
GRUPPEPROSJEKT

Software Engineering

HØST 2020

FABIAN BERGET LINDBLAD
KACPER RAFAL SLIWA
RIKKE HEDELUND HANSEN
ALEKSANDER LENGARD

GRUPPE 19

HØGSKOLEN I ØSTFOLD
AVDELING FOR INFORMASJONSTEKNOLOGI

Innhold

1	Innledning	3
1.1	Gruppemedlemmer	3
2	Introduksjon	4
2.1	Bakgrunn	4
2.2	Problemstilling	4
2.3	Mål	4
3	Målgruppe	6
3.1	Personas	6
3.1.1	Privatperson	6
3.1.2	Bedrifter	7
3.2	Brukerhistorie	7
3.2.1	Privatpersoner	8
3.2.2	Bedrift	9
3.3	User Case	9
3.4	User Case Diagram	13
4	Krav	14
4.1	Funksjonelle krav	14
4.1.1	Opprette bruker	14
4.1.2	Logge inn	14
4.1.3	Tofaktorisering	15
4.1.4	Legge inn/ redigere parkeringsplass	15
4.1.5	Leie parkeringsplass	16
4.1.6	Søke parkeringsplasser	16
4.1.7	Brukerprofil	16
4.1.8	Betaling	16
4.1.9	Redigering av informasjon	17
4.1.10	Varsling	17
4.1.11	Fjerne parkeringsplass/bruker/profil	17
4.1.12	Verifisering	17
4.1.13	Hjelp og Support	18
4.2	Ikke-funksjonelle krav	18
4.2.1	Personopplysningsloven	18
4.2.2	Krav om universell utforming	18
4.2.3	Sikkerhet	18
4.2.4	Tekniske krav	18
4.2.5	Eksterne Avhengigheter	18
4.3	Videre utvikling av tjenesten	19
5	Estimering	20
5.1	T-shirt estimering av kravene	20

6	Systemarkitektur	22
7	Prototype	23
7.1	Beskrivelse av systemet	23
7.1.1	Funksjonalitet og avgrensinger	23
7.1.2	Modellering av prototypen	24
7.1.3	Avhengigheter	30
7.1.4	Kjente svakheter og problemer	31
7.2	Eksempler	31
7.2.1	Profilside	31
7.2.2	Leie parkeringsplass	33
7.3	Installasjon av systemet	35
7.3.1	Server	35
7.3.2	Client	35
7.4	Bruk	36
7.4.1	Server	36
7.4.2	Client	36
7.5	Testing	37
7.5.1	Oversikt over filer	37
7.5.2	Kjøre tester	38

1 Innledning

I faget Software Engineering og Testing har gruppen fått i oppgave å lage produktdokumentasjon for en tjeneste en oppstartsbedrift ønsker å utvikle. Vi skal skape kjernesystemet i denne tjenesten, og avdekke hvilke funksjoner og egenskaper som må være med for å imøtekomme kundens ønske. I tillegg skal det lages en prototype, et «minste brukbare produkt» (MBP) som viser hvordan systemet er tenkt brukt.

1.1 Gruppemedlemmer

Fabian Berget Lindblad
Informatikk, 2. året.
fabian.b.lindblad@hiof.no

Kacper Rafal Sliwa
Informasjonssystemer, 2. året.
kacper.r.sliwa@hiof.no

Rikke Hedelund Hansen
Informatikk, 2. året.
rikke.h.hansen@hiof.no

Aleksander Lengard
Informatikk, 2. året.
aleksander.lengard@hiof.no

2 Introduksjon

2.1 Bakgrunn

I et samfunn hvor flere og flere bruker bil som transportmiddel, er et stadig økende problem i storbyer mangel på parkeringsplasser. Det er flere som bor i utkanten og jobber i sentrum og gjerne bruker bilen til og fra jobb. Dette har skapt stor etterspørsel etter parkering i de områdene det gjelder, noe som ikke prioriteres i moderne byutvikling. De plassene som allerede er tilgjengelig er gjerne dyre og ikke alltid tilgjengelig der du trenger de. Men det finnes også de som eier parkeringsplasser eller områder hvor det kan være mulig for folk å parkere. Noen har parkeringsplass i forbindelse med boligen de eier, men har kanskje ikke bil. Alternativt bruker de sin egen bil til jobb, og dermed er parkeringsplassen ledig i normal arbeidstid.

Det er vanskelig for privatpersoner å ha en oversikt og håndtere dette på egenhånd. Så det er dermed et ønske om å utvikle en tjeneste hvor brukere kan leie tilgjengelige parkeringsplasser i perioder. I tillegg til å kunne leie ut sin egen plass når de ikke trenger den selv.

2.2 Problemstilling

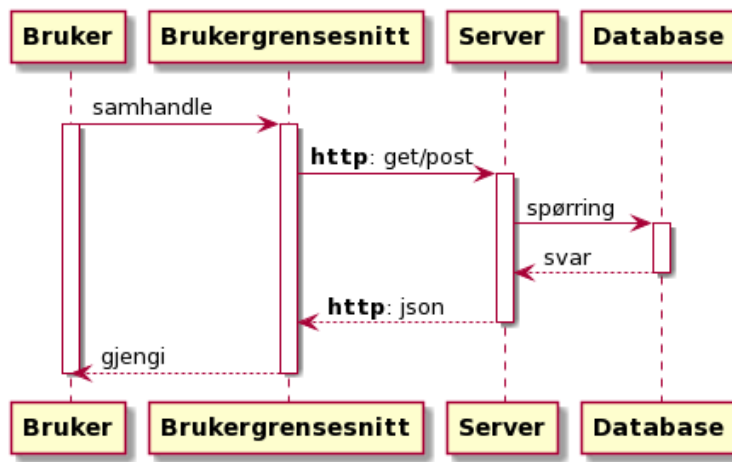
Tjenesten skal ha funksjoner som på en enkel måte lar brukere leie andre sine parkeringsplasser og leie ut sin egen om de ønsker det. Betaling for leien skal skje igjennom tjenesten.

Det skal også være mulig for brukeren å se en oversikt over de plassene man selv leier ut. I tillegg skal det være mulig for brukeren å se plasser som tidligere har vært leid.

Et ønske er at tjenesten skal være tilgjengelig på pc, nettbrett og telefon. Slik at tjenesten er tilgjengelig når man ønsker det.

2.3 Mål

For at tjenesten skal være tilgjengelig på flere plattformer trenger kjernesystemet å være tilgjengelig via internett. Ved å lage en API (*application programming interface*) så kan forskjellige typer brukergrensesnitt kommunisere med systemet på samme måte. Dette uten å måtte tilpasse kjernesystemet til de forskjellige. Det er kjernesystemet som behandler alt av forespørsler, og det er her forretningens logikk håndterer dataflyten til og fra databasen.



Figur 1: Overordnet oversikt over dataflyt mellom brukergrensesnitt, server og database.

I denne oppgaven har vi valgt å lage et forenklet brukergrensesnitt for nettleteren. Men mulighetene er mange for hvor en slik tjeneste kan passe inn. Den kan tilpasses mobil og nettbrett eller mulighet for å integrere tjenesten i andre systemer, slik som hotell-bookingsider, airbnb, eller liknende.

Målet er også å skape brukervennlige funksjoner, som er lette å forstå. Ergo at tjenesten skal kunne brukes av voksne i alle aldre, uten spesiell teknisk kunnskap. Tjenesten skal oppfylle de kravene som raskest gir kunden verdi. I form av en prototype har vi utviklet et «*minste brukbare produkt*», som viser kjernefunksjonene i systemet.

For at oppstartsbedriften skal kunne tjene penger på ideen sin er det viktig at løsningen vil generere penger og brukere samtidig. Vi foreslår at brukere som kun leier plass ikke trenger å betale for tjenesten. For de som ønsker å leie ut, så er de første 6 månedene avgiftsfrie. Etter det vil det påløpe et gebyr på 20% av leieinntektene. Dette vil sørge for at brukere som er interessert kan prøve tjenesten kostnadsfritt. På den måten kan det skape en brukerbasis raskest mulig. Skal du leie ut mer enn 3 plasser, blir du ansett som næringsdrivende. Da vil det bli en løpende månedskostnad på noen hundre kroner.

Videre i denne prosjektdokumentasjonen vil vi beskrive de ulike delene av systemet i mer detalj, og forklare mer om hva og hvordan vi har tenkt at et slik system skal fungere. I tillegg til hvem brukerne er, vise til tenkte bruker-scenarier, estimere og forklare vår prototype.

3 Målgruppe

Målgruppen for tjenesten er voksne privatpersoner som enten har bil eller eier en parkeringsplass. Dette vil si i aldersgruppen 18 år eller eldre. Næringsvirksomhet som har eiendom til disposisjon, skal også ha mulighet for å bruke tjenesten. Tjenesten skal være brukervennlig slik at alle i målgruppen kan på en enkel måte benytte seg av tilbudet. Vi har fokusert på privatpersoners bruk når vi har utviklet prototypen.

3.1 Personas

Personas er personer som blir funnet opp for å etterligne mulige brukere av tjenesten. Meningen med fiktive personer er å finne viktige funksjoner og krav i systemet ut fra brukerens behov. Disse skal virke så ekte som mulig slik at de ligner på potensielle kunder. Hver personas bør være så unik som mulig slik at vi kan få frem forskjellige krav. Det er viktig å lage slike fiktive personer under utvikling av en tjeneste slik at man tidlig kan avdekke brukerens behov til tjenesten. Ved å lage personas er det mye lettere å se for seg hva som trengs i systemet ettersom at man kan se krav i systemet fra brukerens perspektiv. Vi har tatt for oss forskjellige brukergrupper som kommer til å bruke systemet, og laget forskjellige personas til dem. Disse er delt opp i alder for å avdekke forskjellige krav og brukermønster blant brukergruppene.

3.1.1 Privatperson

For privatpersoner er det fire personas som er i forskjellige alder.

Jørn Jensen

Alder: 70

Arbeid: Pensjonert

Sivilstatus: Gift

Simon Berger

Alder: 34

Arbeid: Finansanalytiker hos BANE Nord

Sivilstatus: Gift med to barn

Bård Kronvol

Alder: 55

Arbeid: IKT for kommunen

Sivilstatus: Nyskilt med barn

Siri Kristiansen

Alder: 23

Arbeid: Nyutdannet sykepleier på Rikshospitalet

Sivilstatus: Singel

3.1.2 Bedrifter

For bedrifter har vi valgt en ansatt ved bedriften som har ansvaret for parkeringsplassene. I dette tilfellet er det eieren av bedriften.

Arne Arnolfsen

Alder: 45

Arbeid: Eier av en lokal rørleggerbedrift.

Sivilstatus: Enkemann

3.2 Brukerhistorie

Brukerhistorier er korte historier som beskriver en person og utfordringer der de kan ha nytte av systemet. Brukerhistorie er en beskrivelse av personas i en kontekst som hensiktsmessig skal klargjøre kravene til et system.

Som utviklere ønsker vi å bruke disse historiene for å hente ut korte beskrivelse av en brukers ønsket funksjon og løsning til et gitt problem eller situasjon. Disse formuleres gjerne på en slik form: Som «ROLLE» ønsker jeg «FUNKSJON» for å oppnå «NYTTEVERDI».

3.2.1 Privatpersoner

3.2.1.1 Jørn Jensen

Jørn Jensen er en pensjonert mann som liker å være sosial. Han og kona får tilbud om å gå på Dagsenter i Halden, som han elsker å gjøre. Der får han sitte lenge og skravle med andre eldre. Kona Kjersti har problemer med å gå, og de bor to timers gangavstand unna Dagsenteret slik at de bruker bil for å komme dit. Dagsenteret har få parkeringsplasser, og de er alltid opptatt. Jørn ønsker da en parkeringsplass de kan leie i et gitt tidsrom for en rimelig sum, slik at de kan komme seg til Dagsenteret.

Som en leietager ønsker jeg å leie en parkeringsplass slik at jeg fortsatt kan bruke bil til dagsenteret.

3.2.1.2 Simon Berger

Simon Berger, er en travel person som jobber i finansmiljøet i Oslo. Han bor i utkanten av byen, på Jar i Bærum sammen med sin kone og to barn. Barna er i barnehagealder, og for at Simon skal rekke å hente barna etter jobb er han nødt til å bruke bil – kollektivt tar for lang tid. I nærheten av jobben er det ikke tilgjengelige plasser, og jobben tilbyr ikke de ansatte parkering. Det blir fort dyrt for Simon å parkere i parkeringshus hver dag. Så han skulle ønske det var en mulighet for å parkere rimeligere i byen mens han er på jobb.

Som en leietager ønsker jeg å leie en parkeringsplass rimeligere enn å stå i dyre parkeringshus.

3.2.1.3 Bård Kronvold

Bård Krovold er IKT-ansvarlig i kommunen han jobber for og stortrives med å ha noe og fikse på. Bård er nyskilt og barna hans har flyttet ut, dermed sitter Bård igjen alene i ett større hus sentralt i Oslo med to parkeringsplasser hvor han kun trenger en, og i ukedagene er ofte begge ledig. Ved å leie ut den ekstra parkeringsplassen vil Bård få noen ekstra kroner han kan bruke på hobbyene sine, og kanskje han kan bli kjent med noen nye samtidig.

Som en utleier ønsker jeg å leie ut parkeringsplassen som står tom slik at jeg kan tjene ekstra penger.

3.2.1.4 Siri Kristiansen

Siri Kristiansen er en nyutdannet sykepleier som jobber på Rikshospitalet. Hun er nettopp ferdig med studier, og bor fortsatt i byen sammen med venninnen

Kari. Som nyutdannet er økonomien trang, men hun er heldig som har klart å kjøpe seg en leilighet. Hun leier ut ett rom til venninnen Kari for å få ting til å gå rundt. Hun kunne tenkte seg å tjene noen ekstra kroner, slik at hverdagen blir litt enklere. Verken Kari eller Siri disponerer bil, men leiligheten Siri eier disponerer en parkeringsplass. Denne står for det meste tom, og brukes kun de få gangene de får gjester på besøk, som er på ettermiddagen eller i helgene.

Som en utleier ønsker jeg å leie ut parkeringsplassen når den ikke blir brukt slik at jeg kan betjene boliglånet mitt.

3.2.2 Bedrift

3.2.2.1 Arne Arnolfsen

Arne Arnolfsen er en arbeider med godt mot. Han elsker jobben sin, og etter konas død er det hans eneste positive i hans liv. Rørleggerbedriften har holdt til i samme lokaler sentralt i Oslo i mange år. De er heldige som har en stor parkeringsplass hvor alle firmabilene står parkert over natten. På dagtid står de fleste av plassene tomme, for arbeiderne er ute på jobb hos kunder med bilene. Arne er en kremmer, så han lurar på om det er mulig å leie ut disse plassene på dagtid.

Som en utleier ønsker jeg å legge ut flere parkeringsplasser når de ikke blir brukt slik at bedriften kan tjene mer penger.

3.3 User Case

User Case er en beskrivelse på hvordan en person skal bruke en del av systemet. I hvert user case blir en av personene brukt til å beskrive en situasjon med ønsket funksjon i systemet. Beskrivelsen til en user case bør være så spesifikk som mulig for å kunne få fram interaksjoner mellom brukeren og systemet. Dette hjelper med å finne hvordan forskjellige funksjoner skal implementeres og hvordan deler av systemet skal fungere ut fra brukerens opplevelse.

I tabellene 1, 2, 3, 4 og 5 ser man user casene til de ulike funksjonene som våre personas ønsker å se og bruke.

System	Delingtjeneste for parkeringsplasser.
Use case	Leie parkeringsplass.
Aktor	Jørn Jensen.
Krav	Bruker må være på nettsiden og logget inn.
Trigger	Å kunne finne ledig parkeringsplass.
Beskrivelse	Brukeren åpner nettsiden og logger inn. Deretter trykker brukeren på «se tilgjengelig parkeringsplasser» for å komme til siden for tilgjengelige parkeringsplasser. Der fyller han inn informasjonen om hvor parkeringsplassen skal befinne seg, og får med det opp alle ledige plasser i området. Brukeren velger så den plassen han vil ha og går videre til betalingssiden. Der får han opp plassen han ønsker å leie og prisen. Så velger han foretrukket betalingsmåte, Vipps eller Mastercard. Deretter blir han ført til tredjepartens betalingsside og betaler der. Når brukeren har betalt og betalingen er godkjent så blir han ført tilbake til betalingssiden. Her får brukeren en bekreftelse på siden om at parkeringsplassen er betalt for og nå er aktiv.
Stimuli	Leid parkeringsplass.
Respons	Mottar bekreftelse om at betaling er godkjent, deretter blir parkeringsplassen utlånt til brukeren og utilgjengelig for andre brukere.

Tabell 1: User Case for å leie parkeringsplass

System	Delingtjeneste for parkeringsplasser.
Use case	Søke etter parkeringsplasser.
Aktor	Simon Berger.
Krav	Bruker må være innlogget inn i tjenesten.
Trigger	Ønsker å finne parkeringsplasser i Oslo.
Beskrivelse	Brukeren åpner nettsiden og logger inn. Deretter trykker brukeren på «se tilgjengelig parkeringsplasser» for å komme til siden for tilgjengelige parkeringsplasser. Der fyller brukeren inn informasjonen om hvor parkeringsplassen skal befinne seg, og får med det opp alle ledige plasser i området. Finner brukeren en ledig plass legger han den til og føres videre til betalingssiden. Der får han opp plassen han ønsker å leie og prisen. Så velger han foretrukket betalingsmåte, Vipps eller Mastercard. Deretter blir han ført til tredjepartens betalings-side og betaler der. Når brukeren har betalt og betalingen er godkjent så blir han ført tilbake til betalingssiden. Her får brukeren en bekreftelse på siden om at parkeringsplassen er betalt for og nå er aktiv.
Stimuli	Finner parkeringsplass i området.
Respons	Mottar bekreftelse om at betaling er godkjent, deretter blir parkeringsplassen utlånt til brukeren og utilgjengelig for andre brukere.

Tabell 2: User Case for å søke opp parkeringsplass

System	Delingtjeneste for parkeringsplasser.
Use case	Se oversikt over inntekter.
Aktor	Bård Kronvold.
Krav	Bruker må være innlogget inn i tjenesten
Trigger	Ønsker å se inntektene over sin parkeringsplass
Beskrivelse	Brukeren åpner nettsiden og logger inn. Deretter trykker brukeren på "min side", og kommer til siden hvor han kan se oversikten over inntektene sine for parkeringsplassene. Brukeren kan da velge å se oversikten over alle parkeringsplassene, eller en bestemt parkeringsplass.
Stimuli	Finne hvor mye inntekter brukeren får.
Respons	Ser da hvor mye inntekter han får på den ene parkeringsplassen som han har.

Tabell 3: User Case for å se oversikt

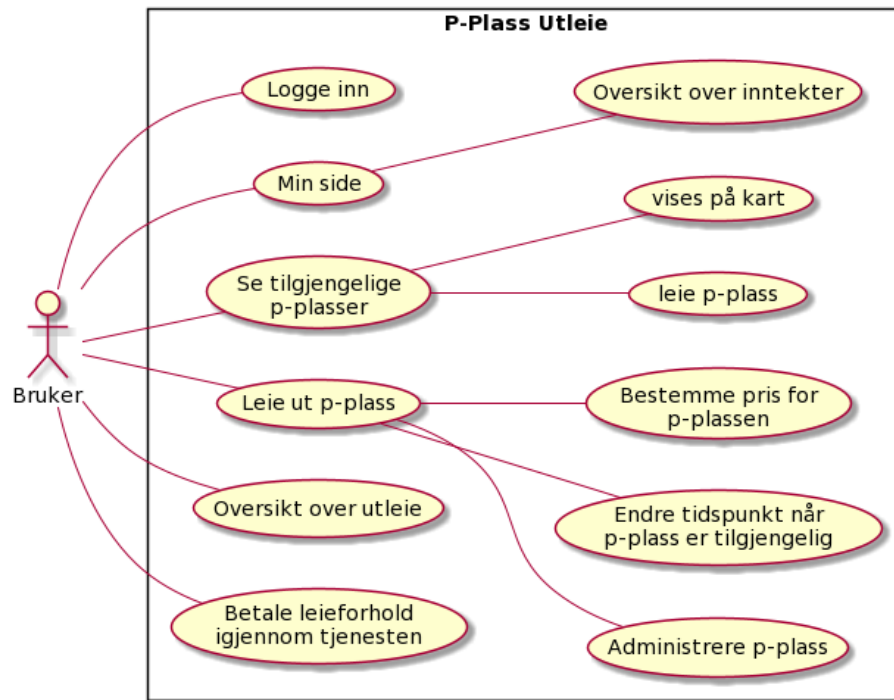
System	Delingtjeneste for parkeringsplasser.
Use case	Legge ut parkeringsplass.
Aktor	Siri Kristiansen.
Krav	Bruker må være innlogget inn i tjenesten. Parkeringsplass må være mulig å legge til.
Trigger	Lyst til å legge ut en parkeringsplass til leie.
Beskrivelse	Brukeren åpner nettsiden og logger inn. Deretter trykker brukeren på «lei ut parkeringsplass» for å komme til siden for å legge til sin parkeringsplass. Der fyller hun ut all informasjonen om parkeringsplassen som pris, lokasjon og tilgjengelighet. Så legger hun den til og får en bekreftelse om at parkeringsplassen er opprettet og gjort tilgjengelig på siden.
Stimuli	Legger ut parkeringsplass i tjenesten.
Respons	Parkeringsplassen har blitt lagt ut og vises i lei parkeringsplass.

Tabell 4: User Case for å lage parkeringsplass

System	Delingtjeneste for parkeringsplasser.
Use case	Legge ut flere parkeringsplasser på siden.
Aktor	Arne Arnolfsen.
Krav	Være innlogget på sin profil og være registrert som firma. Må ha betalt den månedlige prisen og parkeringsplass må være mulig å legge til.
Trigger	Ønske å legge ut parkeringsplassene
Beskrivelse	Brukeren åpner nettsiden og logger inn. Deretter trykker brukeren på «lei ut parkeringsplass» for å komme til siden for å legge til sin parkeringsplass. Der fyller brukeren ut all informasjonen om parkeringsplassen og legger den til. Her får firmabrukeren en bekreftelse på siden om at parkeringsplassen er betalt for og nå er tilgjengelig for andre å leie.
Stimuli	Legger ut parkeringsplassen offentlig så andre kan leie den.
Respons	Bekrefter at parkeringsplassen har blitt lagt ut og gjort tilgjengelig i systemet.

Tabell 5: User Case for å legge ut parkeringsplasser for firmaet

3.4 User Case Diagram



Figur 2: User case diagram med forskjellige funksjoner for tjenesten

User case diagrammet (Figur 2) kommer fra de ulike user casene vi har vist i 3.3. Diagrammet viser hva som er felles for user casene og hvilke funksjoner som kan anses som mer viktig. Lei ut parkeringsplass er for eksempel det som nevnes flere ganger i user casene og dermed er i selve diagrammet.

Bakgrunnen for dette diagrammet er å avdekke krav. Det viser hva personene ønsker å ha med og hva vi kan anse som viktig for systemet.

4 Krav

Denne seksjonen forteller om kravene vi finner til systemet, altså krav til innholdet. Det vil si alle funksjoner og funksjonaliteter som blir implementert i systemet. Disse finner vi ved hjelp av brukerhistoriene, user casene og user case diagrammet som vi har laget i forrige seksjon.

Vi har definert flere roller i kravene. Disse rollene er leietager, utleier, firma og administrator. Det er for å spesifisere hvilke funksjoner som passer til ulike roller.

Firma er en bedriftskunde. De kan bare leie ut plasser, og vil betale en månedlig sum for tjenesten. Utleierne er privatpersoner som leier ut parkeringsplassene sine. De vil betale en prosentdel av inntekten etter en viss periode.

Leietager er privatpersoner som kan leie parkeringsplasser. De vil betale for plassen gjennom tjenesten. Administrator er eierene av tjenesten. Personer med denne rollen skal bidra med å holde tjenesten vedlike.

I systemet er leietager, utleier og firma definert som brukere. De har de samme funksjonene, bortsett fra firma som kun har mulighet til å leie ut.

4.1 Funksjonelle krav

4.1.1 Opprette bruker

Alle brukere skal kunne opprette en ny bruker i tjenesten.

- (a) Alle brukere skal kunne opprette en bruker i tjenesten med hjelp av epost adresse.
- (b) Leietageren/utleieren skal kunne opprette en bruker i tjenesten med hjelp av Facebook konto.
- (c) Alle brukere skal kunne opprette en bruker i tjenesten med hjelp av Google konto.
- (d) Firma må oppgi organisasjonsnummer ved opprettelse av firmabruker.

4.1.2 Logge inn

Alle brukere skal kunne logge inn i tjenesten.

- (a) Alle brukere skal kunne logge inn i tjenesten med hjelp av epost adresse.

- (b) Leietageren/utleieren skal kunne logge inn i tjenesten med hjelp av Facebook konto.
- (c) Alle brukere skal kunne logge inn i tjenesten med hjelp av Google konto.
- (d) Ved feil brukernavn eller feil passord skal systemet gi beskjed til brukeren at brukernavn eller passord er feil.
- (e) Alle brukere skal få beskjed om feil, hvis verifiseringen er feil.
- (f) Alle brukere skal kunne gjenopprette konto ved glemt passord.

4.1.3 Tofaktorisering

Alle brukere skal ha muligheten til å skru på tofaktoraутентisering for å logge seg inn i tjenesten.

- (a) Administrator skal ha tofaktoraутентisering for å logge inn på kontoen.
- (b) Utleieren/firma skal ha tofaktoraутентisering for å legge ut en parkeringsplass.
- (c) Alle brukere skal ha tofaktoraутентisering for å redigere brukerkonto informasjon om seg selv.
- (d) Tofaktoraутентisering skal kunnes gjøres med en SMS.
- (e) Tofaktoraутентisering skal kunne gjøres med en autoriseringsapplikasjon.

4.1.4 Legge inn/ redigere parkeringsplass

Utleieren/firma skal kunne legge til parkeringsplass(er) i tjenesten, slik at den kan bli leid ut.

- (a) Utleieren/firma skal kunne bestemme prisen på sine parkeringsplasser for ulike tidsrom.
- (b) Utleieren skal kunne sette plassen utilgjengelig de dagene utleier bruker plassen selv.
- (c) Ved utleie av større område (næring) skal det være mulig å spesifisere antall plasser.
- (d) Utleieren/firma skal kunne redigere informasjonen om sine egne parkeringsplasser.
- (e) Administrator skal kunne redigere informasjon om en parkeringsplass.

4.1.5 Leie parkeringsplass

Leietageren skal kunne leie en parkeringsplass i et gitt tidsrom.

- (a) Leietageren skal kunne få en bekreftelse om at parkeringsplassen er betalt av leietageren for det gitte tidsrommet
- (b) Leietageren skal kunne se antall utilgjengelig og tilgjengelig plasser på en lokasjon.

4.1.6 Søke parkeringsplasser

Leietageren skal kunne søke opp parkeringsplasser.

- (a) Leietageren skal kunne se tilgjengelige plasser i ønsket område i kart.

4.1.7 Brukerprofil

Alle brukere skal kunne ha en brukerprofil.

- (a) Leietageren skal kunne se oversikt over leide plasser.
- (b) Utleieren/firma skal kunne se oversikt over sine egne parkeringsplasser.
- (c) Alle brukere skal kunne redigere sin egen brukerkonto.
- (d) Administrator skal kunne ha en brukerprofil.
- (e) Administrator skal kunne se en oversikt over alle brukere.
- (f) Administrator skal kunne se en oversikt over alle utleide plasser og hvem som har leid de ut.
- (g) Utleieren/firma skal kunne se en oversikt av inntjeninger av parkeringsplass(ene).
- (h) Tjenesten skal vise en oversikt av inntjeninger i form av egnede grafer.

4.1.8 Betaling

Leietageren skal kunne betale for leie og utleie av parkeringsplasser ved å bruke Vipps/Mastercard.

- (a) Leietageren skal ha muligheten til å angre et eventuelt feilkjøp
- (b) Leietageren skal kunne se hva brukeren har betalt for.

- (c) Leietageren skal ha mulighet til å se alle sine tidligere betalinger.
- (d) Leietageren skal kunne avbestille den betalte parkeringsplassen og få tilbake pengene, senest 24 timer før parkeringen starter.
- (e) Firma skal kunne betale en fast månedsbeløp for å bruke tjenesten med faktura/avtalegiro.
- (f) Utleieren skal kunne betale en prosentvis av leieinntektene etter 6 måneders gratisperiode.
- (g) Utleieren skal kunne få penger for utleie til sin konto.

4.1.9 Redigering av informasjon

Administrator skal kunne redigere innholdet på siden, slik som «ofte stilte spørsmål» og nyheter.

4.1.10 Varsling

Leietageren skal kunne varsle om en utleier/firma som usaklig/upassende

- (a) Utleieren/firma skal kunne varsle om en leietager er usaklig/upassende.

4.1.11 Fjerne parkeringsplass/bruker/profil

Utleieren/firma skal kunne fjerne sine egne parkeringsplasser.

- (a) Administrator skal kunne fjerne en bruker som er usaklig/upassende.
- (b) Administrator skal kunne fjerne parkeringsplasser som er usaklig/upassende.
- (c) Alle brukere skal kunne slette sin egen profil.

4.1.12 Verifisering

Administrator skal kunne verifisere at parkeringsplassen tilhører faktisk utleieren/firma

- (a) Administrator skal kunne fryse en annonse frem til eierskap av plassen er bekreftet

4.1.13 Hjelp og Support

Det skal være enkelt å finne dokumentasjon om hvordan systemet er tenkt brukt i tjenesten. Dette gjøres ved at:

- (a) Alle brukere skal kunne se kontaktinformasjon til kundestøtte.
- (b) Alle brukere skal kunne se ofte stilte spørsmål.

4.2 Ikke-funksjonelle krav

4.2.1 Personopplysningsloven

Tjenesten skal tilfredsille krav i personopplysningsloven, som omhandler innsamling og bruk av personopplysninger. Dette gjelder de nasjonale reglene, samt EUs personvernsforordning (GDPR - General Data Protection Regulation).

- [Datatilsynet](#)

4.2.2 Krav om universell utforming

Tjenesten skal som ett minimum tilfredsstille minstekravet, beskrevet i det kommende regelverket for universell utforming av IKT-løsninger, WAD og WCAG 2.1. [Digitaliseringsdirektoratet](#)

- (a) Nettsiden som brukergrensesnitt skal være kompatibel med forskjellige skjermstørrelser, slik som mobiltelefon og nettbrett.

4.2.3 Sikkerhet

Sensitiv informasjon skal være lagret kryptert.

4.2.4 Tekniske krav

- (a) Applikasjonen skal bli utviklet med JavaScript for Node.js
- (b) Hoveddelen av systemet skal være utviklet mot bruk av en dokumentdatabase.

4.2.5 Eksterne Avhengigheter

Tjenesten skal ha forskjellige eksterne avhengigheter. Disse avhengighetene er

- (a) Skytjenesteleverandør skal være brukt for lagring av databaser
- (b) Skytjenesteleverandør skal være brukt for vert til tjenesten.
- (c) Betalingstjeneste som Vipps og Mastercard.
- (d) Google Maps.
- (e) Innloggingssystem som Facebook og Google.

4.3 Videre utvikling av tjenesten

For å utvikle tjenesten ytterligere, kan man legge til en funksjon hvor brukere kan gi tilbakemelding på parkeringsplassen de har leid. Dette kan være i form av stjerner som baserer seg på deres erfaring slik som plassens lokasjon, pris, og om den var lett tilgjengelig. Det vil også være mulig å legge ved en kommentar i tilbakemeldingen. I tillegg kan det også være nyttig for brukere å kunne sende meldinger for å kommunisere med hverandre.

Videre utvikling av «finn parkeringsplass» siden kan være å legge til flere filterings kriterier, slik at brukeren får sortert parkeringsplasser etter pris og nærmest lokasjon. Det bør også være mulig å utvide filtreringen med spesielle egenskaper slik som om plassen har tilgjengelig el-bil lader, er under tak eller i oppvarmet garasje. En annen egenskap er å legge til bilder av lokasjonen, en beskrivelse av plassen slik som mål og annen informasjon som kan være nyttig for leietager å vite.

For at utleiende og bedrifter kan få mer kontroll, kan det skapes en gruppe med funksjoner som lar disse brukerne se statistikk om sine plasser. Deretter gi dem mulighet for å betale ekstra for mer eksponering.

Ved behov for å leie flere plasser samtidig, skal det være mulig å inngå en avtale med utleiende som tilbyr de om å leie dem til en redusert pris. Det kan også være mulig for leietagere å inngå faste avtaler med utleier når plassen skal brukes regelmessig.

Systemet kan senere bygges ut med kobling til Google Maps, slik at brukeren enkelt kan trykke på en leid parkeringsplass og få veibeskrivelse til stedet.

5 Estimering

Estimering handler om å forutse hva som kan ta mest tid i utviklingen, og beregne hva som skal ha mest prioritet. Ved å sette opp en estimering vil det gjøre planlegging av oppgaver enklere og gi en god oversikt over prioriteringen.

T-shirt estimering er en sortert estimering basert på utviklingstid og forretningsnyttens av hvert krav. T-shirt kommer av skalaen som er definert som klesstørrelser, «small», «medium», «large» og «x-large». Et krav med «small» størrelse vil være enkelt å implementere, og har det «x-large» verdi vil kravet være ekstra viktig å få gjennomført. Nettoverdien vil da være fradraget mellom verdien og størrelsen, og det gir en god oversikt over hva som burde prioriteres i systemet.

	Utviklingstid			
Verdi	X-Large	Large	Medium	Small
X-Large	0	4	6	7
Large	-4	0	2	3
Medium	-6	-2	0	1
Small	-7	-3	-1	0

Tabell 6: Poengtavle for estimeringen av utviklingsstid og forretningsnyttens.

5.1 T-shirt estimering av kravene

Ved å estimere de forskjellige kravene sin utviklingsstid og forretningsnytte kan vi med poengtavlen (6) finne nettoverdien av estimeringen. Høyere nettoverdi vil si at kravet vil gi høy nytte for systemet i forhold til utviklingstiden. Dermed bør dette kravet prioriteres.

Krav	Feature	Tid	Verdi	Netto
4.1.1	Opprette bruker	Small	X-Large	7
4.1.2	Logge inn	Small	X-Large	7
4.1.4	Legge inn/redigere parkeringsplass	Small	X-Large	7
4.1.5	Leie parkeringsplass	Medium	X-Large	6
4.1.8	Betaling	Large	X-Large	4
4.1.6	Søke parkeringsplass	Medium	Large	2
4.1.7	Brukerprofil	Medium	Large	2
4.1.11	Fjerne parkeringsplass/bruker/profil	Medium	Large	2
4.1.13	Hjelp og Support	Medium	Large	2
4.1.10	Varsling	Medium	Medium	0
4.2.2	Universell Utforming	Medium	Medium	0
4.2.3	Sikkerhet	Large	Large	0
4.1.3	Tofaktorausautentisering	Large	Medium	-2
4.1.9	Redigering av informasjon	Large	Medium	-2
4.1.12	Verifisering	Large	Medium	-2
4.2.4	Tekniske krav	Large	Small	-3
4.2.5	Eksterne Avhengigheter	Large	Small	-3
4.2.1	Personopplysningsloven	X-Large	Medium	-6

Tabell 7: Vår estimering av kravene

Utifra Tabell 7 ser vi tydelig at kravene 4.1.1, 4.1.2, 4.1.4, 4.1.5 er, etter vår estimering, ansett å være de viktigste kravene. Det betyr ikke at de andre kravene ikke er viktige, men at det er disse kravene kundene får tidligst verdi og utbytte i tjenesten.

Det medfører at man vil fokusere på disse kravene først, før man implementerer de andre kravene. Det gjør vi i prototypen, hvor da basert på estimeringen har valgt kravene som gir mest utbytte.

6 Systemarkitektur

Systemarkitektur er et forslag til hvordan systemet kan bli bygget opp og samspillet mellom de forskjellige delene.

Tjenesten burde bli utviklet med «Model-View-Controller» arkitekturen. Da vil systemet deles opp i tre deler som samhandler med hverandre. Dette gjør det lettere å utvikle og vedlikeholde koden på større prosjekter. De tre delene er uavhengige av hverandre, som gjør det enkelt å bytte ut eller endre kode uten å påvirke de andre.

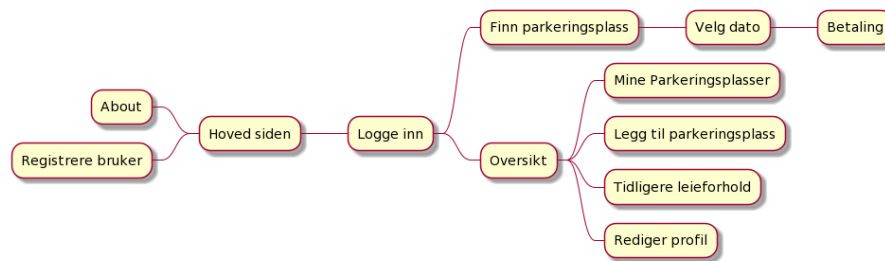
Model representerer programmet data og logikk. *View* korresponderer til brukergrensesnittet, som er visningen av data. Det er her brukeren samhandler med tjenesten vår. *Controller* er leddet mellom *view* og *model*. Denne behandler hendelser som skjer i view og henter dataen fra model og gjør dette tilgjengelig for view.

Tjenestens grensesnitt burde utvikles i samarbeid med brukergrensesnitt- og brukeropplevelse designere. Dette er for å gjøre tjenesten forståelig og lett å bruke. Dette samsvarer med kravet om universell utforming, hvor alle skal kunne bruke tjenesten. Ved å gjøre tjenesten lett og forståelig kan dette skape en større brukerbasis.

7 Prototype

7.1 Beskrivelse av systemet

Prototypen inneholder det vi har sett på som den viktigste funksjonaliteten. For å få mest mulig verdi ut av prototypen har vi begrenset den til å ikke inneholde en komplett løsning av alle kravene, blant annet betalingssystemet. Sikkerheten er også begrenset da prototypen kun skal være en foreslått løsning på hvordan systemet kan bli utviklet.



Figur 3: Oversikt over sidene i systemet

Figur 3 viser hvordan det er mulig å navigere rundt i tjenesten. Det meste av funksjonaliteten er lukket bak innloggingssiden fordi det kreves en brukerkonto for å benytte seg av disse funksjonene. «About» og «Hovedsiden» er tilgjengelig for alle for å gi en beskrivelse av hva tjenesten har å tilby. Deretter kan en bruker velge å opprette en profil for å få tilgang til tjenesten.

7.1.1 Funksjonalitet og avgrensinger

Vi ønsker at prototypen skal vise det vi anser som den mest sentrale delen i et tenkt system. For at kunden skal få mest utbytte av prototypen tidligst mulig har vi har sett på ulike krav vi ønsker å ha med. Dermed kommet frem til ulike funksjonaliteter som er viktige for prototypen. Våre user cases i seksjon 3.3 og estimeringen gjort i tabell 7 har bidratt til å avdekke de viktigste funksjonene i systemet. Muligheten til å leie og legge ut en parkeringsplass er sentrale funksjoner som gir tjenesten mening og dermed viktig å ha med. Muligheten til å søke etter parkeringsplasser er også inkludert sammen med andre funksjoner som gjør brukervennligheten av tjenesten bedre, slik kan kunden få enda mer ut av prototypen. For vår prototype har vi valgt å fokusere på disse kravene:

- 4.1.1.a Opprette bruker m/ epost-adresse
- 4.1.2.a Logge inn m/ epost-adresse
- 4.1.2.d Feilmelding ved feil brukernavn/passord
- 4.1.3 Legge inn parkeringsplass
- 4.1.5 Leie parkeringsplass
- 4.1.6 Søk av parkeringsplass
- 4.1.7 Brukerprofil
- 4.1.9 Betaling

For hvert av disse kravene vil de mest viktige delene bli implementert. Ergo de kravene som tidligst gir verdi for kunden. Det medfører at enkelte krav ikke blir implementert, blant annet administrator sine rolle i tjenesten og oversikten over inntjeninger i form av egnede grafer.

Brukere kan heller ikke se antall utilgjengelig og tilgjengelig plasser. Vi har implementert det slik at for en gitt adresse er det bare én parkeringsplass. Det medfører at firma sin rolle ikke er implementert.

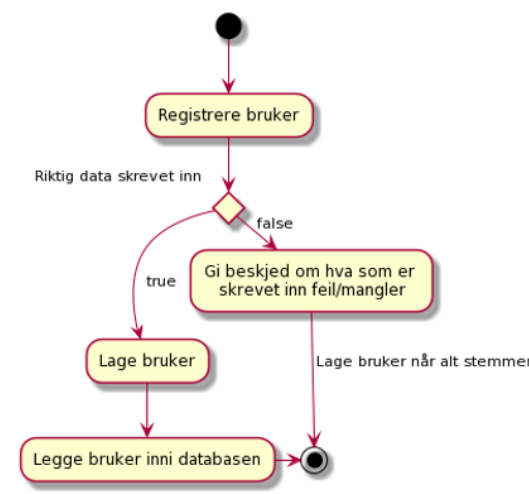
For krav om betaling er det hovedkravet som er i fokus. Det er da en demo av betalingsløsningen. Det medfører at betalingen er gjort med en funksjon i systemet vårt som bestemmer om betalingen blir godkjent eller ikke. Dette kan bli implementert senere med de eksterne avhengighet som vi har valgt.

Søk av parkeringsplass er ikke implementert som en kartløsning, men har heller ordnet det etter poststed.

7.1.2 Modellering av prototypen

Denne seksjonen vil vise modelleringen av prototypen. Det viser hvordan prototypens funksjoner samhandler. Det gjøres med forskjellige diagrammer som flytdiagram, aktivitetsdiagram, sekvensdiagram og lignende.

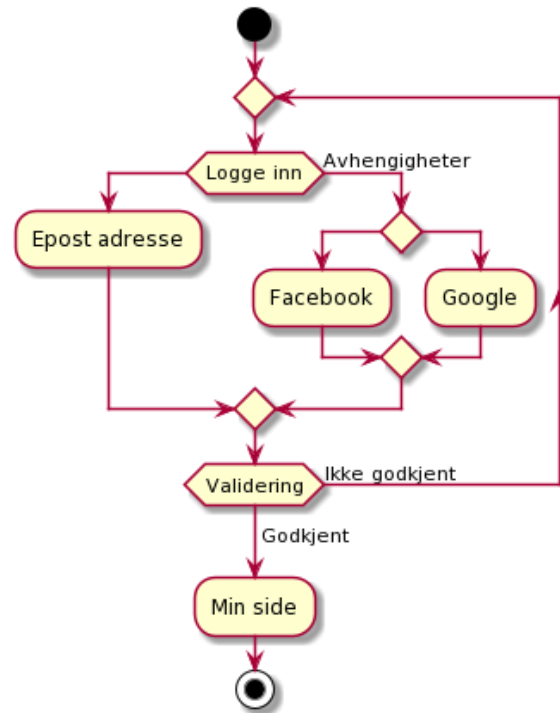
7.1.2.1 Opprette bruker modellering



Figur 4: Sekvensdiagram for opprette bruker

Figur 4 er en modellering av prototypen. Diagrammet viser flyten i hvordan man registrerer en bruker. Dette kommer fra kravet 4.1.1.a hvor brukeren kan opprette en profil med epost-adressen. Brukeren sender inn dataen sin via et skjema, systemet sjekker om dataen er skrevet inn riktig, om e-post og telefonnummer er gyldig, eller at noe data mangler. Hvis alt stemmer så blir profilen dannet og sendt videre til databasen. Ved feil får brukeren tilbakemelding, slik at han kan rette det opp.

7.1.2.2 Logge inn modellering

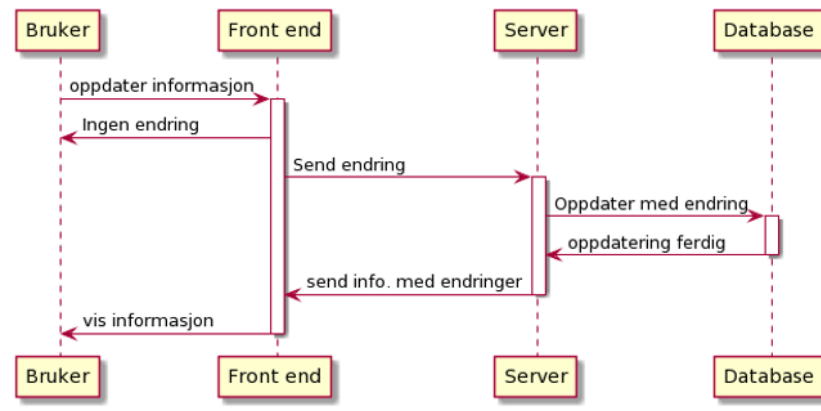


Figur 5: Aktivitetsdiagram for funksjonen å logge inn

Figur 5 er en modellering av systemet. Diagrammet viser hvordan flyten skal være ved å logge inn. Dette kommer fra kravet 4.1.2 hvor brukeren kan logge seg inn enten med Facebook, Google eller epost-adresse. Når brukeren logger inn vil funksjonen validere dataen. Ved feil vil brukeren få tilbakemelding og kan gjøre ett nytt forsøk.

I vår prototype er innlogging med hverken Facebook eller Google implementert, da e-post adresse innlogging gir tidligst verdi for kunden. Dette er beskrevet i seksjonen om vår funksjonalitet og avgrensning til systemet.

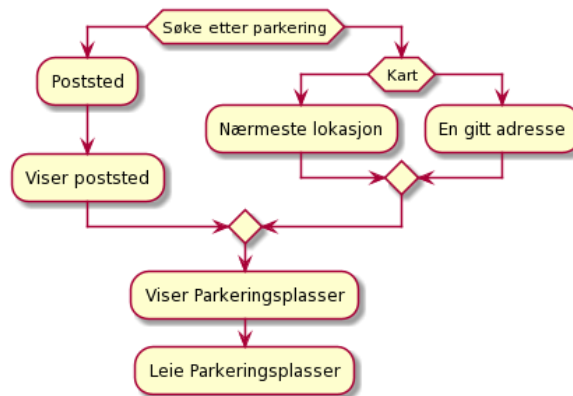
7.1.2.3 Oppdatere brukerinformasjon.



Figur 6: Sekvensdiagram for å endre informasjonen om brukeren lagret i systemet.

Figur 6 er modelleringen for systemet. Diagrammet viser hvordan man gjør endring i brukerprofil. Ved ingen nye endringer vill prosessen ikke gå videre. Ny oppdatert informasjon blir sendt til serveren for validering før det blir oppdatert i databasen. Tilbakemelding til brukeren vil bli gitt om oppgaven var vellykket eller ikke.

7.1.2.4 Søkning av parkeringsplass

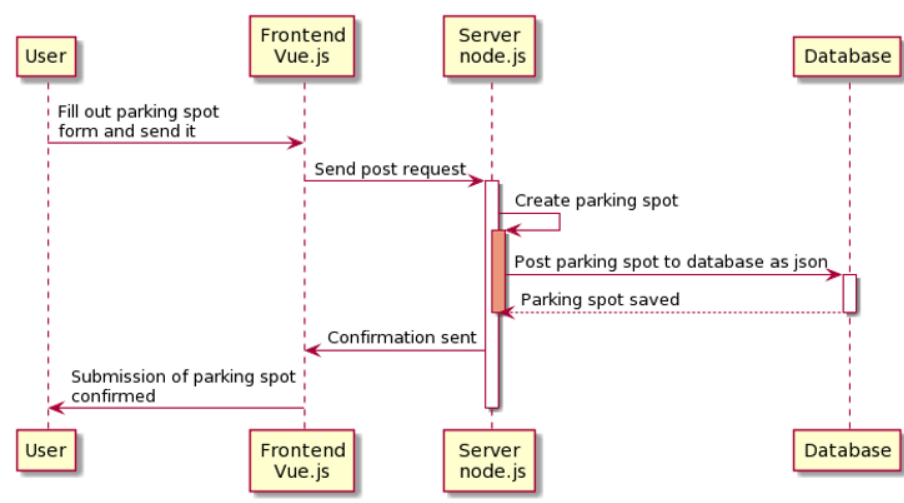


Figur 7: Aktivitetsdiagram for søk av parkeringsplass

Figur 7 er modellering for systemet. Diagrammet viser tenkt funksjon av søk parkeringplass fra krav 4.1.6. Databasen henter ut parkeringsplassene utifra poststedet eller lokasjonen i kartet som er valgt. Brukerne ser dermed parkeringsplassene i ønsket området. Utifra dette kan da brukeren leie en parkeringsplass.

Med en kartfunksjon, kan bruker finne nærmeste ledige parkeringsplasser til ønsket adresse. Vår prototype har fokusert på selve filtrering i form av poststed.

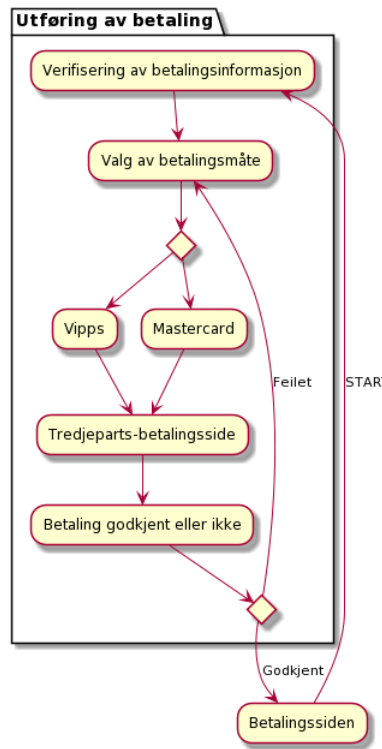
7.1.2.5 Legge ut parkeringsplass



Figur 8: Sekvensdiagram for legge ut en parkeringsplass.

Figur 8 er modelleringen for systemet. Det samsvarer med krav 4.1.4. Diagrammet presenterer en oversiktlig måte av logikken bak det å legge ut en parkeringsplass. Figuren viser hvordan en bruker sender inn skjema fra nettsiden. Data blir kontrollert i serveren, som vil oppdatere databasen og gi en tilbakemelding om prosessen er vellykket. Bruker vil så bli tatt videre til sin annonse for å se resultatet. Ved feil skal bruker få tilbakemelding om dette.

7.1.2.6 Betaling



Figur 9: Aktivitetsdiagram for betalingen

En viktig del av systemet vil være betaling for leie av parkeringsplass. Figur 9 viser hvordan betalingsdelen av systemet vil fungere. Dette kommer fra krav 4.1.8.

Brukeren vil først få en oversikt over bestilt produkt, dette innebærer dato, adresse og sum. Deretter kan ønsket betalingsform velges. Bruker vil så bli sendt til ønsket tredjepart for å utføre betalingen. Systemet vil få en tilbakemelding fra tredjepart om betaling var vellykket eller ei. Bruker vil få bekreftelse om betaling var vellykket og reservasjonen vil bli vist.

Prototypen er en enkel versjon av systemet hvor betalingensdelen er avgrenset. I vår versjon vil betaling bli bekreftet ved en funksjon som vil feile cirka 20% av gangene. En full versjon av betalingen vil kreve avtaler med Vipps og Mastercard.

7.1.3 Avhengigheter

Systemet består av to deler, en server og ett brukergrensesnitt (klient). De har igjen avhengigheter som er nødvendige, og noen som kun brukes ved utvikling. Under er det liste over de forskjellige avhengighetene og hva de brukes til.

Både server og klienten bruker **Node.js** for å kjøre. Node er et runtime-system som lar oss bruke JavaScript kode.

7.1.3.1 Server

Express	Ett Node.js rammeverk for nett applikasjoner. Brukes for å håndtere http forespørsler.
Cors	Cross-Origin Resource Sharing
Simple-json-db	Lar oss enkelt lese og skrive JSON til fil på disk.

For utvikling har i tillegg brukt

Babel	JavaScript compiler, som lar oss bruke nyere JavaScript kode.
Nodemon	Restarter node applikasjonen automatisk når filer endres.

For testing

Supertest	Brukes sammen med jest for å teste http
Jest	Test rammeverk for JavaScript

7.1.3.2 Klient

Vue	Frontend rammeverk
Axios	Brukes til http requests
Bulma	CSS rammeverk
Core-js	Lar oss bruke import istedenfor required i koden vår.
v-calendar	Kalender komponenten
vuex	Lar oss dele verdier mellom vue komponenter.
vue-router	Håndtere routing i vue.

For utvikling har i tillegg brukt

eslint Brukes for å analysere kode og finner feil.

Babel JavaScript compiler, som lar oss bruke nyere JavaScript kode.

7.1.4 Kjente svakheter og problemer

Prototypen er laget for å demonstrere noen av de sentrale funksjonene i tjenesten. Det er ikke ment å være et ferdig utviklet produkt. Under utviklingen har vi kommet borti enkelte problemstillinger som vi enten ikke har løst på en skikkelig måte, eller er ment å bli erstattet av andre komponenter i det ferdige produktet.

Prototypen har ikke implementert en database. Vi ønsker at det ferdige produktet skal benyttes seg av en dokumentdatabase. For å simulere dette på enklest mulig måte har vi valg å skrive JSON-objekter til enkelt-filer på disk.

Nedtrekksmeny til høyre i navigasjonen lukkes ikke etter at en ny side er trykket på. Dette er en konflikt mellom BULMA og Vue.

For at dato-velgeren skulle fungere på en god måte valgte vi å bruke v-calendar. Denne så lovende ut og var godt dokumentert. Men med vårt tenkte formål oppsto det noen problemer. Den lar oss ikke hente datoer fra databasen og sette disse som utilgjengelig. Dette medfører at flere brukere kan reservere plassen samme dag. Ved å bruke vue sin v-model funksjon på utilgjengelige datoer ender vi opp med å skape en konflikt med kalender komponenten. Et annen problem er når man velger dagens dato kan man reservere plassen fra ett tidligere klokkeslett enn det som er nå.

Når man er inne på betaling, eller velg dato siden, og deretter oppdaterer siden, vil systemet miste den plassen man holder på å reservere. Det er ikke laget en funksjon som skal håndtere denne feilen.

APIen kontrollerer ikke om dataen som bli sendt fra brukergrensesnittet faktisk er riktig. Det er kun lagt til «requierd» egenskaper på html elementene. Men dette kan endres i DOM, og dermed kan systemet enkelt bli utnyttet.

7.2 Eksempler

For å vise hvordan en bruker vil benytte seg av systemet så har vi hentet ut noen eksempler fra prototypen.

7.2.1 Profilside

Etter at en bruker har logget inn vil de komme til sin profil-side. Her får de en oversikt over sine plasser, og tidligere leieforhold.

Hei Ola Normann! 🤖

Din profil

Du er registrert med navnet **Ola Normann**, ønsker du å gjøre endringer i din profil, trykk på knappen.

Rediger profil

Figur 10: Viser profilnavn øverst på siden





På toppen av siden vil brukeren få mulighet til å videre redigere sine kontoopplysninger.

Tidligere leieforhold					
Adresse	Poststed	Fra	Til	Pris	Status
Rådyrvegen 8	Frogner	18.11.2020	20.11.2020	1200	Utgått
Sinselveien 10	Sinsen	20.11.2020	20.11.2020	150	Utgått
Sinselveien 10	Sinsen	19.11.2020	20.11.2020	800	Utgått
Stortinget 13	Oslo	26.11.2020	27.11.2020	1600	Aktiv

Finn en parkeringsplass

Figur 11: Oversikt over tidligere leieforhold

Det vil bli vist en oversikt over deres tidligere leieforhold. Med en status om parkeringen fortsatt er gyldig eller har utgått.

Dine parkeringsplasser				
Du har 2 plass(er)				
Totalt leieinntekter: 1350 .				
Se detaljert oversikt				
Adresse	Poststed	Timespris	Døgnpris	Rediger
Slottsplassen 1	Oslo	50	200	 
Sinselveien 10	Sinsen	50	400	 

Legg til parkeringsplass

Figur 12: Oversikt over egne plasser tilgjengelig for utleie

Nederst vil de få oversikt over parkeringsplasser de selv leier ut. Og med en

samlet inntjening av plassene sine.

7.2.2 Leie parkeringsplass

For å leie en plass, trykkes det på knappen «Finn parkeringsplass», som vil ta dem til siden hvor man får en oversikt over tilgjengelige plasser.

Finn parkering

Vis i område: Vis alle				
Adresse	Poststed	Timespris	Døgnpris	Videre
Slottsplassen 1	Oslo	50	200	→
Stortinget 13	Oslo	20	800	→
Rådnyrvegen 8	Frogner	50	400	→
Sinsenveien 10	Sinsen	50	400	→

Figur 13: Gir mulighet for å filtrere plasser etter område

Det aktuelle området man leter etter plass kan filtreres ved å bruke nedtrekksmenyen. Plasser merket med lysegrønn videre-knapp er brukeren sine egne plasser. Disse kan ikke velges.

Velg dato

Stortinget 13 er ledig følgende dager:

<

november 2020

>

M	T	O	T	F	L	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Tips:
For å velge én dag - trykk to ganger på samme dato.

Du har valgt:
Fra: 24-11-2020
Til: 26-11-2020
3 dager vil koste
💰 2400,-
Til betaling

Figur 14: Oversikt over tilgjengelige datoer

Når brukeren har funnet ønsket plass, velger man tilgjengelig dato i kalenderen. For å leie plass i flere dager velger man til og fra dato. Estimert kostnad for perioden vil bli vist på høyre side.

Velg dato

Stortinget 13 er ledig følgende dager:

<

november 2020

>

M	T	O	T	F	L	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Tips:

For å velge én dag - trykk to ganger på samme dato.

Du har valgt:

27-11-2020

8 time(r) vil koste

160,-

Til betaling

FRE. NOV 27 2020

10 : 00

→

FRE. NOV 27 2020

18 : 00

Figur 15: Mulighet for å reservere parkering med klokkeslett.

Hvis det er ønskelig å leie plass kun for noen timer velger man én dato. Det vil da være mulig å spesifisere fra og til klokkeslett de ønsker å leie plassen. Estimert kostnad for perioden vil bli vist på høyre side.

Ved å trykke på «Til betaling» går man videre til betalingssiden.

Betaling

for 8 time(r), på Stortinget 13 i Oslo.

🕒 27-11-2020 → 27-11-2020 🕒

Å betale:

160,-

Velg betalingsmetode

Vipps

Mastercard

Eller gå tilbake for å endre dato

Tilbake

Figur 16: Valg av betalingsmåte

Her blir man vist en oppsummering av bestillingen, og mulighet for å velge betalingsmåte.

7.3 Installasjon av systemet

For å installere og bruke systemet kreves Node og en nettleser. Vi har under utvikling brukt [Google Chrome](#), og Node versjon v14.13.0. Systemet har også blitt testet og virker med seneste versjon v15.2.1. For å installere Node se fremgangsmåte på deres [hjemmeside](#).

Vedlagt produktdokumentasjonen ligger en mappe: «Software-Engineering» som inneholder kildekoden til prototypen. For å fortsette installasjonen av nødvendige avhengigheter trenger vi å åpne terminalen og naviger inn i mappen.

```
$ cd Software-Engineering
```

7.3.1 Server

Systemet er delt opp i to deler, en *client* og en *server*. Vi skal nå gå igjennom installasjonen av serveren.

I terminalen, naviger inn i *server* mappen, og installer avhengigheter:

```
$ cd server  
$ npm install
```

`npm` vil nå laste ned alle avhengigheter til server programmet, og legge disse i mappen *node_modules*.

Når avhengigheten er ferdig lastet ned er server siden av systemet klart til bruk.

7.3.2 Client

Naviger terminalen inn i *client* mappen og installer avhengigheter:

```
$ cd ../client\  
$ npm install
```

`npm` vil nå laste ned alle avhengigheter til klient programmet, og legge disse i mappen *node_modules*.

Når avhengigheten er ferdig lastet ned er klient-siden av systemet klart til bruk.

7.4 Bruk

Når avhengigheter til server og klienten er lastet ned og installert er systemet klart til bruk. **Merk:** Både *server* og *client* må kjøre for at systemet skal virke som det skal!

7.4.1 Server

Vi åpner en ny terminal og navigerer først til *server* mappen. Og starter serveren ved å bruke kommandoen:

```
$ npm run start
```

Du vil nå få en tilbakemelding i terminalen som ser slik ut, systemet er klart til bruk på <http://localhost:5000>. Ikke lukk dette terminalvinduet for da avsluttes programmet.

```
> share-a-spot@1.0.0 start
> node index.js

Server running on port: http://localhost:5000
```

MERK! Pass på at det ikke er noe annet på maskinen din som allerede kjører på denne porten, dette vil medføre at systemet ikke starter!

Hvis du åpner <http://localhost:5000> i nettleseren din, så skal du se meldingen:

```
{
  message: "Share-A-Spot Server - Up and running"
}
```

7.4.2 Client

For å starte brukergrensesnittet, åpner vi ett nytt terminalvindu og navigerer til *client* mappen. Vi starter vue sitt utviklingsmiljø ved:

```
$ npm run serve
```

Dette vil bygge systemet og gjøre det klart til bruk.

```
App running at:
- Local:   http://localhost:8080/
- Network: http://192.168.1.XXX:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Når det er ferdig vil vi få en tilbakemelding i terminalen om at systemet er tilgjengelig på <http://localhost:8080>. Ikke lukk dette terminalvinduet for da avsluttes programmet.

Når begge tjenestene kjører, er systemet klart til å brukes. Du kan nå åpne nettleseren din og navigere til <http://localhost:8080> hvor brukergrensesnittet er.

7.5 Testing

Systemet består av et brukergrensesnitt og en API som håndterer forretningslogikk. Det er mot APIen vi har fokusert på å skrive testene. Det er i all hovedsak fordi brukergrensesnittet kun viser data som er sendt fra APIen, og det skal være minimalt med logikk som ligger her.

Vi har skrevet tester mot alle funksjonene i APIen, men ut ifra kildekode oversikten, så er det et par spesielle punkter hvor vi ikke har gjenskapt enkelte feil-tilfeller. Dette gjelder spesielle tilfeller i noen av funksjonene våre, hvor vi sender en feilmelding tilbake hvis databasen ikke klarer å skrive. Dette scenarioet har vi ikke skrevet en test for. Men logikken for å håndtere det er der.

En annen test som ikke er fullstendig, er betalingen. Her er det skrevet kode som gjør at betalingen feiler cirka 20% av gangene. Vi fant ingen løsning for å teste denne funksjonen på en god måte, så det vi gjorde istedenfor var å forvente at systemet ikke feiler, altså ikke gi «500 Internal Server Error».

7.5.1 Oversikt over filer

Testene som tilhører funksjoner forbundet med innlogging finnes i filen `login.test.js`.

- | | | |
|---------|--|---|
| 4.1.2.a | | Logge inn m/ epost-adresse |
| 4.1.2.d | | Feilmelding ved feil brukernavn/passord |

Brukere skal kunne opprette en konto i tjenesten, og ha en form for brukerprofil med noe funksjonalitet. Tester til følgende krav og funksjoner finnes i filen

`user.test.js`

- 4.1.1.a | Opprette bruker m/ epost-adresse
- 4.1.7 | Brukerprofil

Tester til funksjoner relatert til krav om å legge inn og leie plass finnes i filen `spots.test.js`

- 4.1.4 | Legge inn parkeringsplass
- 4.1.5 | Leie parkeringsplass
- 4.1.6 | Søk av parkeringsplass
- 4.1.8 | Betaling

7.5.2 Kjøre tester

For å kjøre testene åpner vi en terminal og navigerer til `server` mappen. Vi bruker `Jest` som er et test rammeverk for JavaScript. Den vil gå igjennom alle mappene og lete etter filer merket med `<filnavn>.test.js`. Alle filer som ligger i mappen `__tests__` vil automatisk også bli kjørt. Alle testene til programmet vårt ligger i denne mappen.

Når vi er i `server` mappen starter vi testene ved:

```
$ npm run test
```

Dette vil kun gi oss en tilbakemelding om at alle testene var vellykket, eller hvis noen har feilet så vil den gi en tilbakemelding om det. For en mer utdypende rapport bruker vi `--verbose`. Merk, vi trenger `--` før `--verbose`.

```
$ npm run test -- --verbose
```

Her får vi nå en oversikt over alle testene som blir kjørt med navnene deres. Disse navnene er også beskrivende slik at det er tydelig hva de tester. Se utdrag fra eksempel:

```
PASS  __tests__/login.test.js
GET / *
  OK / - Should respond with 403 - Forbidden (56 ms)
  OK / * (any) - Should respond with 403 - Forbidden (6 ms)
POST /login
  OK Valid username and password should respond 200 (28 ms)
  OK Invalid username and password should respond 400 (6 ms)
```

For få se hvor mye av koden testene faktisk dekker kan vi bruke en annen kommando `--coverage`.

```
$ npm run test -- --coverage
```

Dette vil skape en oversiktlig rapport over hvilke filer som blir testet og hvor mye av koden i filene som blir dekket. Også hvis det er noen deler av koden vi ikke har inkludert. I tillegg lager jest en flott rapport som kan åpnes i nettleseren. Her kan man gå inn i de forskjellige filene, og detaljert se koden, i tillegg til hvor testene eventuelt ikke dekker. Denne rapporten finnes under `coverage/lcov-report/index.html`.