

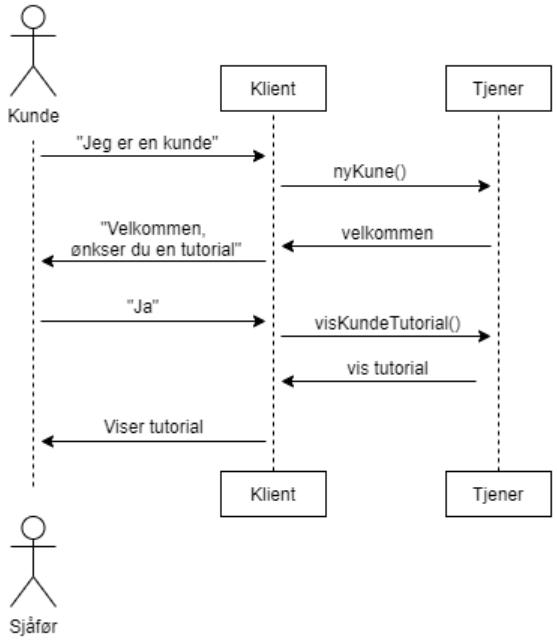
Under finner du en modell som beskriver hvordan databasen til systemet kommer til å se ut.



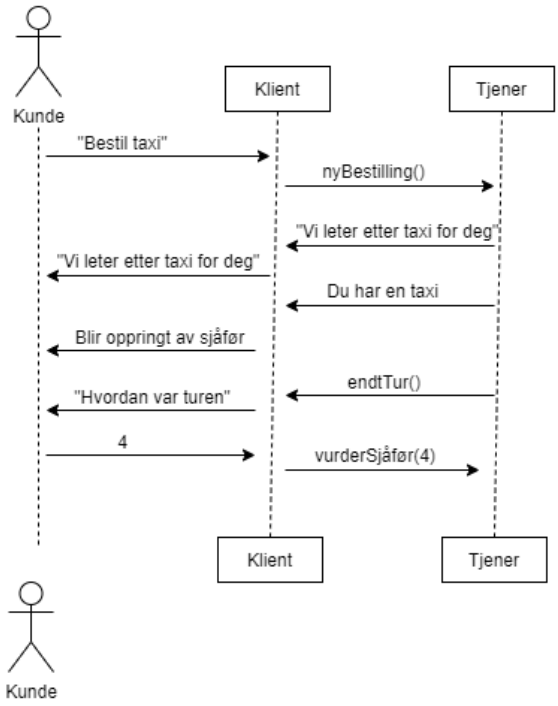
3.3 Sekvensdiagrammer

Her viser dokumentet ett par sekvenser for å beskrive mer detaljert hvordan systemet skal kommunisere med brukerne.

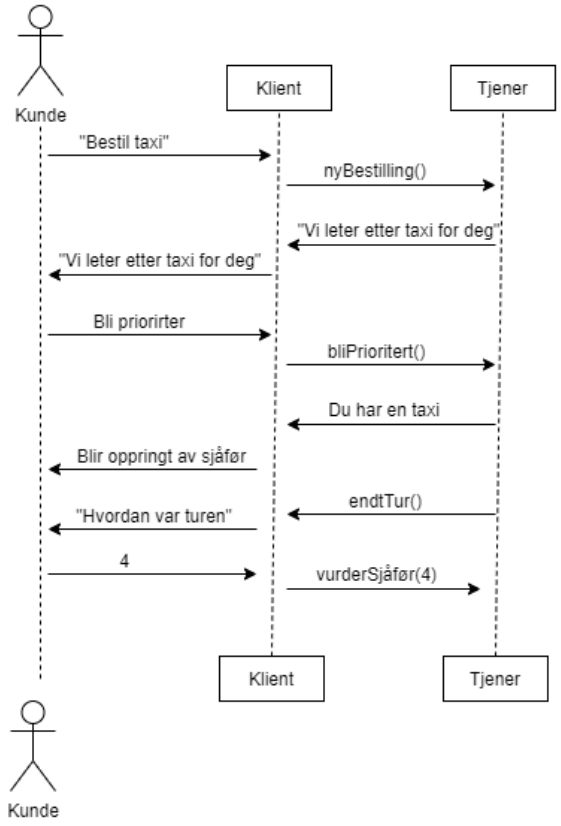
Sekvensdiagram 1 - Registrere seg som kunde



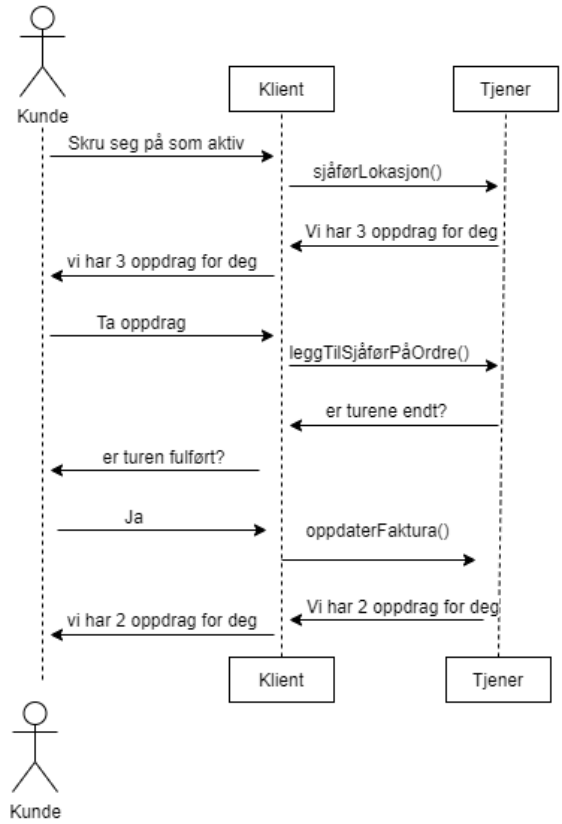
Sekvensdiagram 2 - Bestilling av taxi



Sekvensdiagram 3 - Bli prioritert

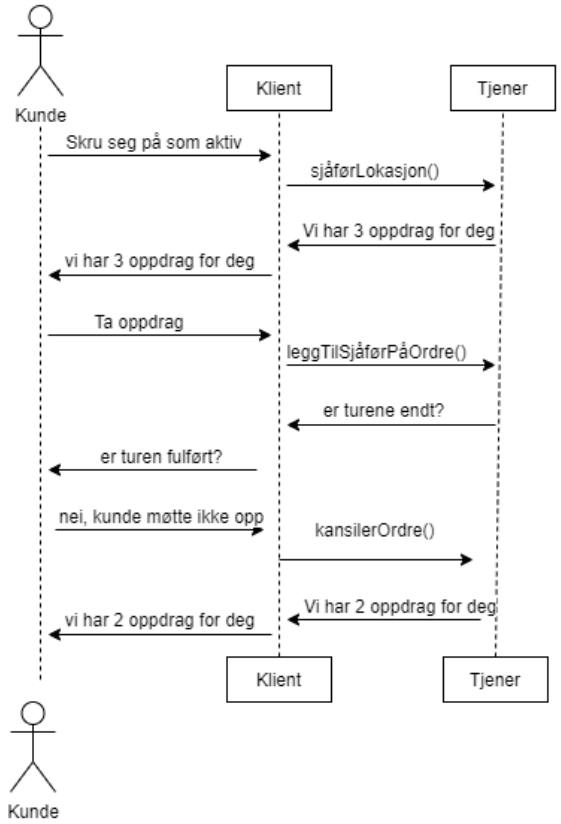


Sekvensdiagram 4 - Sjåfør tar et oppdrag



Sekvensdiagram 5 - Registrere seg som sjåfør

Sekvensdiagram 6 - Kansellere Oppdrag



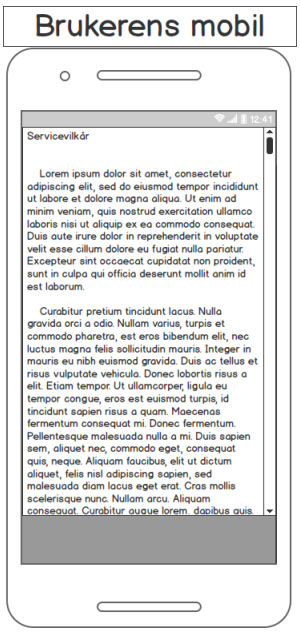
4. Prototyper

Her blir produktet beskrevet ved hjelp av wireframes, på denne måten kan man se hva som skjer etter hverandre eller er avhengig av hverandre. Vi bruker bilder og beskrivelse for å illustrere hvordan man navigerer seg i produktet.

4.1 Wireframes

Det ble brukt Balsamiq Mockups 3 for å lage prototypen og skrive ut bildene av den.

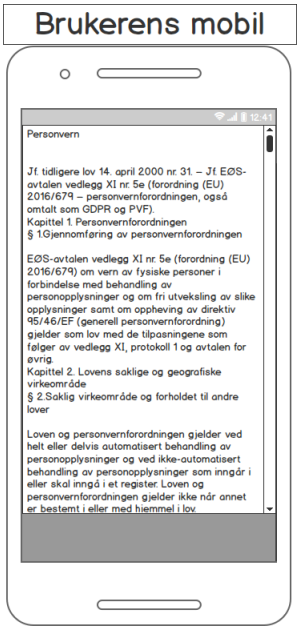
Figur 33 - Brukerens mobil. Ved førstegangskjørings av appen kommer brukeren til registreringsskjerm (Figur 35). Etter det kommer kunden (Figur 48) og sjåføren (Figur 71) til hovedskjermen sin.



Figur 34 – Servicevilkår. Mulig å trykke utenfor teksten for å lukke den.

Figur 36 – Brukeren skriver mobilnummeret sitt og trykker på neste knappen (Figur 38).

Figur 35 - Første gangs startskjerm. Mulig å trykke på knappene: Personvern (Figur 37), Servicevilkår (Figur 34) og Bekreft og Fortsett (Figur 36).

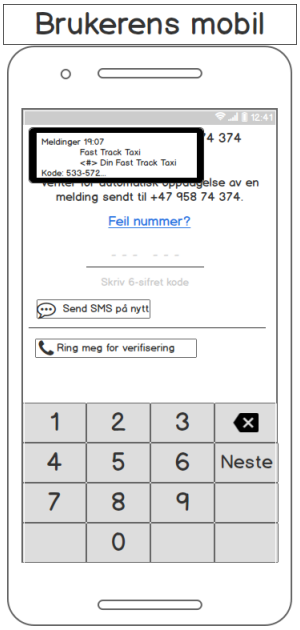


Figur 37 – Personvern. Mulig å trykke utenfor teksten for å lukke den.

Figur 38 - Sjekking om du skrev riktig nummer. Ved å trykke på Endre knappen (Figur 36), OK knappen (Figur 39).

Figur 39 - Venting på melding med kode (Figur 41). Ved å trykke på Feil nummer (Figur 36), Send SMS på nytt blir meldingen med kode sendt på nytt, Ring meg for verifisering blir man oppringt.

Figur 40 - Valg av brukertype. Ved å trykke på Ny kunde? (Figur 42), Ny sjåfør? (Figur 61).



Figur 41 - Meldingen med kode kommer. Automatisk oppdagelse av meldingen, lesing av koden og innloggingen (Figur 43).

Figur 42 - Opplæringsmodus. Kundens velkomstskjerm. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 44).

Figur 43 – Tillatelser. Ved å trykke på Tillatt (Figur 40), Ikke Tillatt (Figur 33).

Figur 44 - Opplæringsmodus. Kundens hovedskjerm. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 45).

Figur 45 - Opplæringsmodus. Kunde. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 46).

Figur 46 - Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 48).

Figur 47 - Opplæringsmodus. Kunde. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 49).

Figur 48 - Kundens hovedskjerm. Ved å trykke på Bestill taxi (Figur 50), Meny (Figur 57).

Figur 49 - Opplæringsmodus. Kunde. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 46).

Figur 50 - Kunden bestilte en taxi og venter på å bli oppringt av en sjåfør. Finnes det en ledig sjåfør som tar oppdraget (Figur 51), ingen ledig taxi (Figur 52). Ved å trykke på Avbestill taxi (Figur 56).

Figur 51 – Fellesskjerm for kunde og sjåfør. Kunden avtaler hentested med sjåføren. Etter samtalen ser kunden skjerm (Figur 53), mens sjåføren ser skjerm (Figur 72).

Figur 52 - Mulighet til å kjøpe seg ut av køen når det er mangel på ledige taxier. Ved å trykke på Kjøp deg ut av køen (Figur 54), avbestill taxi (Figur 56).

Figur 53 - Kundensskjerm etter samtalen med sjåføren. Når kunden og sjåføren befinner seg like hverandre ser kunden skjerm (Figur 55).

Figur 55 - Kunden blir bedt om å gi vurdering av turen. Kunden kan vurdere turen ved å trykke på stjerner. Ved å trykke på stjerne eller Nei, takk (Figur 48).

Figur 57 - Kundens meny. Ved å trykke på Priser (Figur 59), Om Fast Track Taxi appen (Figur 78), Se på guided tour (Figur 42), Avslutt (Figur 33).

Figur 54 - Etter bestilt prioritert taxi. Finnes det en ledig sjåfør som tar oppdraget (Figur 51). Ved å trykke på Avbestill taxi (Figur 56).

Figur 56 - Mulighet for å avbestille taxi. Ved å trykke på Ja (Figur 48), Nei går man tilbake til den opprinnelige skjermen.

Figur 58 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 60).

Figur 59 - Kundens prisoversikt. Ved å trykke utenfor prisoversiktsvinduet går kunden til hovedskjermen (Figur 48).

Figur 60 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 62).

Figur 61 – Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 58).

Figur 62 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (

Figur 63).

Figur 63 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 65).

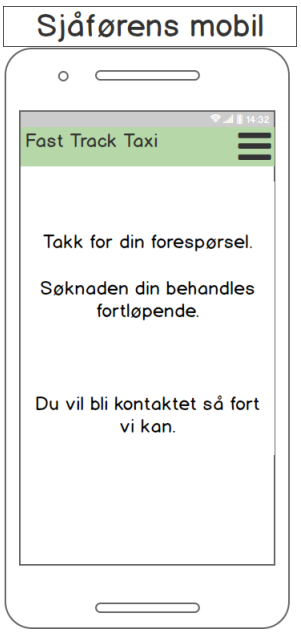
Figur 64 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 66).

Figur 65 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 67).

Figur 66 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 68).

Figur 67 - Sjåfør. Opplæringsmodus. Ved å trykke på luke ikonet går man ut av opplæringsmodusen, hvor som helst på skjermen (Figur 64).

Figur 68 - Sjåførens søknad. Alle feltene må fylles. Ved å trykke på Send inn søknad ser sjåføren en velkomstskjerm (Figur 69).



Figur 69 - Sjåførens skjerm etter registreringen. Er sjåføren godkjent ser han/henne hovedskjermen (Figur 71).

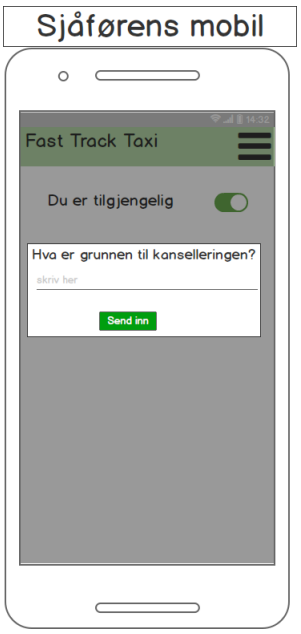
Figur 70 -Sjåførens turtilbudsliste. Ved å trykke på en av Aksepter knappene skal sjåføren godta tilbudet og ringe til kunden med det samme (Figur 51).

Figur 71 - Sjåførens hovedskjerm. Ved å trykke på slå på/slå av knappen logger sjåføren seg og får turtilbud (Figur 73), Meny (Figur 77).

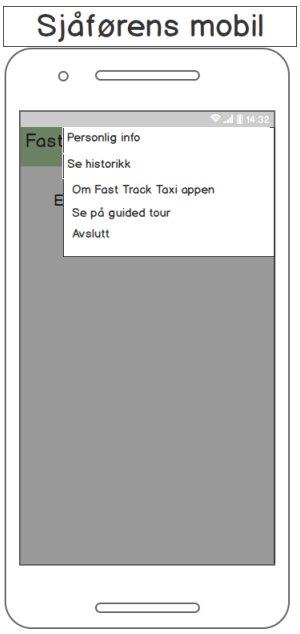
Figur 72 - Sjåførens skjerm under oppdraget. Ved å trykke på Oppdraget utført (Figur 73), Kanseller oppdraget (Figur 74).

Figur 73 - Sjåføren er tilgjengelig og får turtilbud etter hvert (Figur 70).

Figur 74 - Sjåførens valg av kanselleringen. Ved å trykke på Kunden møtte ikke opp (Figur 73), Noe annet (Figur 75).



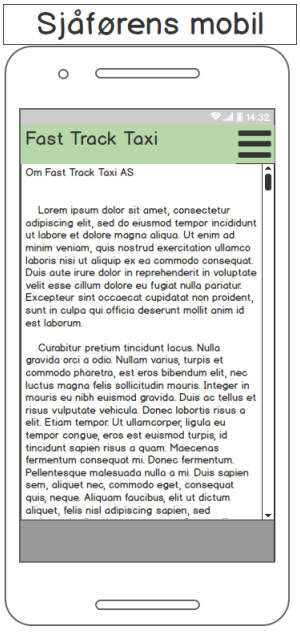
Figur 75 - Sjåføren må skrive grunnen til kanselleringen. Ved å trykke på Send inn (Figur 73).



Figur 77 - Sjåførens meny. Ved å trykke på Personlig info (Figur 79), Se historikk (Figur 76), Om Fast Track Taxi appen (Figur 78), Se på guided tour (Figur 58), Avslutt (Figur 33).

Figur 79 - Sjåførens personlig info. Sjåføren kan endre den personlige infoen sin. Ved å trykke på Avbryt eller Lagre kommer sjåføren tilbake til sin opprinnelige skjerm med eller uten endringen.

Figur 76 - Sjåførens historikk. Her har sjåføren oversikt over de turene den blir/har blitt fakturert for.



Figur 78 - Info om FFT AS. Her kan man lese info om FFT AS firma.

5. Referanser

Balsamiq Mockups 3 hentet fra: https://balsamiq.com/

Fast Track Taxi
Systemdokumentasjon

Versjon 0.9

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
03.03.2020	0.1	Første utkast, generell struktur for dokumentet.	Karol Debik, Kenneth Langdahl Krogstad
06.04.2020	0.2	2. utgave, skrev i kapittel 2, 3 og 4.	Kenneth Langdahl Krogstad
13.05.2020	0.3	3. utgave, skrev i kapittel 3, 5 og 8.	Kenneth Langdahl Krogstad
14.05.2020	0.4	4. utgave, skrev i kapittel 4 og 6.	Kenneth Langdahl Krogstad
26.05.2020	0.5	5. utgave, skrev i kapittel 6 og 8.	Kenneth Langdahl Krogstad
27.05.2020	0.6	6. utgave, skrev om kapittel 1, 2, 3 og 4.	Karol Debik
28.05.2020	0.7	7. utgave, skrev i kapittel 3.	Karol Debik
29.05.2020	0.8	8.utgave, skrev i kapittel 2 og 3.	Karol Debik
30.05.2020		Skrev i kapittel 3 og 4	Karol Debik
01.06.2020	0.9	9.utgave, skrev overalt	Karol Debik

Innholdsfortegnelse

Figurtabell

1. Introduksjon

Dette dokumentet er laget i forbindelse med Bacheloroppgave og beskriver systemet Fast Track Taxi **Invalid source specified.**. Fast Track Taxi er det en app som kommuniserer seg over internett med en server og en database for å oppnå en enklere løsning for bestilling av drosje for drosjemarkedet. Her skal hele systemet beskrives slik at personer som ønsker å vite hvordan systemet fungerer, hvordan er det bygget opp og hvilke løsninger som er brukt for å oppnå denne løsningen, kan lese seg opp. Dette dokumentet inneholder detaljert beskrivelsen av arkitekturen, strukturen, databasemodellen, server-tjenester, sikkerheten til systemet, samt hvordan kan systemet installeres og kjøres. Det er også beskrevet dokumentasjon av kildekoden og kontinuerlig integrasjon og testing.

2. Arkitektur

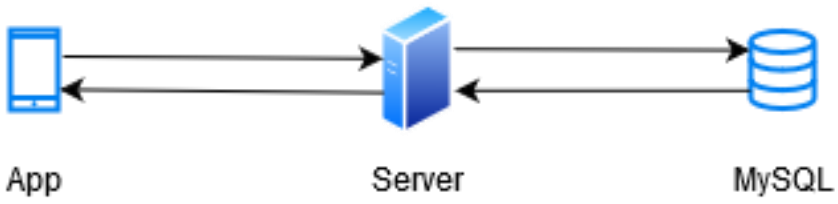
Vi har tre nivåer i vårt system, først har vi en app som fungere på både Android **Invalid source specified.** og iOS **Invalid source specified.**. Appen er skrevet med rammeverket for mobile applikasjoner React Native [16]. Dette rammeverket er basert på React **Invalid source specified.** og lar deg skrive i JavaScript **Invalid source specified.** for å oppnå kildekode for blant annet Android og iOS.

Vi så har en webserver som er programmert og kjørt ved hjelp av rammeverket Node.js **Invalid source specified.**.

Deretter har vi en database for å lagre langsiktig informasjon som er nødvendig. Databasen som er brukt er MySQL **Invalid source specified.**. MySQL bruker Structured Query Language (SQL), et spørrespråk som lar oss skrive og kjøre setninger mot en relasjonsdatabase.

Vi har to typer brukere på appen som blir laget. En kunde her av kalt drosjekunde som benytter tjenesten for service, og den andre, her av kalt sjåfør, utfører denne servicen.

For administrative funksjonaliteter benytter vi oss av SQL setninger.



Figur 80. Strukturen til prosjektet

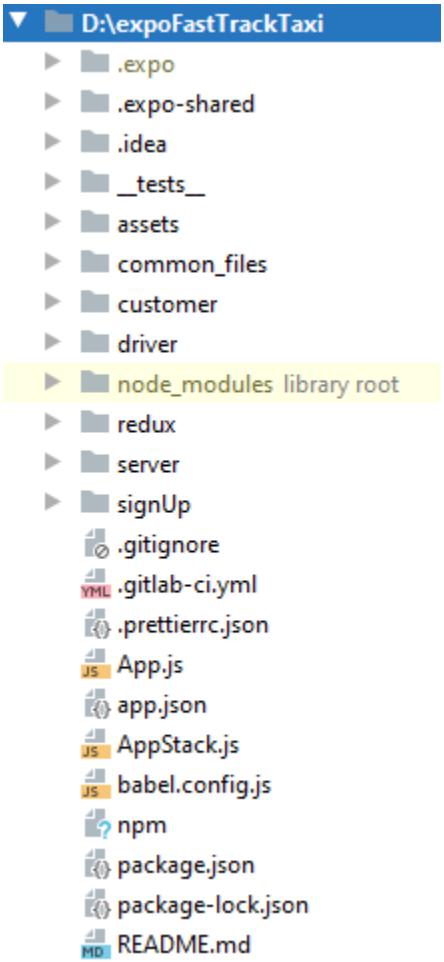
Kommunikasjon i vårt system (**Error! Reference source not found.**Figur 80) er slik at appen kommuniserer seg med serveren, dvs. at alle databasespørringer går via serveren. Serveren sender en forespørsel til databasen og når den får svar så sender den svaret tilbake til appen. På den måten er det ingen direkte kobling mellom appen og databasen.

3. Prosjektstruktur

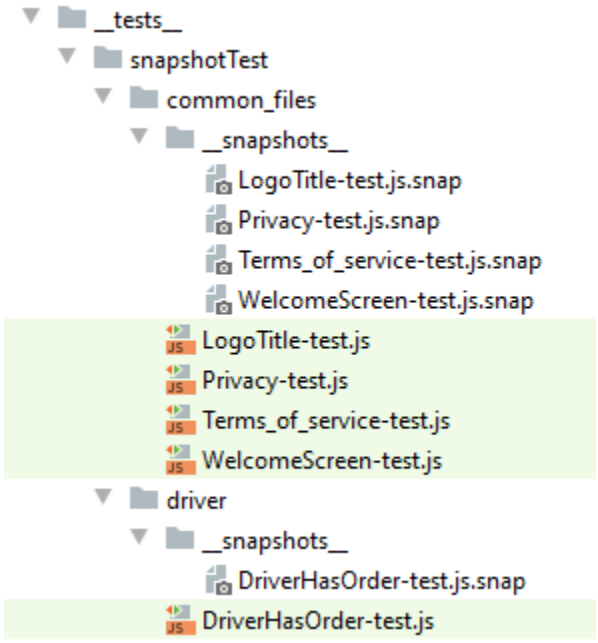
Her blir strukturen av prosjektet detaljert. Det blir splittet opp mellom applikasjonen og serveren.

3.1 Applikasjonen

Strukturen i applikasjonen ble først initiert av en expo init expoFastTrackTaxi kommando **Invalid source specified.**. Denne kommandoen har generert alle nødvendige filer og formert strukturen for prosjektet. Under utviklingsperioden har strukturen endret seg noe, dvs. det ble lagt noen mapper for å holde orden i filene.



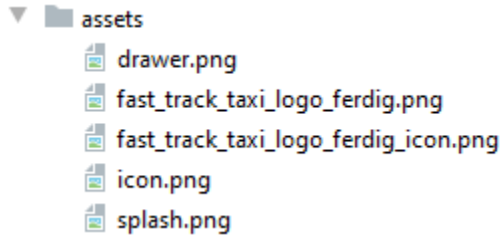
Figur 81. Strukturen til prosjektmappen



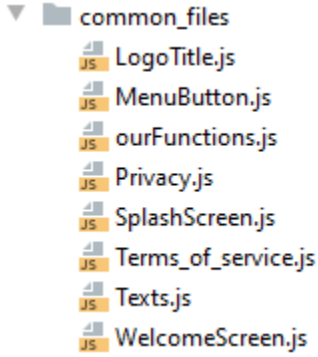
Figur 82. Strukturen til testmappen

Prosjektstrukturen (Figur 2) i applikasjonsdelen ble delt opp i mindre deler ut ifra hvem eller hva gjald funksjonaliteten. I tillegg ble det også lagt til en server mappe for å ha all kode sammen.

Testing av applikasjonen ble gjennomført ved hjelp av *jest* **Invalid source specified.**. I test mappa (Figur 82)ble det kjørt tester på noen GUI i applikasjonen. I de tilsvarende mappene hvor det befinner seg test filene har *jest* laget en __snapshot__ mappe med testresultater. Neste gang testing skal kjøres blir de *.snap* filene brukt som referanse til neste testing.



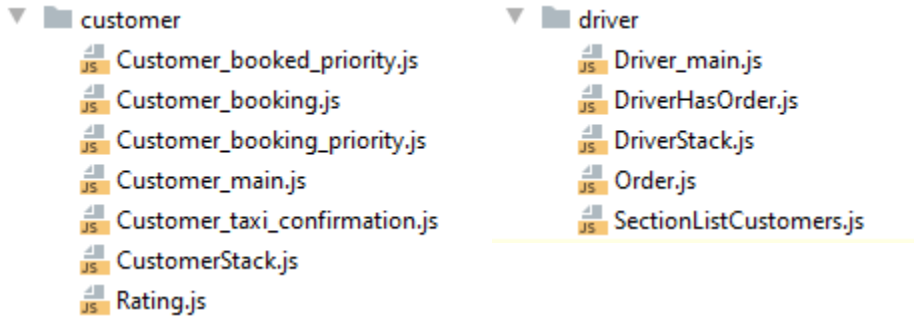
Figur 84. Mappen med ikoner



Figur 83. Mappen med felles filer

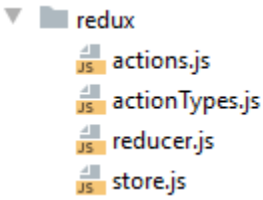
Denne mappen (Figur 84) inneholder bilder/ikoner som er blitt brukt i prosjektet. Her finner vi logo til applikasjonen som ble brukt i splashscreen, samt logo-ikonen som ble brukt i topbaren på alle skjermer i kunde og sjåfør modusen.

Mappen common_files (Figur 4) inneholder filer som er blitt brukt av alle brukere.

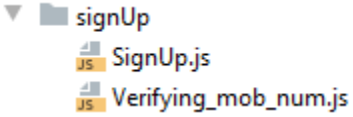


Figur 85. Mappene med filer brukt i kunden- og sjåførprofilen

Mappene (Figur 6) inneholder filer som genererer ulike skjermer for kunden/sjåføren.



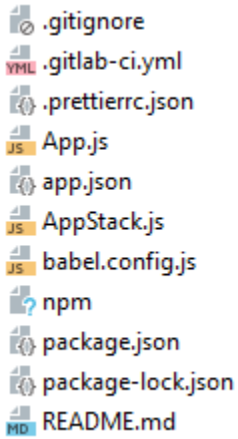
Figur 86. Redux mappe



Figur 87. Registrering mappe

Filer i redux mappen (Figur 7) hjelper med å lagre nødvendige variabler på mobilen. Vi brukte Redux biblioteket **Invalid source specified.** for at FFT appen skal ha en hukommelse. Den trengs til å huske state i appen, blant annet hvem brukeren er (kunde eller sjåfør).

Registrering mappen (Figur 87) inneholder filene som er ansvarlige for å vise de første skjermene etter at brukeren har lastet ned appen og kjørt den for første gang.



Figur 88. Frittliggende filer

Noen av disse filene (Figur 88) ble generert automatisk av expo. De er på en måte hjernen av applikasjonen fordi den første filen som er lest når appen starter er App.js. Her ser vi også andre filer med konfigurasjon som blant annet .gitignore, gitlab-ci.yml eller package,json.

3.2 Serveren

```
drwxrwxr-x   4 administrator administrator  4096 mai   28 12:36 .
-rw-rw-r--   1 administrator administrator 12305 mai   28 12:36 routers.js
drwxrwxr-x 753 administrator administrator 28672 mai   28 10:12 node_modules
drwxr-xr-x  24 administrator administrator  4096 mai   27 21:37 ..
-rw-rw-r--   1 administrator administrator  1343 mai   20 14:58 tools.js
-rw-rw-r--   1 administrator administrator   643 mai   20 14:44 server.js
-rw-rw-r--   1 administrator administrator  1199 mai    6 11:50 package.json
drwxr-xr-x   3 root          root           4096 mai    3 22:34 .well-known
```

Figur 89. Strukturmappe med serverfilene

```
root@ftt:/etc/letsencrypt/live/ftt.idi.ntnu.no# ls -alt
total 12
drwxr-xr-x 2 root root 4096 mai    6 11:29 .
drwx------ 3 root root 4096 mai    6 11:29 ..
lrwxrwxrwx 1 root root   39 mai    6 11:29 cert.pem -> ../../archive/ftt.idi.ntnu.no/cert1.pem
lrwxrwxrwx 1 root root   40 mai    6 11:29 chain.pem -> ../../archive/ftt.idi.ntnu.no/chain1.pem
lrwxrwxrwx 1 root root   44 mai    6 11:29 fullchain.pem -> ../../archive/ftt.idi.ntnu.no/fullchain1.pem
lrwxrwxrwx 1 root root   42 mai    6 11:29 privkey.pem -> ../../archive/ftt.idi.ntnu.no/privkey1.pem
-rw-r--r-- 1 root root  692 mai    6 11:29 README
```

Figur 90. Mappen med sertifikat filer

I hoved mappen (Figur 89) har vi server.js som setter opp server detaljer som hvilken port og adresse serveren skal kjøres på, samt hvor og hvilket sertifikat som blir bruktr spesifisert her.

I routers.js ligger oppsettet av endepunkter og oppkobling mot databasen.

tools.js har vi funksjoner har laget selv, disse hovedsakelig tar seg av generering og verifisering av token.

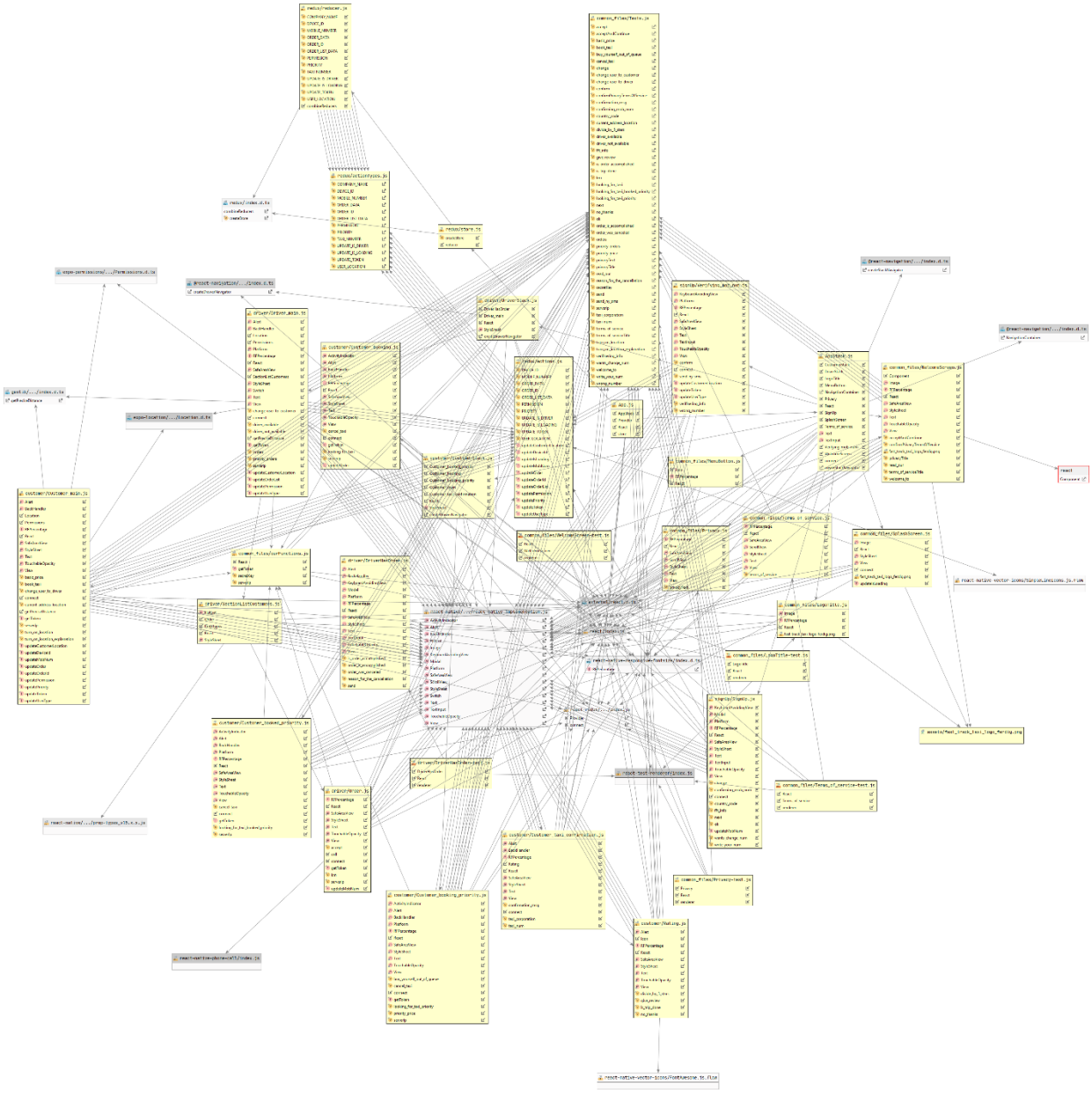
Mappen .well-known ble brukt for å lage sertifikat fra Let's Encrypt **Invalid source specified.**.

I mappen /etc/letsencrypt/live/ftt.idi.ntnu.no (Figur 90) har vi filene "fullchain.pem" og "privkey.pem". Disse inneholder nøkkelen og sertifikatet vi bruker for å bruke HTTPS.

I mappen node_modules har vi alle eksterne biblioteker som er nødvendige for å ha den funksjonaliteten vi trengte for serveren.

4. Klassediagram

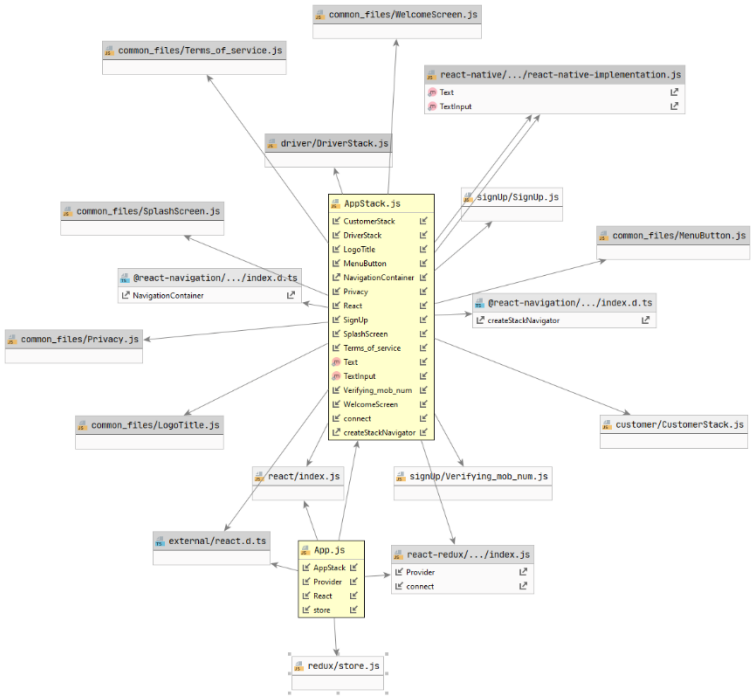
4.1 Applikasjonen



Figur 91. Klassediagram til applikasjonen

Strukturen til applikasjonen (Figur 91) er stort og ganske komplisert. Det er mange sammenhenger mellom forskjellige klasser. I klassediagrammet til applikasjonen ser vi i tillegg til de pakkene vi laget at det også er et par andre eksterne pakker. Siden klassediagrammet er så stort, har vi delt det opp i mindre deler.

4.1.1 App og AppStack



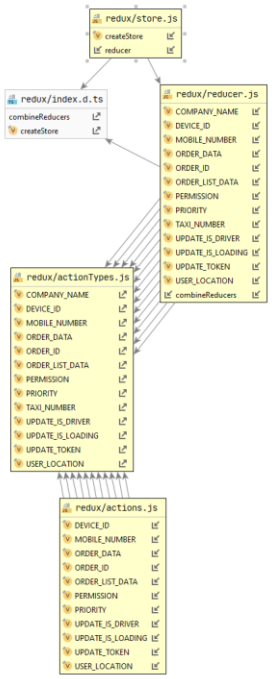
Figur 92. Klassediagram til App og AppStack

Figur 92 viser de klassene som blir kjørt ved første gang appen kjøres. Klassene App, AppStack og SplashScreen blir kjørt hver gang appen kjøres, mens andre klasser som SignUp, WelcomeScreen, Privacy, Terms_of_service og Verifying_mob_num, er klasser som blir kjørt bare en gang når brukeren skal utføre registrering. Ved vellykket registrering blir brukeren send til Customer_main klassen. Alle andre klasser er enten eksterne klasser eller våre egne klasser som er brukt til generering av skjermene.

Den første klassen som blir kjørt aller først av appen er App klassen. Den er roten til applikasjonen. Her er redux store implementert ved hjelp av en ekstern Provider klasse som gjør det mulig at store blir tilgjengelig i hele appens treet. AppStack klassen er pakket inn i Provider klassen.

AppStack klassen er selve hjernen til applikasjonen. I staken er flyten mellom forskjellige skjermer implementert. Flyten er styrt ved hjelp av appens hukommelsen, nemlig redux store.

4.1.2 Redux



Figur 93. Klassediagram til redux filene

Redux mappen inneholder klasser actionTypes, actions, reducer og store (Figur 93). Klassen index.d.ts er en ekstern klasse brukt av store og reducer klassene.

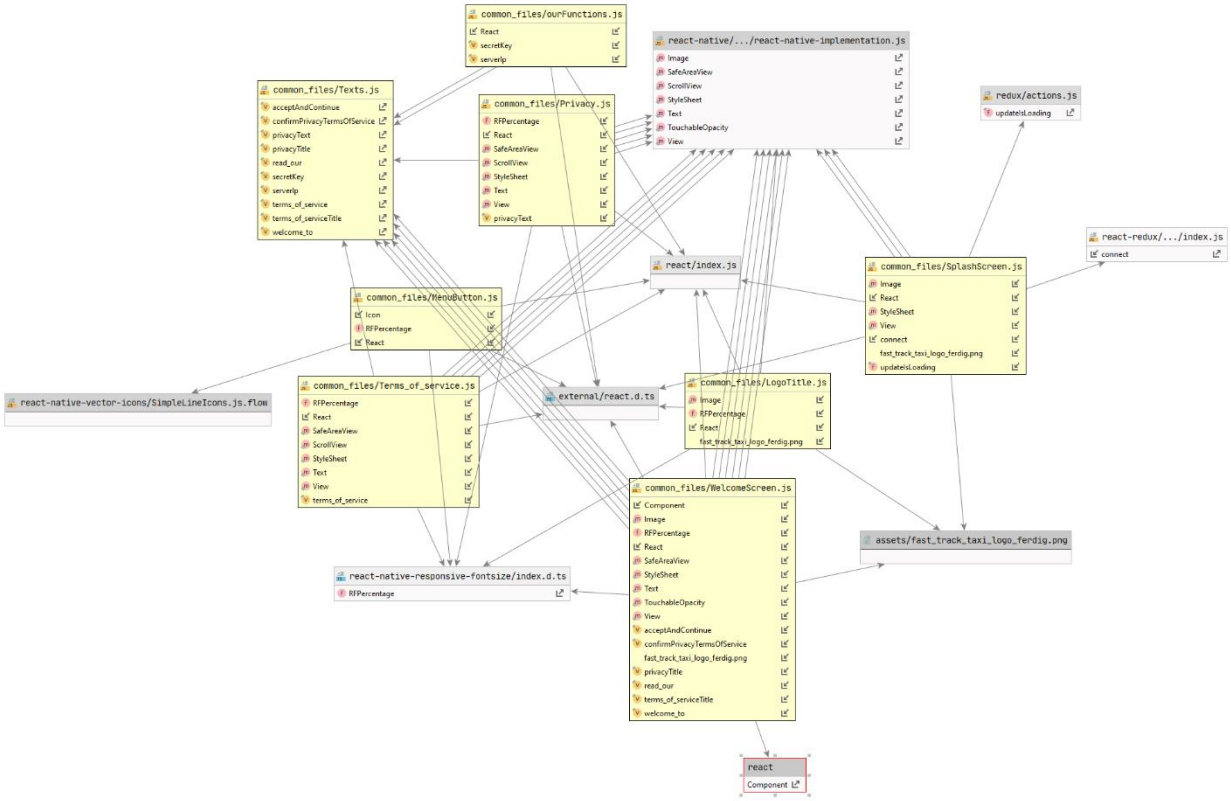
Klassen store har som sin oppgave å lage et nytt lager (store). Det muliggjør lagring av nødvendige data i appen.

ActionTypes klassen beskriver typer til aksjoner som er mulige og tilgjengelige i store. Den er koblingen mellom actions og reducer klassene.

I actions klassen er det implementert flere action creators som er ikke noe annet enn funksjoner som skaper aksjoner. Aksjoner sender nyttelasten fra applikasjonen til lageret.

I reducer klassen er det implementert mange reducere som er samlet sammen i en stor reducer ved hjelp av combineReducers metoden. Hver reducer har som sin oppgave å endre state i forhold til nyttelasten fra aksjoner.

4.1.3 Common_files

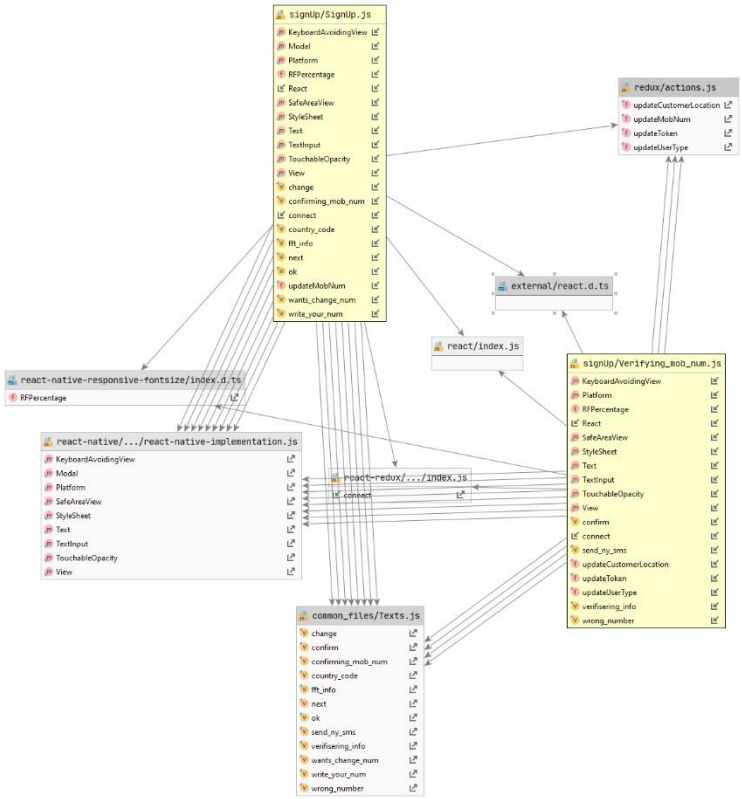


Figur 94. Klassediagram til filene i common_files mappen

I mappen common_files (Figur 94) finnes det klasser som er brukt av både kunde og sjåføren. Igjen er alle gule klasser og klassen actions laget av oss. Alle andre klasser er eksterne klasser.

Når appen kjøres for aller første gang så møter brukeren splash skjermen generert av SplashScreen klassen, så sendes den til registreringsskjerm. Sist nevnte skjermen er generert av WelcomeScreen klasse. Denne klassen henter mange tekster fra Texts klassen samt et logo-bilde fra assets mappen. Fra denne klassen har vi direkte tilgang til Personvern og Servicevilkår skjermene som er henholdsvis generert av Privacy og Terms_of_service klassene.

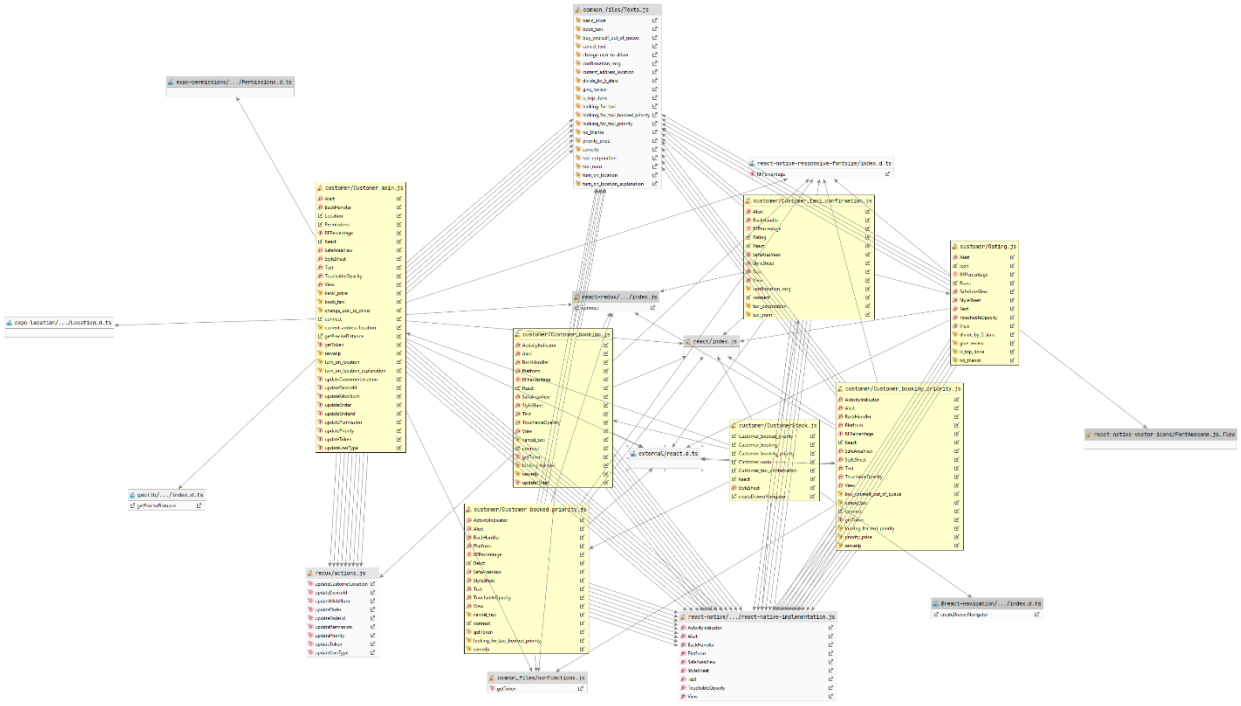
4.1.4 SignUp



Figur 95. Klassediagram til filene i SignUp mappen

SignUp mappen inneholder klassene SignUp og Verifying_mob_num som er ansvarlige for å ta vare på input data fra brukeren og utføre reigstrering mot databasen via serveren. I Figur 95 ser vi at de to klassene har kobling til Texts klassen for å hente noen tekster, samt actions klassen for å lagre nødvendige bruker data.

4.1.5 Customer



Figur 96. Klassediagram til filene i customer mappen

I customer mappen (Figur 96) ser vi flere klasser som er enten laget bare for kunden eller er det felles klasser brukt av både kunde og sjåføren. I tillegg ser vi noen eksterne klasser.

Alle de gul fargede klasser er klassene vi lagde, og de gjelder bare kunden. Klassene Texts.js, react_redux/…/index.js, actions.js og ourFunctions.js er andre klasser vi laget som er felles for brukere av appen. Alle andre klasser som er visst på klassediagrammet er eksterne klasser.

Klassen CustomerStack holder orden for navigering mellom alle kundens skjermer.

Ut ifra Figur 96 ser vi at det er flere klasser som er ansvarlige for å generere den riktige skjermen. Alt ut ifra hvilke beslutninger kunder har utført.

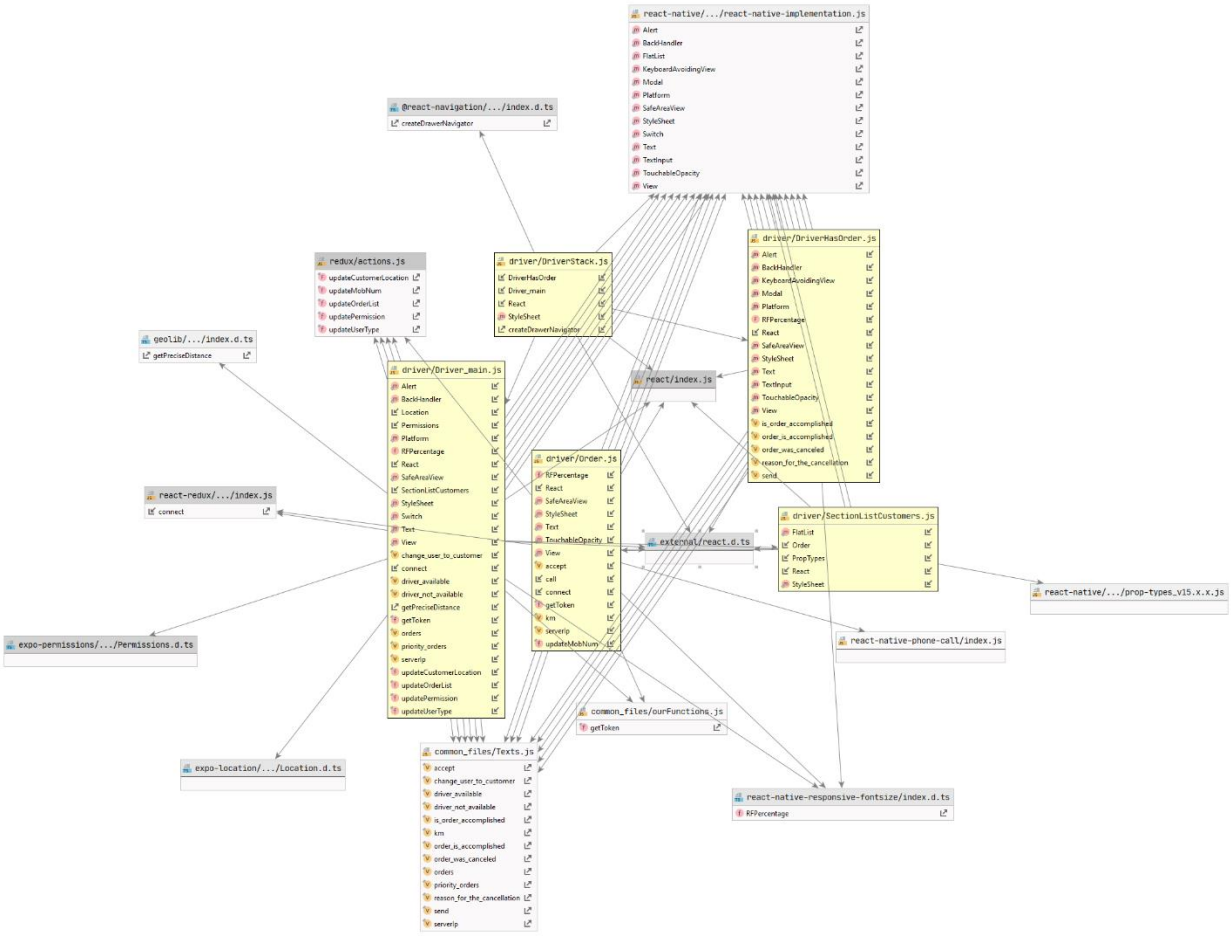
Klassen Customer_main genererer den første skjermen kunder møter når den er ferdig registrert i appen og hver gang den slår på appen. I denne klassen er det utført en spørring mot databasen via serveren for å registrere ønske om bestilling av en drosje.

Customer_main klassen har også direkte kobling til expo-location og expo-permission klassene for å utføre spørringen om tillatelse til lokasjonen og til å få kundens nåværende lokasjon.

I andre klasser som Customer_booking, Customer_booking_priority, Customer_booked_priority og Customer_taxi_confirmation er det også spørringer mot databasen som igjen går via serveren. De spørringene er enten for å oppdatere bestillingen, sende en bekreftelse for at en drosje er på vei eller sende en vurdering av kjøreturen.

I Figur 96 ser vi også bruk av andre våre klasser som er utenfor customer mappen. De er brukt til enten lagring av kundens data på mobilen eller for å vise tekster i de genererte skjermene.

4.1.6 Driver



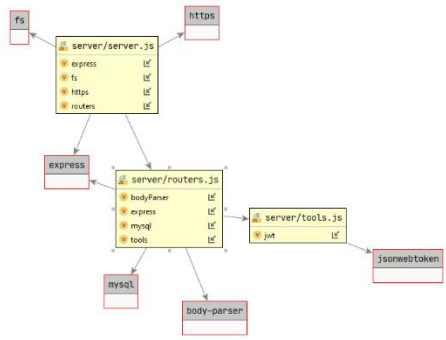
Figur 97. Klassediagram til filene i driver mappen

Driver mappen (Figur 97) har en DriverStack klasse som holder orden for navigeringen mellom sjåførens skjermer, samt klassen Driver_main som genererer start skjermen for sjåføren og DriverHasOrder klasse som genererer skjermen når sjåføren tok ordren.

I Dirver_main klassen har vi tilsvarende kobling til expo-permissions og expo-location klasser som i Customer_main klassen, samt kobling til andre våre klasser som Texts og actions. I denne klassen er det implementert henting av ordre listen fra serveren, hvor hver ordre er pakket inn i en SectionList.

Både i Customer (Figur 96) og Driver (Figur 97) klassene er det brukt klassen ourFunctions som har en metode for å hente token fra serveren.

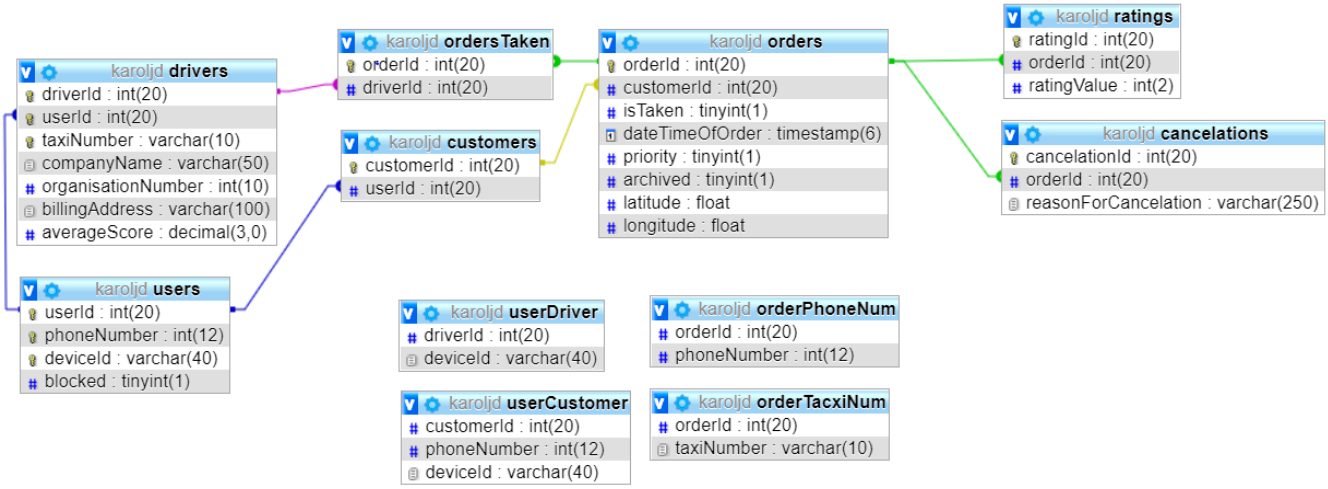
4.2 Server



Figur 98. Klassediagram til filene i server mappen

I server mappen (Figur 98) ser vi server, routers og tools klassene. Sammen danner de klassene serverern trenger for å motta meldinger fra appen, slik at de blir behandlet og utfører spørringen mot databasen for å returene ett svar på meldingen.

5. Databasemodell



Figur 99. Databasemodellen

I Figur 99 ser vi oversikt over databasemodellen. Her ser vi tabellene og views som ble brukt i prosjektet.

Dersom brukeren er en kunde, blir kunden lagt inn i “users” og “customers” tabellen med telefon nummer og smart telefonens unike id, da den verifiserer en engangs kode som skal bli sendt til telefonen. Hvis brukeren er en sjåfør, skal de først bli lagt inn i “drivers” og telefon nummer bli lagt inn i “users”. På denne måten blir sjåføren identifisert som en sjåfør ved verifisering i appen.

Deretter har vi ordre, når en kunde bestiller drosje blir kundens id og lokasjon lagt inn. En kunde kan bestille prioritert drosjetur, og da endre verdien på “priority”. Da sjåføren ønsker å utføre turen, blir “isTaken” endret til sann, samt det blir lagt inn sjåfør id og ordre id i en tabell kalt “ordersTaken”. Da turen er ferdig skal ordre bli oppdatert slik at “archived” er sann og kunde har mulighet å gi en vurdering mellom 1-5. Dersom det oppstår noen grunn for å kansellere drosjeturen blir ordrenummer og grunn lagt inn i “cancellations” og “archived” endret til sann.

Vi har også ett par “views” for å ha god struktur og begrenset muligheter for feil. Disse er “userDriver”, “userCustomer”, “orderPhoneNum” og “orderTaxiNum”.

6. Server-tjenester

Node.js er ett RESTfull rammeverk, vi har derfor disse endepunktene i systemet.

6.1 Felles endepunkter

De endepunktene som begge typer brukere benytter.

```
router.post('/token', function (req, res) {...})
```

Denne tar inn ett «hemmelig» passord, dersom dette er korrekt blir en gyldig TOKEN returnert. Alle andre endepunkter er nødt til å inneholde denne token for å utføre ønsket funksjonalitet. Verifisering av token blir gjort i en egen funksjon, som ligger i filen tools.js.

6.2 Sjåfør endepunkter

```
router.get('/getorders', function(req, res) {...}); // A driver asks for all the orders
router.post('/takeorder', function(req, res) {...}); // A driver takes an order
router.put('/endTrip', function(req, res) {...}); // The driver ends the trip
```

getOrders: Denne tar inn token, returnerer alle ordre i databasen som ikke er tatt av andre sjåfører eller arkivert.

takeOrder: Tar inn token, ordreId og deviceId, endrer verdien «isTaken» hos den rette ordre, samt legge inn sjåførens id og ordre id inn i «ordersTaken». Returnerer så telefon nummer hos kunden slik at sjåføren kan kontakte kunden.

endTrip: Tar inn token og ordre id. Endrer verdien «archived» i databasen til «1» (sann), returnerer sann for å bekrefte at dette ble gjort.

6.3 Kunde endepunkter

```
router.post('/makeorder', function(req, res) {...}); // A customer makes an order
router.put('/makeprio', function(req, res) {...}); // Customer makes the order into a priority
router.get('/getOrderTaxiNum', function(req, res) {...}); // Customer get the taxi number for the driver on the order
router.post('/cancelOrder', function(req, res) {...}); // Customer wanted to cancel before a driver called
```

makeOrder: Tar inn token, deviceId, latitude, longitude. godkjenner token, oppretter ordre i databasen og returnerer ordre id.

makePrio: Tar inn token, ordre id. Godkjenner token, endrer verdien på rett ordre til prioritert, sender tilbake «sann»

getOrderTaxiNum: Tar inn token og ordre id. Godkjenner token, returnerer drosjenummer og bedrift navn dersom ordre finnes i ordersTaken.

cancelOrder: Tar inn token og ordre id. Godkjenner token, endrer verdien på ordre til archived, legger inn ordre id og grunn i «kanselleringer» tabellen i databasen.

7. Sikkerhet

Ettersom vi bruker SQL løsning må vi beskytte oss for SQL-injeksjon. Ved Node.js har vi noe kalt «Named Placeholders». På denne måten blir brukerens data innkapslet og kan dermed ikke bli brukt for å misbruke systemet.

Serveren har sertifikat av Let's Encrypt, ett gratis system som gir kortvarige sertifikater. Den bruker kun HTTPS for å ha en kryptert kobling mellom tjener og klientene.

Selv om vi beskyttet oss for SQL-injection er systemet vårt utviklet slik at det er ingen mulighet for brukeren å utføre en SQL-injection. I selve appen er det ikke mulig å skrive en spørring mot databasen. Eneste input tekst fra brukeren er hos sjåfør delen hvor han/hun kan skrive årsaken til kanselleringen av oppdraget, men den får ingen tilbakemelding fra serveren. Andre spørringer som er utført mot databasen er utført etter at brukeren har trykket på en knapp, dvs. at det er ikke mulig å misbruke systemet på den måten.

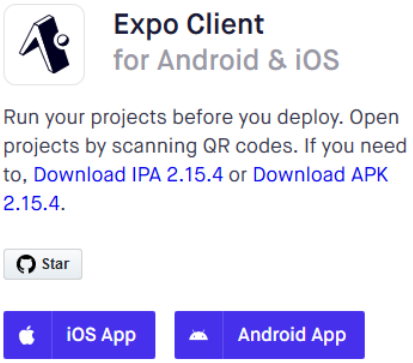
Med begrenset tid fikk vi ikke tid til å implementere gode sikkerhets tiltak, men vi bruker en Token generator kalt JSON Web Tokens (JWT). Ettersom vi ikke har hatt nokk tid til å implementer den sikkerheten som er ønsket har vi en statisk hjemlighet som ligger på serveren og appen. På denne måten er det kun de med denne hjemligheten som kan kommunisere med serveren.

8. Installasjon og kjøring

Alle avhengigheter er listet i package.json. Under kommer informasjon om hvordan installasjon og kjøring blir gjort for server og applikasjonen. Det finnes to package.json filer, en for applikasjonen og en for serveren.

For å kjøre appen på mobilen må det i tillegg lastes ned en Expo Client app (Figur 100) som kjører selve appen.

Her er Android versjon av expo client og iOS versjon av expo client.



Figur 100. Logo til Expo Client app

For å få full nytte av applikasjonen bør den kjøres på mer enn en mobil, slik at det blir minst en mobil for kunden og en mobil for sjåføren.

8.1 Applikasjon

Siden vi underskrev en avtale hvor oppdragsgiveren har alle rettighetene til kildekoden er den ikke tilgjengelig for andre. For å få tilgang til Gitlab **Invalid source specified.** repoet send en epost til Karol Debik eller Kenneth Langdahl Krogstad.

For å installere applikasjonen må den først «lastes ned» fra Fast Track Taxi Gitlab repo.

Videre er det en sekvens av kommandoer som må skrives i terminalen/kommando vinduet i roten av prosjektet fordi der befinner seg package.json filen (Figur 101) som inneholder listen over alle bibliotekene brukt i prosjektet.

```
{
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web",
    "eject": "expo eject",
    "test": "jest"
  },
  "dependencies": {
    "@react-native-community/geolocation": "^2.0.2",
    "@react-native-community/masked-view": "0.1.6",
    "@react-navigation/bottom-tabs": "^5.4.2",
    "@react-navigation/drawer": "^5.7.2",
    "@react-navigation/native": "^5.3.0",
    "@react-navigation/stack": "^5.3.2",
    "core-js": "^3.6.5",
    "expo": "^37.0.9",
    "expo-cli": "^3.20.9",
    "expo-constants": "~9.0.0",
    "expo-location": "~8.1.0",
    "expo-permissions": "~8.1.0",
    "geolib": "^3.2.1",
    "prop-types": "^15.7.2",
    "react": "~16.9.0",
    "react-dom": "~16.9.0",
    "react-native": "https://github.com/expo/react-native/archive/sdk-37.0.1.tar.gz",
    "react-native-device-info": "^5.5.7",
    "react-native-gesture-handler": "~1.6.0",
    "react-native-modalbox": "^2.0.0",
    "react-native-phone-call": "^1.0.9",
    "react-native-reanimated": "~1.7.0",
    "react-native-responsive-fontsize": "^0.4.3",
    "react-native-safe-area-context": "^0.7.3",
    "react-native-screens": "~2.2.0",
    "react-native-vector-icons": "^6.6.0",
    "react-native-web": "~0.11.7",
    "react-navigation": "^4.3.9",
    "react-redux": "^7.2.0",
    "react-test-renderer": "^16.13.1",
    "redux": "^4.0.5"
  },
  "devDependencies": {
    "@babel/core": "^7.9.6",
    "babel-preset-expo": "~8.1.0",
    "jest": "^26.0.1",
    "jsdoc": "^3.6.4",
    "prettier": "^2.0.5"
  },
  "jest": {
    "preset": "react-native"
  },
  "private": true
}
```


Figur 101. package.json fil

npm install

Neste steg er å kjøre en npm install kommando. Siden prosjektet ble ikke publisert blir også devDependencies installert fra package.json filen.

expo start

Siden appen ble utviklet ved hjelp av expo rammeverket for å starte den må det kjøres expo start.

Etter at expo start kommandoen har startet skal det dukke opp en QR kode i terminalen/kommando vinduet, samt skal det åpnes et vindu i standard nettleseren din hvor det også skal en QR kode vises. Da er det å skanne denne QR koden ved hjelp av Expo Client appen. Etter at JavaScript pakke ble bygd opp og lastet ned til mobilen skal Fast Track Taxi appen startes.

8.2 Server

For å installere server trengs filene som vist i prosjektstruktur 2.1. Dersom man er på *ubuntu* kjører man «sudo apt install nodejs». Da man har dette på plass kan man laste ned de nødvendige avhengigheter, disse er listet i package.json.

```
"express": "^4.17.1",
"jsonwebtoken": "^8.5.1",
"mysql": "^2.18.1",
"nodemon": "^2.0.3",
```

Express er typen node server vi benytter

Jsonwebtoken er API som gir oss mulighet til å lage token og verifisere disse.

MySQL brukes for oppkobling mot databasen vi benytter.

Nodemon er brukt for hurtig endring og start av server, denne restarter serveren dersom det har skjedd endringer på koden.

I server.js har vi disse avhengighetene:

```
var express = require('express');
var fs = require('fs');
//var http = require('http');
var https = require('https');
var routers = require('./routers');
```

Express: Dette er biblioteket som fir oss muligheten til å opprette en webserver og kjøring av den.

fs er API som gir muligheten for å lese forskjellige fil typer.

http brukes for testing av utvikling på lokal server, benyttet før vi fikk på plass sertifikat

https er det som lar oss opprette en server oppkobling med ett sertifikat.

Routers: routers.js inneholder de endepunktene som blir brukt.

I router.js har vi disse avhengighetene

```
var express = require('express');
var tools = require('./tools');
var mysql = require('mysql');
```

Exprees: Dette er det som gir oss endepunkt funskjone i server

Tools: Her hentes funksjonen for å verifisere en token.

Mysql: Gir oss funksjonaliteten for å koble opp og snakke med databasen vi bruker.

I tools.js har vi disse avhengighetene:

```
var jwt ='require('jsonwebtoken');
```

Jwt: Dette er biblioteket som gir oss funksjonaliteten for å lage og verifisere en token.

For å installere alle disse avhengighetene fra package.json brukes npm (node package manager), dersom man ikke har dette installert kjøres kommandoen «sudo apt install npm». Når dette er gjort kan man kjøre «npm install» da blir alle de nødvendige bibliotekene installert. Man kan så starte serveren med kommandoen «sudo node server.js». Det blir så skrevet i kommandolinjen at serveren har startet på port 443.

9. Dokumentasjon av kildekode

Dokumentasjon av kildekode skjer i selve koden, *git* historikk og kommentarer finns på *GitLab* hvor vi hadde vår repository.

Dokumentasjon av kildekode ble gjort ved hjelp av JSDoc markup language, som videre ble skrevet ut og lagt ved som vedlegg.

10. Kontinuerlig integrasjon og testing

Ettersom appen er skrevet med React Native har vi satt opp noen tester ved hjelp av et Jest rammeverk. Videre ble det satt opp gitlab-ci.yml fil for å kjøre koden i en Gitlab Runner som skal sjekke om alt er satt riktig og om det er mulig å kjøre koden og tilhørende jest tester.

Vi har også testet appen fysisk på Android og iOS telefoner for å blant annet avdekke mulige avvik i GUI.

10.1 Applikasjonen

Det ble skrevet bare noen få tester på appens generering av skjermer (Figur 82).

De testene som ble skrevet er såkalte snapshot tester for LogoTitle, Privacy, Terms_of_service, WelcomeScreen og DriverHasOrder klassene.

10.2 Server

Serveren brukte applikasjonen for å teste endepunkter og funksjoner., Ut av de endepunktene og funksjonene som ble laget var det 5 av 7 endepunkter som ble testet. For å teste disse ble funskjonene i tools.js testet ved hver test.. Oppkobling mot databasen blir brukt i alle nesten alle endepunkter og derfor testet.

11. Referanser

[1] B. &. K. N. Forssell, «Store Norske Leksikon,» 2020. [Internett]. Available: https://snl.no/GPS. [Funnet 17 April 2020].

[2] T. Logemann, «intersoft consulting,» 2018. [Internett]. Available: https://gdpr-info.eu/. [Funnet 20 April 2020].

[3] «NRK - Spalte med siste info på toppen av siden,» [Internett]. Available: https://www.nrk.no/. [Funnet 05 06 2020].

[4] «Drosjetransport - årlig - SSB,» [Internett]. Available: https://www.ssb.no/transport-og-reiseliv/statistikker/drosje. [Funnet 05 06 2020].

[5] S. Stephansen, «Regjeringen,» 2020. [Internett]. Available: https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/. [Funnet 23 April 2020].

[6] Uber, «Uber Newsroom,» 2020. [Internett]. Available: https://www.uber.com/nb-NO/newsroom/historikk/. [Funnet 18 May 2020].

[7] D. Bach, «David BachDavid Bach,» 2019. [Internett]. Available: https://e24.no/naeringsliv/i/AdkBEq/ubers-norden-sjef-varsler-comeback-i-norge. [Funnet 23 April 2020].

[8] A. R. Sverre, «Erfaringer med deregulering av drosjemarkedet i,» Oslo Economics, Oslo, 2019.

[9] S. S. Tor Midtbø, «Regjeringen,» 2020. [Internett]. Available: https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/ytransport/sporsmal-og-svar-om-nytt-drosjeregelverk/id2641640/. [Funnet 22 April 2020].

[10] B. E. Thon, «Datatilsynet,» 2019. [Internett]. Available: https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/. [Funnet 24 April 2020].

[11] O. Storm-Paulsen, «Lovdata, Personopplysningsloven,» 15 Juni 2018. [Internett]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38. [Funnet 22 April 2020].

[12] «Lov om behandling av personopplysninger (personopplysningsloven),» [Internett]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38/gdpr/ARTIKKEL_83#gdpr/ARTIKKEL_83. [Funnet 03 06 2020].

[13] «Hva er nytt med personvernforordningen?,» [Internett]. Available: https://www.datatilsynet.no/regelverk-og-verktoy/lover-og-regler/hva-er-nytt/. [Funnet 03 06 2020].

[14] H. Øverby, «smarttelefon i Store norske leksikon,» 2018. [Internett]. Available: https://snl.no/smarttelefon. [Funnet 27 April 2020].

[15] A. Dossey, «Clearbridge mobile,» 2019. [Internett]. Available: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/. [Funnet 27 April 2020].

[16] R. Native, «React Native,» 2019. [Internett]. Available: https://reactnative.dev/. [Funnet 27 April 2020].

[17] «Web framework ranking,» [Internett]. Available: https://hotframeworks.com/. [Funnet 04 06 2020].

[18] T. Occhino, «Facebook Engineering,» 2015. [Internett]. Available: https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/. [Funnet 22 April 2020].

[19] R. G. J. R. Bhairav Acharya, «Cell Phone Location Tracking,» Samuelson Law, Technology & Public Policy Clinic, UC Berkeley, 2016.

[20] M. Smallcombe, «Xplenty,» 2019. [Internett]. Available: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/. [Funnet 24 April 2020].

[21] K. Sander, «estudie,» 2019. [Internett]. Available: https://estudie.no/nettverksserver/. [Funnet 28 April 2020].

[22] D. d. Borde, «Two-factor authentication,» Siemens Insight Consulting , UK, 2007.

[23] K. Sander, «estudie,» 2019. [Internett]. Available: https://estudie.no/ssl/. [Funnet 29 April 2020].

[24] Acuentix, «Acuentix,» 2020. [Internett]. Available: https://www.acunetix.com/websitesecurity/sql-injection/. [Funnet 29 April 2020].

[25] G. B. C. L. B. V. Pete Deemer, «THE SCRUM PRIMER,» 2009. [Internett]. Available: https://docplayer.net/15545261-The-scrum-primer-by-pete-deemer-gabrielle-benefield-craig-larman-bas-vodde-version-1-1-certified-scrum-training-worldwide-www-scrumti.html. [Funnet 25 Mai 2020].

[26] J. K, «Acodez,» 01 Juni 2018. [Internett]. Available: https://acodez.in/12-best-software-development-methodologies-pros-cons/. [Funnet 25 Mai 2020].

[27] flexify, «flexify,» 2019. [Internett]. Available: https://www.flexify.no/blogg/agile-metoder. [Funnet 25 Mai 2020].

[28] O. Storm-Paulsen, «Lovdata,» 15 Juni 2018. [Internett]. Available: https://lovdata.no/dokument/NL/lov/2018-06-15-38. [Funnet 22 April 2020].

[29] «CS50's Mobile App Development with React Native,» [Internett]. Available: https://courses.edx.org/courses/course-v1:HarvardX+CS50M+Mobile/course/. [Funnet 2020].

[30] «Expo Documentation,» [Internett]. Available: https://docs.expo.io/. [Funnet 2020].

[31] «React Navigation,» [Internett]. Available: https://reactnavigation.org/. [Funnet 2020].

[32] «A Predictable State Container for JavaScript Apps,» [Internett]. Available: https://redux.js.org/. [Funnet 2020].

[33] «WebStorm,» [Internett]. Available: https://www.jetbrains.com/webstorm/. [Funnet 2020].

[34] «Let's Encrypt - Free SSL/TLS Certificates,» [Internett]. Available: https://letsencrypt.org/. [Funnet 2020].

[35] «Jest | Delightful JavaScript testing,» [Internett]. Available: https://jestjs.io/.

[36] «JSDoc documentation,» [Internett]. Available: https://jsdoc.app/. [Funnet 05 2020].

[37] «Prettier - Opinionated Code Formatter,» [Internett]. Available: https://prettier.io/. [Funnet 05 2020].

[38] G. B. C. L. B. V. Pete Deemer, «THE SCRUM PRIMER,» 2012. [Internett]. Available: https://scrumprimer.org/scrumprimer20.pdf. [Funnet 24 Mai 2020].

12. Vedlegg

A. JSDoc

versjon 06.01.2020

Avtale mellom partene

Bedrift/virksomhet: MI OG MA HOLDING AS

Student(ene): Karol Debik, Kevin Andre Helgeland, Kenneth Langdahl Krogstad

og NTNU, IDI AIT

Oppg.nr: 40 Studentprosjekt Fast Track Taxi

Studieprogram: Dataingeniør (ITHINGDA)

Gjennomføring
Studenten skal gjennomføre et studentprosjekt i samarbeid med bedriften/virksomheten. IDI AIT veileder arbeidet faglig. Det er utarbeidet retningslinjer for gjennomføring av studentprosjekt som beskriver oppgavefordeling og hvordan studentprosjekter gjennomføres. Retningslinjene tar også opp ansvarsfraskrivelse, opphavsrettigheter og tilgjengelighet med muligheter for individuelle avtaler.

Ansvarsfraskrivelse
Instituttet er ikke ansvarlig for eventuelle ødeleggelser som studenten måtte påføre oppgavestillers utstyr direkte eller som følge av programvare studenten lager og/eller bruker, eller som studenten på annen måte medvirker til.

Opphavsrett og tilgjengelighet
Når ikke annet er avtalt, eier studenter selv den IPR (immaterielle rettigheter) de skaper som en del av studier/studieopphold ved IDI AIT. Alle resultater er åpent tilgjengelig. Opphavsretten reguleres av Åndsverksloven. Avtaler som inngås mellom IDI AIT og studenter skal som minimum sikre instituttet rett til å bruke generert IPR til utdannings- og forskningsformål. IDI AIT skal også motta en vurderingskopi av arbeidet inkludert eventuell kildekode. NTNU oppfordrer sterkt til publisering av alle bachelor- og masteroppgaver.

Marker med kryss det som gjelder denne oppgaven:

- [ ] Normalsituasjonen: Studentene har selv alle rettigheter knyttet til resultatet fra bacheloroppgaven, med de unntak som er beskrevet over.
- [x] Oppdragsgiveren har rettighetene og kan utnytte produktet kommersielt og videreutvikle produktet/metoden. Instituttet vil ikke utnytte produktet kommersielt, men vil kunne arbeide videre med den grunnlagskompetansen som er vunnet gjennom prosjektet, som beskrevet over.
- [ ] Resultatene fra arbeidet legges ut som OpenSource iht lisens _____ (Se http://creativecommons.no/lisenser).
- [ ] Bacheloroppgaven (det skriftlige arbeidet) skal være undergitt utsatt offentliggjøring i __ år (maks 3).

Oppdragsgiver er selv ansvarlig for å avtale håndtering av eventuelle konfidensielle opplysninger med veileder/sensor og studenten(e).

Denne avtalen er underskrevet i 3 – tre - eksemplarer hvor partene skal ha hver sin.

10.01.2020, Trondheim
(dato, sted)

Bedrift/virksomhet	Veileder ved IDI AIT	Student(ene)
		Karol Debik
		Kevin A. Helgeland
		Kenneth L. Krogstad

Arbeidstittel:	Fast Track Taxi

Hensikten med oppgaven:
I Taxi-næringen skjer det stor omveltninger sommeren 2020. Det blir nesten fri-slipp på løyver, og en må regne med at det dukker opp flere nye selskaper og at de vil konkurrere på pris, og ant taxier øker.
Dette gjør at lønnsomhet minsker for sjåfører og taxi-eiere, da de i større del av dagen vil vente på neste tur. Sjåfør-yrket blir mindre attraktivt pga nedgang i lønn.
Som en følge av dette ser vi at det nå kommer sjåfører som behersket norsk dårligere, og som kan ha begrensende geografikunnskaper.
Ikke alle setter pris på denne utviklingen.
For mange av kundene som er 50+ er ikke pris viktig, men servise i form av at de kan enkelt snakke med sjåfører og sjåføren vet hvor de skal hentes og hvor de skal kjøres.
Denne kundegruppen er heller ikke spesielt glad i å stå i taxi-kø, og betaler gjerne litt for å slippe det.
Mange i denne kundegruppen har også ofte noe nedsatt syn, og ønsker ikke å knote på en app med å skive inn henteadresse ol.
De vil trykke på en knapp, bli oppringt av sjåføren, og si hvor de vil bli hentet. Er det mangler på ledige taxier, vil de kjøpe seg ut av køen.
Sjåførene blir av oss kvalitetssikret både i norsk muntlig, og lokal geografi.
Vurdering av interessenter:
- Kundene får de sjåførene som gir servicen de vil ha
- De beste sjåførene for mer kjøring i perioder med lite etterspørsel av taxier, og bedre betalt for turen i perioder med stor etterspørsel etter taxier. Totalt bedre inntjening
- Vi tjener noen kroner pr tur som går via Fast Track Taxi
Appen skal være gratis å laste ned, og betaling skjer via et lite påslag på taxi-tur-prisen.
Vi har god erfaring i at de som utvikler IT-løsningen fortsetter å jobbe noe i firmaet i ettertid, eller også går inn på eiersiden.

Kort beskrivelse av oppgaveforslag:
Utvikle en app som gir taxi-kunder følgende 2 fordeler:
- Når det er ledige Taxier, får kundene de beste sjåførene.
- Når det er mangler på Taxier, kan kundene kjøpe seg fram i køen.
Taxi-sjåfører skal ha samme app men med annen funksjonalitet.
- Når en kunder ber om tur, skal de 12 nærmeste taxien i rekkefølge ut fra avstand til kunde få tilbud om å ta turen.
- taxisjåfør skal kunne ringe kunde tilbake, eller svare med sms ut fra hva kunden foretrekker.
Appen skal ved hjelp av GPS holder orden på hvor kunder og taxier er, og koble sammen de som er nærmest.
Det vil være noe sanntidsutfordringer, og køproblematikk ved at flere taxier får samme tilbud om å ta tur, men den som trykker først får turen.

Oppgaven passer for (kryss av de(t) som passer og skriv evt. en kommentar til oss):	- Bacheloroppgave - Arne Pukstad
Kan oppgavestiller stille arbeidsplass med nødvendig utstyr og programvare:	Nei, men lite behov utover pc
Hvis ikke, hva kreves av maskin og programvare:	PC og app-utviklingsverktøy
Begrensninger i tilgjenglighet av opplysninger o.l:	Forventer at konseptet ikke deles med utenforstående
Oppgaven passer best for, antall studenter:	- 3 - 4

Opplysninger om oppgavestiller

Navn på bedrift eller organisasjon:	MI OG MA HOLDING AS
Addresse	Postadresse: Brattvollvegen 2A - Kontoradresse: Kjøpmannsgata 65
Postnummer	7056
Poststed	RANHEIM
Navn på kontaktperson/veileder:	Arne Pukstad Juliussen
Telefon:	92634300
Epost:	arne.pukstad@gmail.com
Navn på kontaktperson 2/veileder 2:	Mikael Alexander Pukstad
Telefon kontaktperson 2/veileder 2:	92634300
Epost kontaktperson 2/veileder 2:	mapuksta@stud.ntnu.no

Utfyllende kommentarer til hva oppgaven gjelder:
Det skal ikke være noe betalingsløsing inn i app, men historiske data på bestilling av alle turer må være lett tilgjengelig.
Vi skal heller ikke ha noe kart som viser ledige biler.
Det viktigste er at det skal være ENKELT for kunde. Trykke på en knapp på mobil – si hvor de skal hentes – bil kommer.

Mi og Ma Holding AS er et selskap som utvikler og etablerer nye selskaper. Vi er i dag på eiersiden i 2 selskaper, og har som mål at denne oppgaven skal i løpet av 2020 resultere i selskapet "Fast Track Taxi System AS"
Er det ønske fra studentene, stiller vi gjerne opp til et møte for å klargjøre ytterligere forhold rundt oppgaven. Ring 926 34 300 eller mail til arne.pukstad@gmail.com