Software Engineering 2020 - Gruppe 2

Prosjektdokumentasjon «ParkX»

Joakim Jensen, Mathias Ernest Thomas Nygaard Jarbekk, Michal Kowalski og Thomas Tennøy Johannessen

Høsten 2020

Innhold

Introduksjon

Prototypen løser oppgaven ved å ta for seg de viktigste kravene i et MVP format og ser på hvordan programmet kunne utvikles til å senere kunne lede til en versjon av produktet som ville være klart for distribusjon. Du vil senere i dokumentasjonen kunne lese om krav-spesifikasjonen og se hvordan den endelige MVPen samsvarer med disse kravene og hvilke krav vi ville videreført i systemet.

Som beskrevet i delen over så er en viktig del av dette systemet den grafiske framvisning av denne delingsøkonomiske ideen til oppdragsgiver. Dette har vi fått til ved bruk at rammeverket Kivy for Python. I dette grafiske rammeverket kan vi oppfylle alle kravene som relaterer til hva brukerne skal se og gjøre i produktet. Imens vår bruk av Python v3.7 lar oss kode bl.a. alt av kode funksjoner og systemer for å feste funksjoner til dette grafiske rammeverket, pris kalkulatorer og lister over alle tilgjengelige parkeringer. Til slutt så har vi laget en løsning for å fake en integrert form for betalingssystem, for å vise funksjonaliteten i dette. Vi har ikke integrert mot faktiske tredjepartssystemer.

I prototypen vår så har vi måttet gjøre noen valg med tanke på hva som skulle være med i MVP'en vår og begrensningene dette ville gi i systemet vårt. Den første begrensningen vi bestemte oss for å gjøre var at vi utelat en administrator rolle i programmet vårt. Vi har også bestemt oss for å utelate et innlogging/brukere-system. Blant kravene våre så ønsket vi at man skulle kunne reservere en plass fram i tid, men denne funksjonen ble ikke med i MVP'en. Det er heller ikke noe betalingssystem integrert i denne prototypen. Alt av lagring av verdier og informasjon i systemet vårt gjøres i et repository i denne versjonen av systemet. I en ferdig versjon vil dette bli integrert fremfor å fake funksjonaliteten. Dette er noen eksempler av begrensningene og antagelser vi måtte gjøre da vi bygget denne prototypen. Det er en bedre beskrivelse av disse senere i dokumentasjonen.

Problemstilling:

Oppdraget gitt av oppdragsgiver forteller, hvordan flere bedrifter kaster seg ut i konseptet om delingsøkonomi. I dette tilfelle er en oppstartsbedrift opptatt av å lage slik løsning med hensyn på parkering. Oppstartsbedriften vil lage en løsning hvor både bedrifter og privatpersoner kan legge ut egne parkeringsplasser for leie, når de ikke selv er i bruk.

Sluttbrukeren skal da være i stand til å se, reservere, leie og betale for leie av en parkeringsplass.

Det vil altså si en del av problemet er å vise noe grafisk for brukeren, siden bruker må kunne se parkeringsplasser. Vi antar her at det betyr brukeren også må kunne se detaljer om en parkeringsplass. Når en bruker må kunne se detaljer om parkeringsplassen, skal det også være mulig å legge til detaljer om parkeringsplassen. Problemet forklarer også både privatpersoner og bedrifter skal kunne legge ut parkeringsplasser. For privat personer kan det tenkes seg at de legger ut en parkeringsplass ut, mens en bedrift kanskje kan tenkes ønsker legge ut flere parkeringsplasser. Et problem blir da å håndtere, leing av mange parkeringsplasser for bedrifter som eier større parkeringsplasser, og få parkeringsplasser for privatpersoner. Når en sluttbruker må kunne reserverer en parkeringsplass, involverer det også at bruker må kunne velge hvilken parkeringsplass som brukeren vil leie et sted i fremtiden. Bruker må da også være i stand å velge hvilke parkeringsplasser, som skal reserveres og til hvilken tid. Sist må bruker også kunne betale for leing av parkeringsplass, det involvere systemet klarer å utregne hvor mye hver person skylder, utfra hvor lang tid en parkeringsplass er blitt leid. I tillegg må et betalingssystem også innføres, for overføring betalingene mellom kontoer.

Domenet:

Domenet for prosjektet handler om utleie av parkeringsplasser for biler. Dette gjelder både små parkeringsplasser, med helt ned til 1 plass eid av privatpersoner og helt opp til store parkeringsplasser med flere hundrede parkeringsplasser eid av bedrifter. Dermed blir brukere også privatpersoner, som både har lyst til leie en parkeringsplass eller leie ut parkeringsplassen sin brukere av systemet. Samt også større bedrifter som vil leie ut parkeringsplassene sine eller leie flere parkeringsplasser, for ansatte som eksempel skal på konferanse. ParkX blir også en mulighet for privatpersoner med ekstra parkeringsplass å tjene ekstra penger på siden.

ParkX har flere forskjellige brukssituasjoner bestemt av om man er en bedrift eller privat person. Som privatperson skal du kanskje på jobb, men finner at det ikke er noen ledige parkeringsplasser der du vanligvis parkerer. ParkX kan dermed bli en nødløsning for de som trenger en rask parkeringsplass. Men ParkX er også nyttig for de som planlegger frem i tid og

kanskje trenger en parkeringsplass i helga når det er spesielt mange folk i byen eller skal på større arrangementer. For større bedrifter kan vår løsning bli en ekstra inntekt, ved å leie ut parkeringsplasser i de tider hvor det ikke er i bruk så mange parkeringsplasser. ParkX blir også et kraftfullt verktøy, når flere medarbeidere må møte opp på et spesielt sted, og på en sikker måte kan sikre seg parkeringsplasser.

ParkX henvender seg til alle typer miljøer og steder en bil lovlig kan parkere. Så lenge det finns en adresse får det gitte sted skal det være mulig å lage en parkeringsplass der. Dvs. for spesielle arrangementer, hvor eksempelvis en gressplen midlertidig blir gjort om til en parkeringsplass, kan denne også ligges til ParkX så lenge en adresse finns. ParkX kommer mest sannsynlig til å henvende seg mer til store byer hvor plassen er liten, og det er større etterspørsel etter parkeringsplasser.

Brukere av systemet

Her følger noen bruekrhistorier skrevet i starten av planleggingsfasen. Disse brukerhistoriene har tatt utgangspunkt i det ferdige systemet. Brukerhistoriene er brukt for å finne krav til systemet og få en overenstemmelse om hvordan systemet skal fungere og se ut. Siden prototypen i systemet kun er en MVP er det flere av funksjonene beskrevet her som ikke er med, men vi har fokusert på å bygge opp kjernesystemet, og få frem de viktigste funksjonene i systemet. Vurderingen i forhold til hva som er tatt med og ikke finnes i estimat delen under «Krav».

Brukerhistorie 1:

Katrine er en bruker i ParkX systemet. Hun vil gjerne bruke appen til å leie ut sin parkeringsplass, når hun selv ikke skal bruke den.

Katrine åpner appen. Først blir hun møt av en meny, som gir henne valgene, min profil, dine parkeringsplasser, leide parkeringsplasser og lei en parkeringsplass. Når Katrine trykker på dine parkeringsplasser, blir hun sendt til en ny side.

Denne siden presentere seg i form av en liste med bokser, som inneholder navn, pris per time, antall, adresse og bilde av parkeringsplasser, som Katrine har lagt ut.

Når Katrine vil aktivere en parkeringsplass trykker hun på en av disse parkeringsplasser bokser i listen. Dette vil åpne ny side, som spør hva hun vil gjøre, med denne parkeringsplass. Aktivere, endre, eller slette. Katrine trykker aktiver som åpne muligheten for å skrive inn et tidsintervall, hvor parkeringsplassen kan være leiet.

Katrine setter inn tidsintervallet og klikker på knappen aktiver. Deretter blir hun sendt tilbake til listen over parkeringsplassene hennes. I toppen har hun tre valg se log, parkeringsplasser og aktive parkeringsplasser.

Når Katrine trykker på aktive parkeringsplasser, vil hun få opp en liste med de aktive parkeringsplasser. Hun kan i tillegg se en leie status om parkeringsplassen er i leie eller ikke, og en rød deaktiveringsknapp.

Brukerhistorie 2:

Katrine kommer tidligere hjem fra jobb og trenger en av de parkeringsplassene hun har satt til leie. Hun går til aktive parkeringsplasser og trykker deaktiver på knappen.

Hvis ingen har leiet parkeringsplassen vil den deaktivere uten problemer.

Hvis den er i leie, vil en melding poppe opp om at parkeringsplassen er i leie, og leieren må kontaktes, med en knapp se leiers profil.

Parkeringsplassen blir deaktivert, men vil fortsatt telle penger fra leier inntil, leier avbryter leing av parkeringsplassen. Leier vil også motta beskjed om at parkeringsplassen er blitt deaktivert.

Brukerhistorie 3:

Går inn på vindu siden til legg inn parkering. Skjemaet ber han skrive inn navn, adresse, bilde, pris, antall parkeringsplasser, detaljer. I tillegg til om parkeringsplassen har lader, via en check box. I toppen av venstre hjørne er det en knapp som sier avbryt. Om Peter klikker denne, kommer han tilbake til vis parkerings liste vinduet. Nede i høyre hjørne på siden, er det en knapp som sier legg til. Om Peter klikker denne og ikke har fylt ut alle feltene, blir bedt om å fylle dem ut. Hvis alle felter er blitt lagt inn, blir han sendt til listen over sine parkeringsplasser.

Brukerhistorie 4:

Olav er på vei til Oslo, og har planer om å bruke ParkX til å leie en parkeringsplass. Olav åpner appen og blir møtt av en meny. Her trykker han på lei en parkeringsplass. Han blir da møtt av en søke meny som ber han skrive inn en adresse eller by. Olav skriver inn Oslo og trykker knappen søk. Deretter kommer det opp en liste over parkeringsplasser i Oslo.

Hver boks i denne liste inneholder adresse, antall, pris per time, leie intervall og bilde. Olav trykker på en av disse boksene for å åpne parkeringsplassen.

En ny scene blir åpnet som viser parkeringsplassen i detalj. Nederst er det en knapp, som sier lei. Når Olav trykker på denne knappen blir et vindu åpnet som ber om et tidsintervall han vil leie parkeringsplassen i. Dette tidsintervallet blir sjekket opp mot detaljer om parkeringsplassen.

Parkeringsplass detalj siden inneholder også en knapp som guider Olav til utleierens profilside slik at han kan se dens omtale.

Brukerhistorie 5:

Olav går til leide parkeringsplasser og får en liste over parkeringsplasser som han leier nå. Her kan klikke på parkeringsplass og får valget å deaktivere eller utvide lånetiden. Trykker han deaktiver, blir han spurt om han virkelig vil deaktivere.

Deaktivere blir boksen fjernet fra denne liste.

Brukerhistorie 6:

Jeg er en Bruker. Jeg er ute og kjører og må finne en parkeringsplass i nærheten av der jeg skal. Jeg tar frem telefonen og åpner ParkX appen. I denne appen vil jeg kunne legge inn adressen jeg skal til og finne parkeringer i nærheten. Når jeg finner en parkering jeg liker så kan jeg reservere den for en kort tid, slik at jeg vet hvor jeg kan parkere når jeg kommer fram. Når jeg har reservert en plass får jeg en popup om jeg ønsker å legge den inn som destinasjonen i maps/waze(eller lignende). Når jeg kommer fram så vil jeg få et varsel om å sette tid for parkeringen, men denne kan jeg enten utvide eller minske i etterkant. Jeg kan nå gå ifra bilen min uten å tenke for mye på tiden siden jeg vil få en push-notifikasjon når parkeringen snart er over.

Brukerhistorie 7:

Som en leier ønsker jeg å kunne leie en parkeringsplass nå, slik at jeg kan få parkert bilen min. Dette vil jeg gjøre ved at jeg åpner applikasjonen på min pc/smart telefon. Jeg skriver inn min epostadresse og passord og logger inn. Her vil jeg også ha mulighet til å opprette ny bruker og få nytt passord hvis jeg har glemt dette. Videre går jeg inn i applikasjonen og får opp listen med parkeringer. Jeg har og et søkefelt øverst slik at jeg kan søke på bestemte adresser eller byer. Kun ledige parkeringer skal komme opp. Når jeg trykker på en parkering jeg ønsker, får jeg opp informasjonen om plassen og jeg har en knapp der det står «Parker her». Trykker jeg på denne må jeg skrive inn registreringsnummer på bilen som skal stå der, og parkeringen starter.

Når jeg er ferdig å parkere går jeg inn i applikasjonen og trykker øverst på den som viser aktiv parkering over alle ledige. Trykker jeg på denne får jeg spørsmål om jeg vil stoppe parkeringen. Klikker jeg ja får jeg opp beskjeden «takk for at du brukte ParkX til å parkere», og parkeringen er avsluttet og pengene trukket. Oppe til venstre har jeg en meny der jeg kan se tidlige parkeringer, starte med utleie og endre kontoinformasjon.

Brukerhistorie 8:

Som en utleier ønsker jeg å kunne leie ut parkeringsplassen min, slik at jeg kan tjene penger på den når jeg ikke bruker den. Dette vil jeg gjøre ved at jeg åpner applikasjonen på min pc/smart telefon. Jeg skriver inn min epostadresse og passord og logger inn. Her vil jeg også ha mulighet til å opprette ny bruker og få nytt passord hvis jeg har glemt dette. Videre går jeg inn i applikasjonen og får opp listen med parkeringer. Jeg trykker da på menyen øverst til venstre og får opp alternativene der jeg kan se tidlige parkeringer, utleie og endre kontoinformasjon. Jeg trykker meg inn på «Utleie» og får opp et skjema der jeg må fylle ut all info om parkeringsplassen jeg vil leie ut. Jeg har og mulighet for å laste opp bilder av den. Deretter trykker jeg på knappen «lei ut plassen min». Jeg kommer da tilbake til utleie siden i menyen der parkeringsplassen min nå har kommet opp. Dette er listen over mine utleide plasser. Jeg har og mulighet her til å gå inn på plassen og få endre detaljer, deaktivere plassen

dersom jeg trenger den selv (så lenge den er ledig), eller slette plassen så fort den blir ledig. Her kan jeg også se hvem som står på plassen til enhver tid. Oppe til høyre har jeg og et pluss tegn for å opprette nye parkeringsplasser.

Brukerhistorie 9:

Anne skal på en konsert i Trondheim. Hun har bestemt seg for å ta bilen, men trenger et sted hvor bilen kan stå parkert gjennom konserten. Anne har en applikasjon som heter ParkX, åpner den og logger seg inn med brukernavn og passord. Hun søker seg fram til området hvor konserten skal foregå, og ser en hel haug med tilbud om ledige parkeringsplasser. Hun går et steg videre og sorterer både på pris, og dato hun ønsker å leie plassen i. Anne blir fornøyd med en billig plass som kan leies i noen timer og velger den, hun blir bedt om å velge mellom betal nå, eller å legge til parkeringsplassen i en handlekurv og betale senere Anne velger å betale med en gang og gjør seg klar til å betale. Hun tar frem kortet og betaler. Etter at Anne har betalt får hun en e-mail med bekreftelse av ordre. Hun blir nå sendt tilbake til hovedsiden, hvor reservasjonen vises, og om betalingen har blitt gjennomført.

Avhengigheter i systemet

Eksterne avhengigheter:

I vår applikasjon, har vi valgt å lage en «fake» for databasen, klassen kalt ListRepository. I realiteten hadde denne klassen egentlig hatt en eller annen form for kommunikasjon med en database gjennom en server, hvor fra den kan hente, sende og endre data til og fra. Dette eksterne systemet vil da ha ansvar for å oppbevare de forskjellige dataene om brukere, parkeringsplasser, reservasjoner og hvor mye hver bruker må betale. Klassen vil også ha del ansvar i å sørge for at brukere ikke kan be om data som de ikke bør se.

Når det gjelder betalinger trengs det et eksternt betalingssystem som kommuniserer med applikasjonen, når bruker må betale for leing av parkeringsplass. Det vil altså si dette eksterne system har ansvar for overføring av penger mellom mange forskjellige kontoer. Eksempel på et slikt system kunne være Vipps eller Klarna.

I det endelige systemet ville man også hatt avhengigheter mot kartsystemet vi hadde brukt. Programmet ville trolig kjørt på en ekstern server og dataene hadde blitt lagret i en database eksternt.

Interne avhengigheter:

Kivy er vårt rammeverk for å lage det grafiske utseende. Kivy har i tillegg noen avhengigheter som også må lastes ned før Kivy kan brukes. Dette er kivy-dep.angle, kivy-deps.glew, kivy-deps.sdl2. Disse 3 må være nedlastet i miljøet for at Kivy skal fungerer korrekt. I tillegg brukes det også et rammeverk for testing kalt pytest. Tester er avhengig av dette rammeverk at testene skal kunne kjøre korrekt. Noen tester har vi også vært nødt til å test om funksjoner blir aktivert i det hele tatt, til dette har vi brukt biblioteket mock. Noen tester blir da også avhengig av dette bibliotek, for å kunne kjøre. For å kunne teste aspektet rundt tid bruker vi også rammeverket Freezegun, som brukes til å teste mot tid.

Alle disse blir installert automatisk i miljøet ditt med setup.py filen. For at disse skal fungere må man pr produksjonsdato for prototypen bruke Python 3.7. Les mer om oppsett av miljø og setup.py i «kom i gang» seksjonen.

Krav

Funksjonelle-krav:

ParkX.Innlogging

01. En bruker skal identifiseres i systemet ved bruk av e-post adresse.

02. Brukeren skal ha mulighet til å resette passordet sitt gjennom en lenke de får tilsendt på epost, ved å trykke på glemt passord knapp i innloggingsmenyen.

03. En ny bruker skal kunne registrere seg inn i systemet ved å fylle ut et registreringsskjema.

ParkX.Personvern

01. Brukere skal kunne fjerne seg fra hele tjenesten om de ønsker det, ved å trykke på slett bruker knappen i brukerinnstillinger.

02. Administrator skal kunne utestenge brukere fra tjenesten, ved å bruke menyen som bare administratorer har tilgang til.

ParkX.Tilbakemeldingssystem

01 Brukere skal kunne skrive klager, ved å bruke eget skjema tilegnet klager.

02. Brukere skal kunne se klager, ved å se klagehistorikken i en egen klagemeny.

02.01 Brukere skal se klager de skriver, ved at klagene de får opp i klagehistorikken har en «skrevet» tagg ved siden av tittelen.

02.02 Brukere skal se klager de mottar, ved at klagene de får opp i klagehistorikken har en rød «mottatt» tagg ved siden av tittelen.

03. Brukere skal kunne rapportere andre brukere ved brudd av reglementet, ved hjelp av klagesystemet.

03.01. Brudd på reglementet skal skrives i klage ved å oppgi brukernavnet til gjerningspersonen, tidspunktet og en kort beskrivelse i et eget skjema.

04. Utleiere skal kunne blokkere visse brukere fra å leie plassen igjen (Viktig hvis utleieren ikke var fornøyd med kunden)

05.Brukere skal kunne gi hverandre rating, ved å bruke ratingsystemet etter enhver leie

05.01. Leier skal kunne gi parkeringsplassen den utleide en terningkast basert på hvor fornøyd den var med plassen.

05.02. Utleier skal kunne gi leieren et terningkast basert på hvor fornøyd utleieren var med den som leide plassen.

ParkX.Leie-plassen

01. Leiere skal kunne leie en parkeringsplass, ved å få opp en liste over plasser

//Implementert
//Testes i test_ControllerIntegration.py -> test_returnsListFromRepositoryProperly()
//Testes også i
//test_controllerSendsRequestToChangeParkingPlaceStatusAndSavesStartDateCorrectly()

02. Leiere skal kunne reservere parkeringsplass frem i tid. (Dette vil være hensiktsmessig for de som planlegger fremtidige besøk)

03. Leiere skal kunne utvide parkeringstid på en reservasjon, i ledig tidsrom.

04. Leieren skal kunne sette seg på venteliste dersom ønsket parkeringsplass er reservert.

04.01 Leieren skal få en mulighet til å få varsel når parkeringsplassen blir ledig igjen.

05. En leier skal kunne starte en tilgjengelig parkering med en gang, og betale med en gang den stanses

//Implementert
//Testes i test_ControllerIntegration.py ->
//test_controllerReturnsCalculatedPriceForParkingBasedOnTimePassedSinceParkingStartCor-
//rectly()

05.01. En leier skal kunne stoppe en parkering de leier ved å trykke på «stopp parkering» knappen ved siden av parkeringen under «mine parkeringer» menyen.

//Implementert
//Testes i test_ControllerIntegration.py ->
//test_controllerSendsRequestToChangeParkingPlaceStatusAndSavesStartDateCorre
//ctly()

06. Utleier skal kunne tilby langtidsleie, ved å markere parkeringsplassen som «til månedlig leie».

07. En leier skal kunne se detaljene på en parkeringsplass før den leier plassen

//Implementert

//Testes i test_ControllerIntegration.py -> test_controllerGetsSpecificParkingPlaceWithId()

ParkX.Utleie-plassen

01. Utleier skal kunne legge til en ny parkeringsplass, med detaljer angående miljøet og stedet som plassen befinner seg i, ved å fylle ut et «registrer ny parkeringsplass» skjema i hovedmenyen til utleier.

//Implementert

//Testes i test_ControllerIntegration.py

// -> test_receivesDictionaryFromUserSavesItInRepositoryAndCreatesThe-

//ObjectProperly()

02. Utleier skal kunne slette egne plasser, ved å trykke på «fjern tilbud» knappen som befinner seg ved siden av enhver parkeringsplass i «min profil» visningsmenyen.

02.01. Utleier skal kunne slette egne plasser individuelt, ved å bare trykke på fjern tilbud knappen på en enkel parkeringsplass.

//Implementert

//Testes i test_ControllerIntegration.py ->

//test_receivesIdFromUserAndDeletesCorrectParkingPlaceObject()

02.02 Utleier skal kunne slette egne plasser i mengder, ved å velge flere plasser for sletting samtidig.

03. Utleier skal kunne endre detaljene på en parkeringsplass, ved å trykke på endre detaljer knappen som befinner seg i den detaljerte visningen av en parkeringsplass.

//Implementert

//Testes i test_ControllerIntegration.py ->

//test_controllerChangesParkingPlaceAttributesProperly()

04. Utleier skal kunne midlertidig deaktivere plassen sin for utleie (Dette er nødvendig dersom utleieren blir nødt til å bruke plassen selv)

05.Utleier skal kunne få en oversikt over alle plassen som den har lagt ut til utleie, ved å gå inn på dens profil-visning.

06. Utleier skal kunne se hvem som leier plassen nå, ved å gå inn i utleie-historikk menyen.

06.01 Utleier skal kunne se hvem som har leid plassen i fortiden, ved å gå inn i utleie-historikk menyen

ParkX.Søking

01. Leier skal kunne søke på en plass i et ønsket geografisk område, ved å bruke filter by eller avstand fra nåværende posisjon

02. Leier skal kunne søke på en plass i et ønsket miljø, ved å bruke checkboksene ved siden av søketreffene. (Et miljø er altså om parkeringsplassen befinner seg inne/ute, eller om det er med lader, lys o.s.v)

ParkX.Historikk

01. Leier skal kunne få en oversikt over aktive og tidligere parkeringer, ved å gå inn i historikk menyen.

02. Brukere skal kunne se eldre betalinger, ved å gå inn i betalingshistorikk i historikk menyen.

ParkX.Administrator

01. En Administrator skal ha alle rettigheter som andre brukere har.

02. En Administrator skal ha alle nødvendige rettigheter til å opprettholde orden og rettferdighet i systemet, ved å logge inn som en spesiell bruker.

ParkX.Økonomi

01. Brukere skal legge inn kort for automatisk trekk, ved å legge til kortet i min profil menyen.

02. Brukere skal kunne trekkes månedlig for langtidsleie, ved å trekkes automatisk fra kortet.

02.01 Brukere skal kunne få tilsendt faktura, dersom de ikke har lagt til kortet sitt for automatisk trekk.

03. Utleiere skal få månedlige utbetalinger i form av totalsum, samlet opp gjennom hele måneden, ved at systemet sender ut pengene en dag i måneden.

04. Brukere skal kunne få en liste med alle gjenstående betalinger som de har å betale ned, ved å gå inn på min profil vinduet.

//Implementert

//Testes i ControllerIntegration.py -> test_can_get_list_with_all_payments()

05. Brukere skal kunne betale gjenstående betalinger, ved å gå inn på min profil vinduet og trykke på «betal alle gjenstående parkeringer»

//Implementert

//Testes i ControllerIntegration.py ->

//test_can_empty_all_payments_from_list_if_acceptedPaymentDetails_is_true()

//+ ->test_can_not_empty_all_payments_from_list_if_acceptedPaymentDetails_is_false()

ParkX.Validering

01. Brukere skal få tilbakemelding dersom de fyller ut et skjema feil.

01.01 Brukere skal få tilbakemelding dersom et skjema de fyller ut inneholder minst et tomt felt.

//Implementert

//Testes i ControllerIntegration.py ->

//controllerRaisesValueExceptionIfInputFieldsWhereIntExpectedIncludesLetters()

01.02 Brukere skal få tilbakemelding dersom et skjema de fyller ut inneholder bokstaver steder hvor bare tall forventes.

//Implementert

//Testes i ControllerIntegration.py ->

//test_controllerRaisesUserWarningIfInputFieldIsEmpty()

02. Brukere skal få tilbakemelding dersom feil oppstår under betaling.

    //Implementert

    //Testes i ControllerIntegration.py ->

    //test_can_get_accepted_payment_details_state_from_Payment()

Ikke-Funksjonelle krav:

ParkX.Data

01. Systemet skal samle data, ved å sende statistikk til eieren av systemet, en gang i måneden.

02. Systemet skal lagre data i MYSQL.

ParkX.System

01. Systemet skal skrives i Python 3.7.

02. Systemet skal kodes på engelsk, men brukergrensesnittet skal skrives på Norsk.

03. Systemet skal kunne kjøres på datamaskin, nett og Android, ved å bruke kivy rammeverket.

04. Systemet skal maks bruke 3 sekunder per scene bytte.

05. Systemet skal ikke bruke mer enn 1GB RAM minne.

06. Systemet skal kreve at en bruker identifiseres seg før den får lov til å bruke systemet.

ParkX.Sikkerhet

01. Systemet skal kryptere alt dataen, som sendes ut av systemet.

02. Systemet skal ta vare på personvern, ved å følge GDPR regelverket.

ParkX.Tilgjengelighet

01. Den nettbaserte delen av systemet skal være universelt utformet og tilgjengeliggjort etter WCAG-2.0 standarden.

02. Systemet skal vær enkelt å bruke, slik at brukere trenger hjelp med systemet gjennomsnittlig en gang per 6 måneder.

ParkX.Driftsikkerhet

01. Systemet skal ha en feilfrekvens (ROCOF) på mindre enn 1/100 000, ved hjelp av gode feilbehandlinger.
02. Systemet skal være tilgjengelig (AVAIL) 99.5% av tiden.
03. Systemet skal håndtere feil, slik at den ikke bruker mer enn 5 minutter på å restarte og bli brukbart igjen.

Estimat av krav:

Navn i kravliste:	Nr i krav-liste:	Under-krav nr:	Feature:	Utviklings størrelse	Verdi:	Sum:	Proto-type?
Innlogging	1		Identifiseres ved bruk av e-post adresse	MEDIUM	X-LARGE	6	NEI
	2		resette passordet ved bruk av lenke i epost	MEDIUM	LARGE	2	NEI
	3		En ny bruker registrere seg med registreringsskjema	MEDIUM	X-LARGE	6	NEI
Personvern	1		Brukere skal kunne fjernes slette med slett-bruker knappen	SMALL	LARGE	3	NEI
	2		Administrator skal kunne fjerne brukere fra tjenesten	MEDIUM	MEDIUM	0	NEI
Tilbakemld.sys.	1		Brukere skal kunne skrive klager	MEDIUM	MEDIUM	0	NEI
	2	0	Brukere skal kunne se klager	X-LARGE	MEDIUM	-6	NEI
		1	Brukere skal se klager de skriver	LARGE	MEDIUM	-5	NEI
		2	Brukere skal se klager de mottar	LARGE	MEDIUM	-5	NEI
	3	0	Brukere skal kunne rapportere andre brukere ved brudd av reglementet	MEDIUM	MEDIUM	0	NEI
		1	Ved brudd på reglementet skal brukernavnet til gjerningspersonen, tidspunktet og en kort beskrivelse nevnes i eget skjema	MEDIUM	MEDIUM	0	NEI
	4		Utleiere kunne blokkere brukere fra å leie plassen igjen	MEDIUM	SMALL	-1	NEI
	5	0	Brukere skal kunne gi hverandre rating	MEDIUM	MEDIUM	0	NEI
		1	Leier gi parkeringsplassen terningkast	X-LARGE	MEDIUM	-6	NEI
		2	Utleier gi leieren en vurdering	X-LARGE	MEDIUM	-6	NEI
Leie-plassen	1		Leiere skal kunne leie en parkeringsplass, fra en liste	LARGE	X-LARGE	4	JA

	2		Leiere skal kunne reservere parkeringsplass frem i tid.	MEDIUM	LARGE	2	NEI
	3		Leiere skal kunne utvide en reservasjon	MEDIUM	LARGE	2	NEI
	4	0	Leieren skal kunne sette seg på venteliste	LARGE	MEDIUM	-2	NEI
		1	Leieren skal få en mulighet til å få varsel når parkeringsplassen blir ledig igjen	MEDIUM	MEDIUM	0	NEI
	5	0	En leier skal kunne starte en tilgjengelig parkering med en gang	SMALL	X-LARGE	7	JA
		1	En leier skal kunne stoppe en parkering de leier ved å trykke på «stopp parkering» knappen	SMALL	X-LARGE	7	JA
	6		Leier skal kunne bruke langtidsleie	MEDIUM	MEDIUM	0	NEI
	7		En leier skal kunne se detaljene på en parkeringsplass før den leier plassen	SMALL	X-LARGE	7	JA
Utleie-plassen	1		Utleier skal kunne legge til en ny parkeringsplass, med detaljer	X-LARGE	X-LARGE	0	JA
	2	0	Utleier skal kunne slette egne plasser	X-LARGE	X-LARGE	0	JA
		1	Utleier skal kunne slette egne plasser individuelt	MEDIUM	LARGE	2	JA
		2	Utleier skal kunne slette egne plasser i mengder	MEDIUM	LARGE	2	NEI
	3		Utleier skal kunne endre detaljene på en parkeringsplass	SMALL	LARGE	3	JA
	4		Utleier skal kunne midlertidig deaktivere plassen sin for utleie	SMALL	LARGE	3	NEI
	5		Utleier skal kunne få en oversikt over alle plassen som den har lagt ut til utleie	LARGE	X-LARGE	4	JA
	6	0	Utleier skal kunne se om plassen leies nå	LARGE	LARGE	0	JA
		1	Utleier skal kunne se hvem som har leid plassen i fortiden	LARGE	LARGE	0	NEI
Søking	1		Leier skal kunne søke på en plass i et ønsket geografisk område	LARGE	LARGE	0	NEI
	2		Leier skal kunne søke på en plass i et ønsket miljø	LARGE	MEDIUM	-2	NEI
Historikk	1		Leier skal kunne få en oversikt over aktive og tidligere parkeringer	MEDIUM	X-LARGE	6	NEI
	2		Brukere skal kunne se eldre betalinger	MEDIUM	LARGE	2	NEI
Administrator	1		En Administrator skal ha alle rettigheter som andre brukere har.	LARGE	MEDIUM	-2	NEI
	2		En Administrator skal ha alle nødvendige rettigheter til å opprettholde orden og rettferdighet i systemet	MEDIUM	MEDIUM	0	NEI
Økonomi	1		Brukere skal legge inn kort for automatisk trekk	MEDIUM	X-LARGE	6	NEI
	2	0	Brukere skal kunne trekkes månedlig for langtidsleie	MEDIUM	LARGE	2	NEI

		1	Brukere skal kunne få tilsendt faktura, dersom de ikke har lagt til kortet sitt for automatisk trekk.	SMALL	MEDIUM	2	NEI
	3		Utleiere skal få månedlige utbetalinger i form av totalsum	MEDIUM	MEDIUM	0	NEI
	4		Ikke-godkjente betalinger skal legges til i liste for senere betaling	MEDIUM	MEDIUM	0	JA
	5		Brukere skal kunne betale gjenstående betalinger, ved å gå inn på min profil og trykke «betal alle gjenstående parkeringer»	MEDIUM	MEDIUM	0	JA
Validering	1	0	Brukere skal få tilbakemelding dersom de fyller ut et skjema feil.	SMALL	LARGE	3	JA
		1	Brukere skal få tilbakemelding dersom et skjema de fyller ut inneholder minst et tomt felt.	SMALL	LARGE	3	JA
		2	Bruker skal få tilbakemelding dersom den har bokstaver i et felt der det forventes tall	SMALL	LARGE	3	JA
	2		Bruker skal få tilbakemelding om betalingen ikke blir godkjent	MEDIUM	LARGE	2	JA

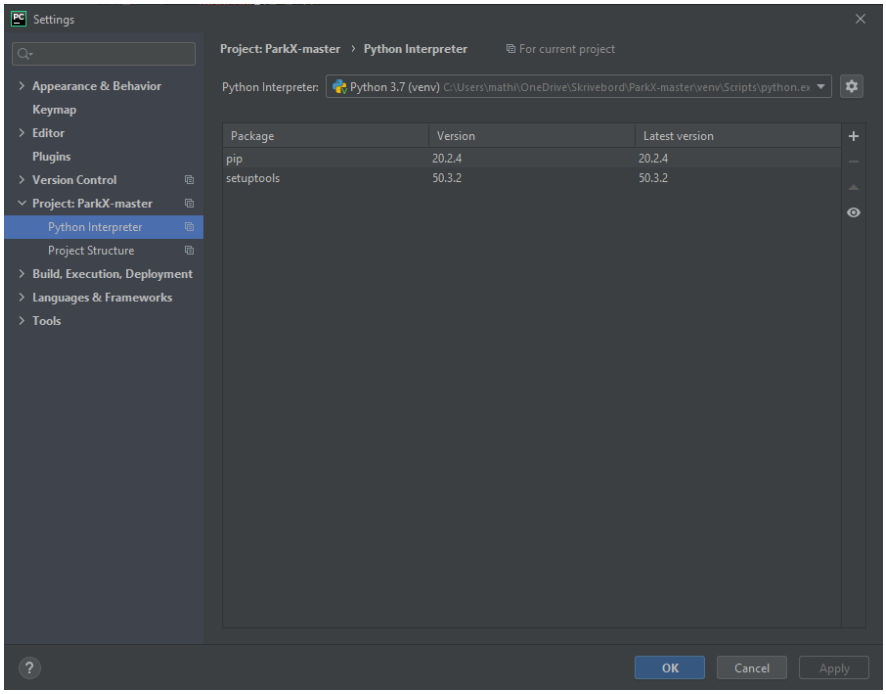
Kom i gang med systemet

For å kunne kjøre programmet må du ha et Python IDE (Vi anbefaler PyCharm community edition (https://www.jetbrains.com/pycharm/download/#section=windows)) installert på PC'en din. Dette IDE'et må ha in interpreter for Python 3.7 installert siden GUI rammeverket vi bruker dessverre ikke fungerer på en nyere versjon av Python enda.

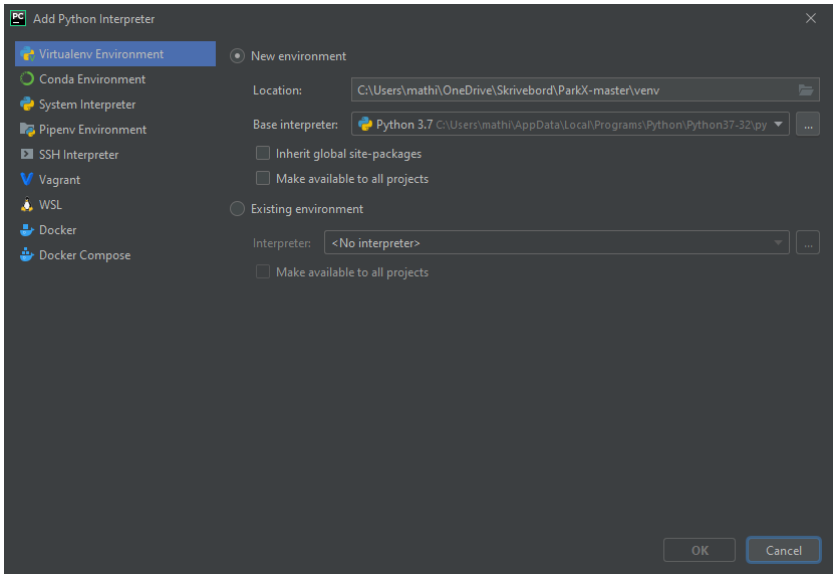
Når alt dette er på plass kan du åpne prosjekt-filen. Siden dette er en .zip-fil må du så extracte denne mappen til et sted på maskinen din hvor du lett kan finne fram til den (skrivebordet ditt for eksempel). Etter at du har gjort dette så kan du navigere til den mappen på maskinen din og høyreklikke på den og åpne med det IDE'et du har installert. **NB! Her er det viktig at du åpner hele mappen «ParkX» og ikke den undermappen som heter «parkx_kode» som rotmappe.**

Det er også viktig at du har ett helt nytt utviklingsmiljø satt opp på maskinen din, det gjør du på denne måten:

1. Åpne instillinger i Pycharm ved å trykke «Ctrl+Alt+S», dette vil åpne et vindu som ser slik ut:

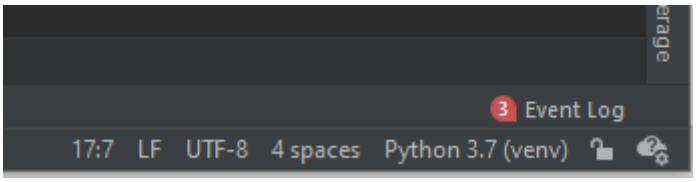


2. på venstre side så må du klikke der det står «Project:ParkX» og så på «Python Interpreter»
3. I høyresiden av vinduet har du nå en dropdown-meny med et tannhjul vedsiden av. Trykk på dette tannhjulet og trykk «Add»

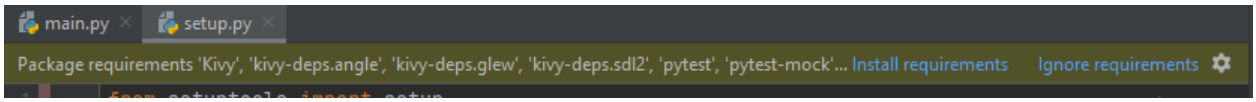


4. Det åpner vinduet som er illustrert over her, klikk på «Virtualenv Environment» og så pass på at du har en Python-versjon 3.5-3.7 valgt i «Base Interpreter» siden det er dette som bestemmer hvilken versjon av Python du kjører koden vår i.
5. Du kan så klikke «OK» nederst i høyre hjørne av det vinduet.

6. Neste skritt er så å passe på at du har valgt dette miljøet i IDE'et, det gjør du nederst til høyre i programmet:



7. Der det står Python 3.7 (venv) kan du klikke for å bytte mellom miljøer, når du laster ditt nye miljø så vil du få opp denne popup'en i IDE'et:



8. Klikk så på «Install requirements» og la dem installere. Etter at dette er gjort så har du sørget for at du har ett «ferskt» miljø og at du har installert de nødvendige pluginsene du trenger for å kjøre systemet.

Nå som du har fått åpnet programvaren vår så kan du nå på venstresiden i klikke deg inn på mappen som heter "parkx_kode" og derunder finne en fil som heter "main.py". Hvis du høyreklikker på den så vil få opp en liten meny hvor du kan klikke på "Run 'main.py'". Dette starter programmet.

Når programvaren er oppe og kjører vil du få opp ett vindu på skjermen med knappene "Leier", "Utleier" og "Min Profil". Du kan nå klikke deg gjennom programmet og teste ut de forskjellige funksjonene i programvaren som å legge til nye parkeringsplasser som en "utleier" og så leie dem som en "leier" og se hvordan prisen blir beregnet basert på hvor lenge du leier den for. Du kan også som en utleier fjerne eller endre på detaljene til parkeringsplassene du har lagt ut. Prototypen har flere funksjoner enn dette som blir beskrevet senere i dokumentasjonen.

Til slutt så vil det være naturlig å ville kjøre testene våre for å være sikker på at alt fungerer som det skal i koden vår, og det gjør du veldig enkelt ved å høyreklikke på "test" mappen i det venstre feltet av IDE'et og klikke run. Eller så kan du åpne "test" mappen i IDE'et og høyreklikke å kjøre alle test filene hver for seg. Du kan også klikke på den grønne pilen med ett skjold ved siden av for å se hvor god dekning vi har på testene våre, altså hvor mye av koden vår som vi har kjørt tester på.

Beskrivelse av systemet

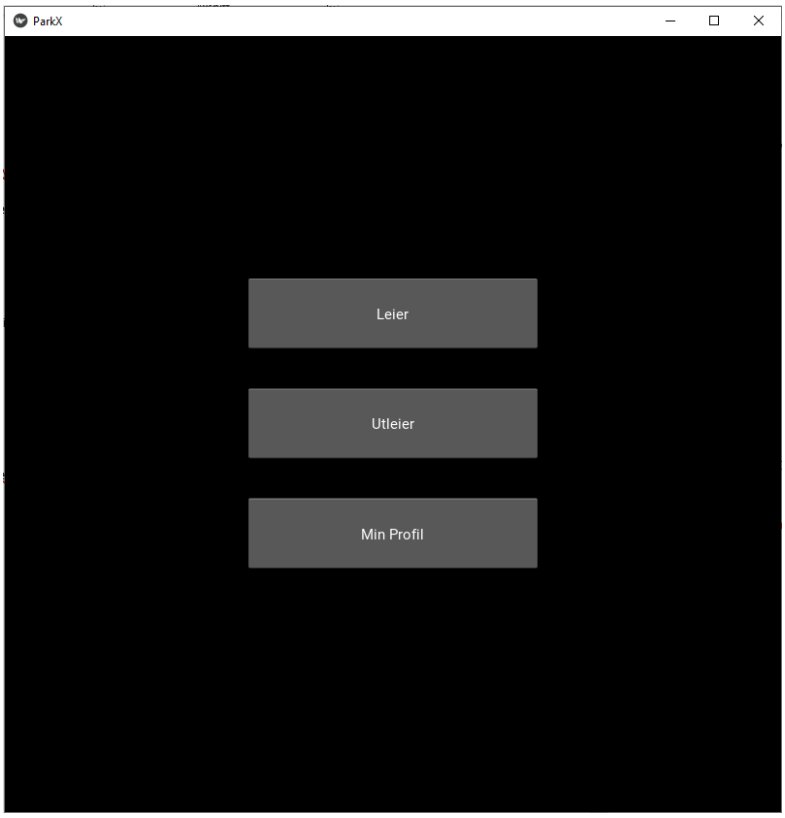
ParkX består av flere individuelle deler kalt klasser som kommunisere med hverandre. Hver del har ansvaret for en spesiell del av det systemet gjør. Sammen sørger alle klassene for å lage et større system.

Prototypens komponenter:

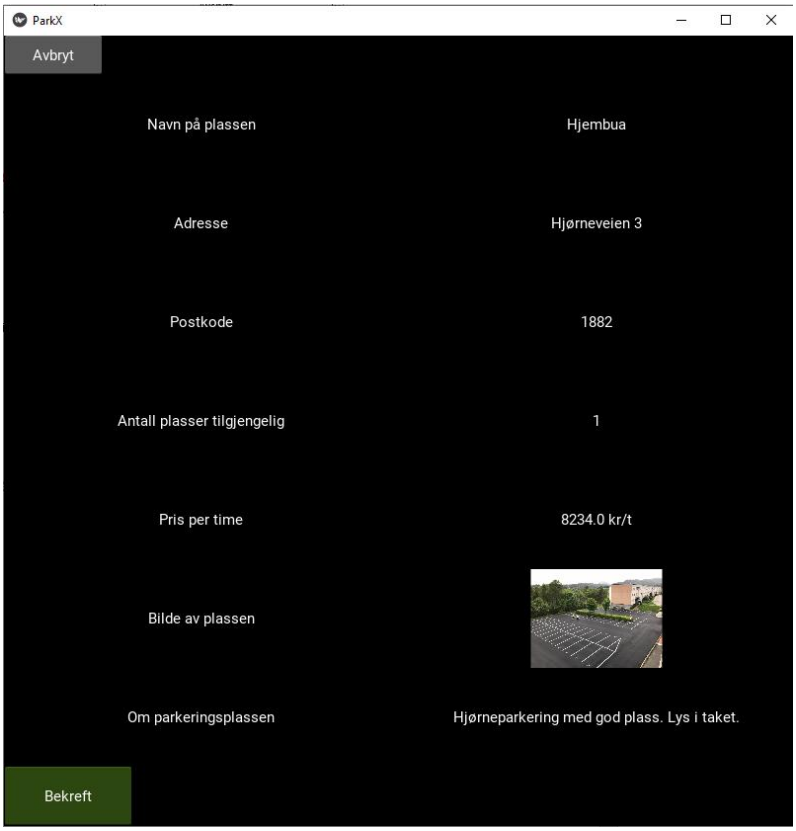
Parkingplace er klassen som holder styr på lagring av den data som hører til hvert enkle parkeringsplass. Det vil altså si Parkingplace eksempelvis: holder styr på data som adresse, pris_per_time, antall og om parkeringsplassen er i bruk. Hver parkeringsplass har også en id som brukes hyppig av andre klasser for å få takk i den riktige parkeringsplass. I tillegg holder også Parkingplace funksjonalitet til å manipulere denne data. Eksempelvis, kan Parkingplace oppdaterer sin egen status til si om parkering er blitt startet av en annen bruker, eller sagt på en annen måte, om parkeringsplassen er i bruk. I tillegg til å endre statusen til motsatt vei, altså at parkeringsplassen ikke lengere er i bruk. Hovedjobben til Parkingplace er holde den individuelle data for hver parkeringsplass, som enkelt kan trekkes ut og brukes av andre klasser.

ListRepository er en klasse som tar seg av jobben med å behandle alle parkeringsplasser. Klassen har en liste som skal holde alle parkeringsplasser som skal vises for brukeren. ListRepository holder i tillegg en del funksjoner for å operere på denne listen som holder alle parkeringsplasser. Her kommer bl.a. sletting, endring og laging av nye parkeringsplasser. Alle disse tar bruk av parkeringsplassens id for å kunne, eksempelvis: slette den riktige parkeringsplass. Når parkeringsplasser lages er det viktig og noterer seg at informasjonen som kommer inn for å kunne lage en parkeringsplass kommer i form av en dictionary. Dictionary er en samling av data. I tillegg finnes det også en del funksjonalitet for å trekke ut en eller hele listen med parkeringsplasser. For å trekke ut en spesifikk brukes også id til parkeringsplassen. ListRepository er en fake database, siden vi ikke kan koble en riktig database til prototypen. ListRepository hadde derfor i en mer realistisk implementasjon ikke holdt funksjonalitet for å endre direkte på parkeringsplasser, men hellere holdt funksjonalitet for å kontakte en ekstern database, også fortelle hvordan den skal gi eller endre data. Hovedjobben til ListRepository er og oppfører seg som en fake database, som kan hente og opererer på parkeringsplasser.

Gui er vår frontend klasse. Klassen extender BoxLayout som er en klasse fra avhengigheten Kivy. Denne hjelper med å enkelt kunne lage et grafisk brukergrensesnitt. Gui klassen består hovedsakelig av funksjoner som legger inn elementer som knapper, bilder og tekst i vinduet. Når bruker klikker på en knapp, blir andre funksjoner i Gui aktivert, disse er kalt handlers. Det kan skje flere forskjellige ting når en bruker klikker på en knapp. Noen knapper som eksempelvis i startsiden av programmet vil, få vinduet til å bytte side(figur1). Dette gjøres ved å fjerne elementer som er på siden og bytte dem ut med nye. Når nye sider må bygges opp basert på parkeringsplassen eller mange parkeringsplasser, vil Gui aktiverer funksjoner i klassen kalt ParkingController (mer om denne under). Disse funksjoner vil ofte aktiverer andre funksjoner i ListRepository som gjør at ParkingController får en parkeringsplass eller en liste med parkeringsplasser. Deretter vil ParkingController returnerer disse til Gui, som da kan bygge opp siden utfra denne data. Andre knapper som «Bekreft» knappen som ses på siden hvor man bekrefter at man vil leie parkeringen(figur2). Vil i tillegg til å bytte side også aktiverer funksjoner i ParkingController som forteller ListRepository hvordan den eksempelvis skal endre en parkeringsplass (mer om hvordan funksjonene henger sammen i hvordan systemet henger sammen). Gui klassens hovedoppgave er og presenterer informasjonen om parkeringsplasser til brukeren, men skal også fungerer som startpunktet for brukerens input.



Figur 1:Utklipp av hovedmeny-siden til ParkX



Figur 2: Bilde av detaljert visning av plassen etter klikket på lei parkering

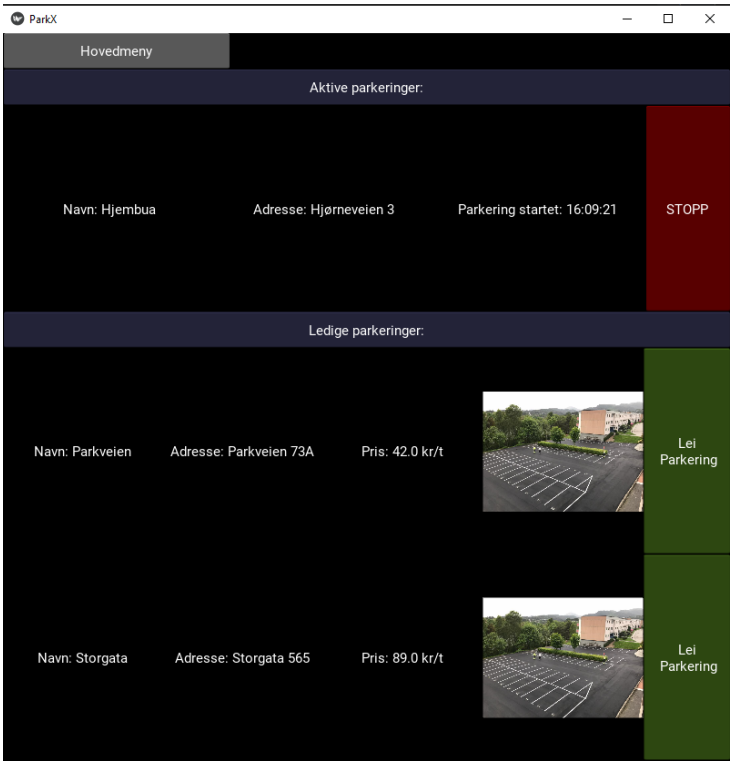
ParkingController klassen er et mellom lag mellom ListRepository og Gui klassen. Den har som oppgave sørge for data flyter enkelt mellom Gui og ListRepository. ParkingController spiller derfor en viktig rolle når Gui ber om data som den vil presenterer for brukeren, eller hvis bruker lager en nye parkeringsplass som må legges til ListRepository. ParkingController har derfor både funksjoner for å kunne kalle ListRepository for å kunne hente parkeringsplasser, samt og leverer informasjon for å kunne lage parkeringsplasser. Disse funksjoner blir oftest aktivert av at Gui kaller dem, som en reaksjon til at bruker gjør noe. ParkingController har også som oppgave å sørge for at informasjon som kommer inn når en parkeringsplass detaljer må endres at de kommer inn på riktig form. Når nye parkeringsplasser skal lages, sørger også kontrolleren for å tildele en ny id til den nye parkeringsplassen. ParkingController hovedoppgave er og fungerer som et mellomlag som kan se gjennom dataen som sendes til ListRepository, samt at den klarer å hente parkeringsplasser fra ListRepository og sende dem til Gui.

Sist, men ikke minst har vi også en klasse som styrer betaling. Klassen er skal mest fungerer som en stub, altså en klasse som representere noe funksjonalitet som egentlig en tredjepart program skal styre. Klassen styrer funksjonalitet for automatisk betaling, men har også funksjonalitet for å lage nye betalinger samt å betale dem. Payment klassen har også ansvar for å sørge for om godkjente betalingsmidler er aktivert eller ikke. Egentlig bør et tredjepart program holde styr på dette, så vi har satt to knapper hvor man kan slå av og på godkjente betalingsmidler for å se hvordan programmet oppfører seg med og uten betalingsmidler.

Hvordan komponentene henger sammen

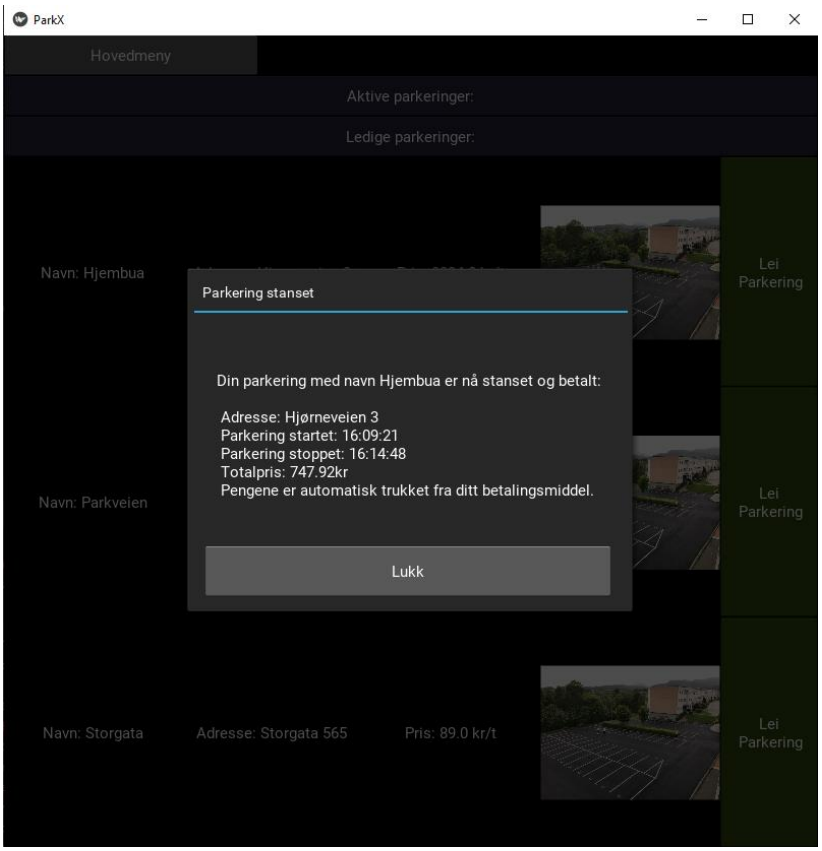
GUI / hvordan sidene går til hver side:

Bruker vil alltid starte på start siden kalt hovedmeny. Her vil bruker få tre velg leier, utleier og min profil (se figur 1). Hvis bruker klikker på leier, blir bruker sendt til leier listen. Her vises en liste med parkeringsplasser (se figur 3). Dette er parkeringsplasser bruker kan leie og allerede er i gang med å leie.

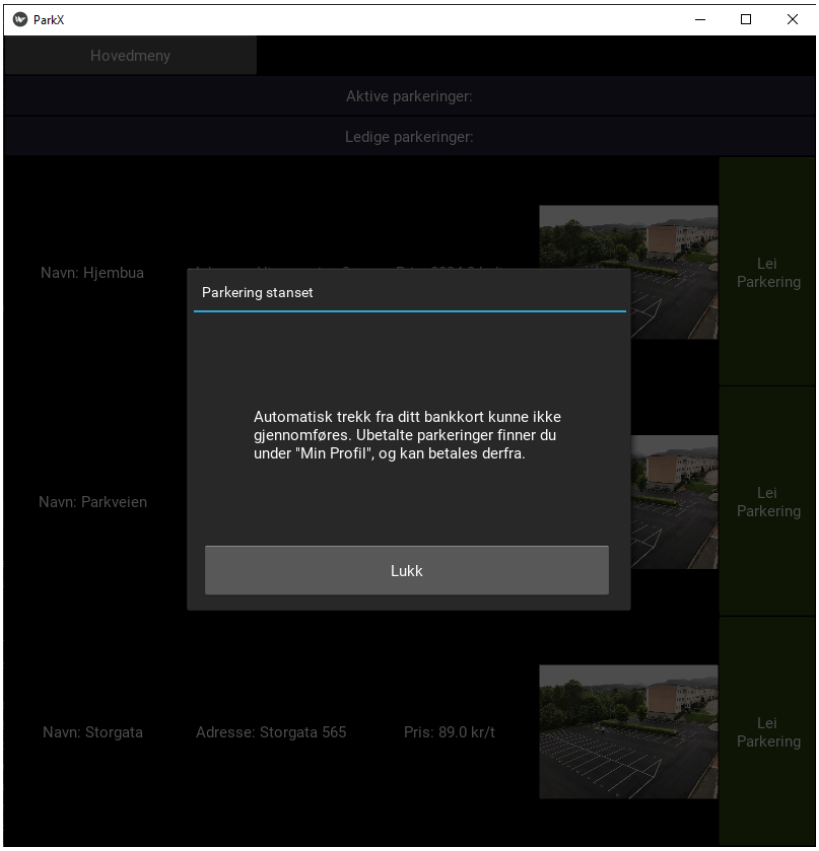


Figur 3: leiers liste med parkeringsplasser. To ledige plasser, og en aktiv parkering.

Her fra har bruker 3 valg, hovedmeny, stopp eller lei parkering. Trykker bruker hovedmeny, går bruker tilbake til hovedmeny. Trykker bruker stopp parkeringsplass vil gi en popup. Denne er forskjellig på en betingelse om godkjente betalingsmidler er gitt, les mer om dette under Betaling. Hvis godkjente betalingsmidler er gitt vil en popup vise informasjon om den nå betalte parkering (se figur 4). Hvis godkjente betalingsmidler ikke er gitt vil en pop-up vises som forteller godkjente betalingsmidler ikke er aktivert (se figur 5).



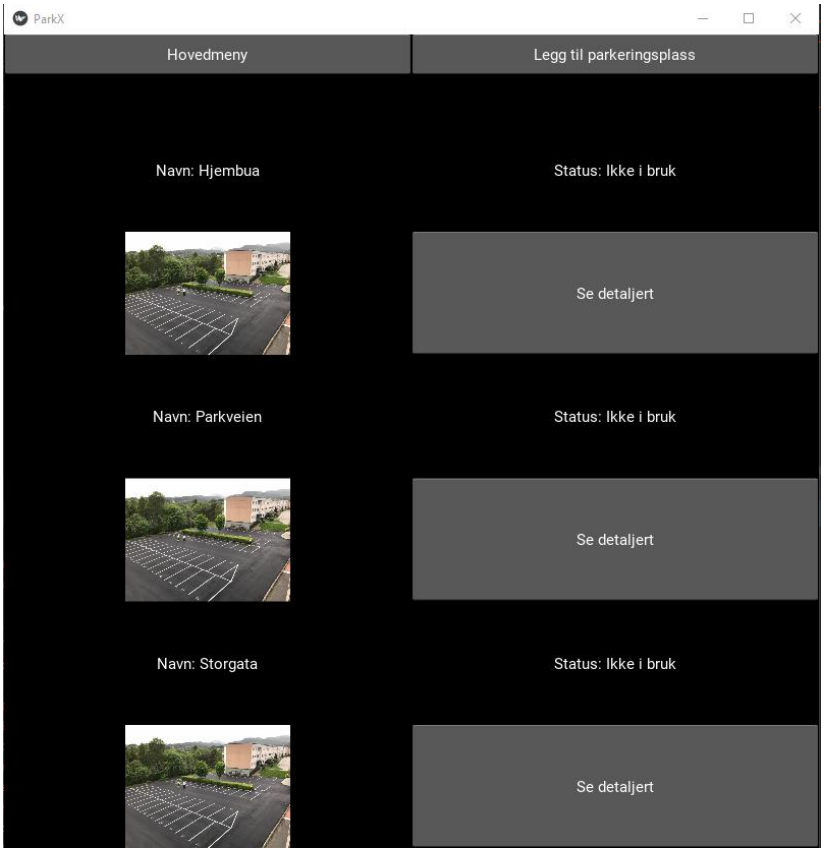
Figur 4: pop-up av informasjon om avsluttet parkering



Figur 5: pop-up info om ikke gitte betalingsmidler

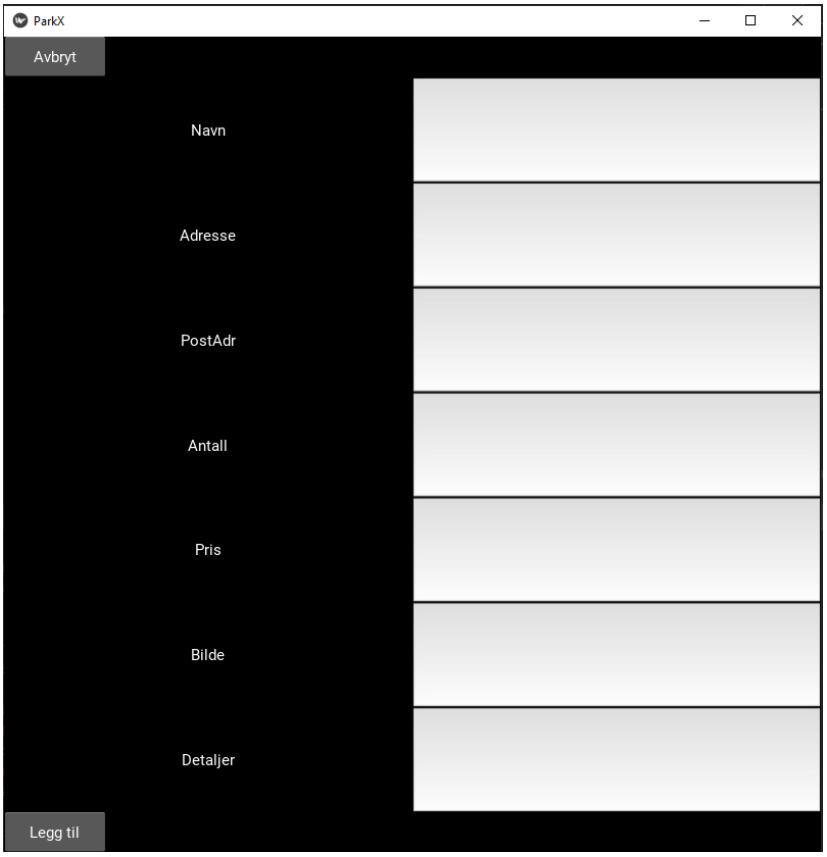
Hvis bruker klikker lei parkering, vil en de vises parkering i detalj (se figur 2). Her får bruker valget å trykke avbryt som sender bruker tilbake til listen med parkeringsplasser. Men bruker får også valget å trykke bekreft, hvor heretter brukes sendes tilbake til listen, med parkeringsplasser. Den parkeringsplassen som ble trykket bekreft på skal ligges til listen over aktive parkeringer som leies av brukeren.

I hovedmenyen får bruker også valget å trykke utleier. Her vil bruker også sendt til en liste med parkeringsplasser (se figur 6). Dette er listen over parkeringsplasser som brukeren selv har lagt ut. Her har brukeren også valget å tilbake til hovedmenyen ved å klikke hovedmeny. Bruker kan tillegg også legge til en parkeringsplass ved å klikke på den knappen som indikerer det. Heretter vil bruker bli sendt til et skjema, hvor den må fylle inn informasjon om den nye parkeringsplassen (se figur 7). Her kan bruker avbryte, som sender bruker tilbake til listen igjen. Bruker kan også klikke legg til. Hvis all informasjon er skrevet inn og korrekt vil bruker bli sendt tilbake til listen, hvis ikke vil en pop-up feilmelding si at noe gikk galt (se figur 8).

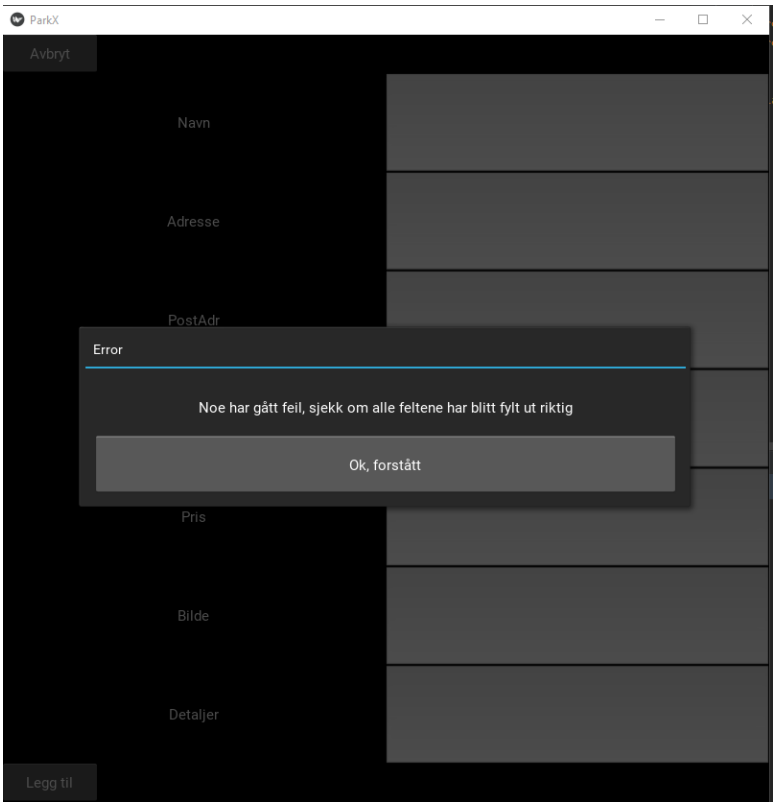


Figur 6: utleiers liste med parkeringsplasser

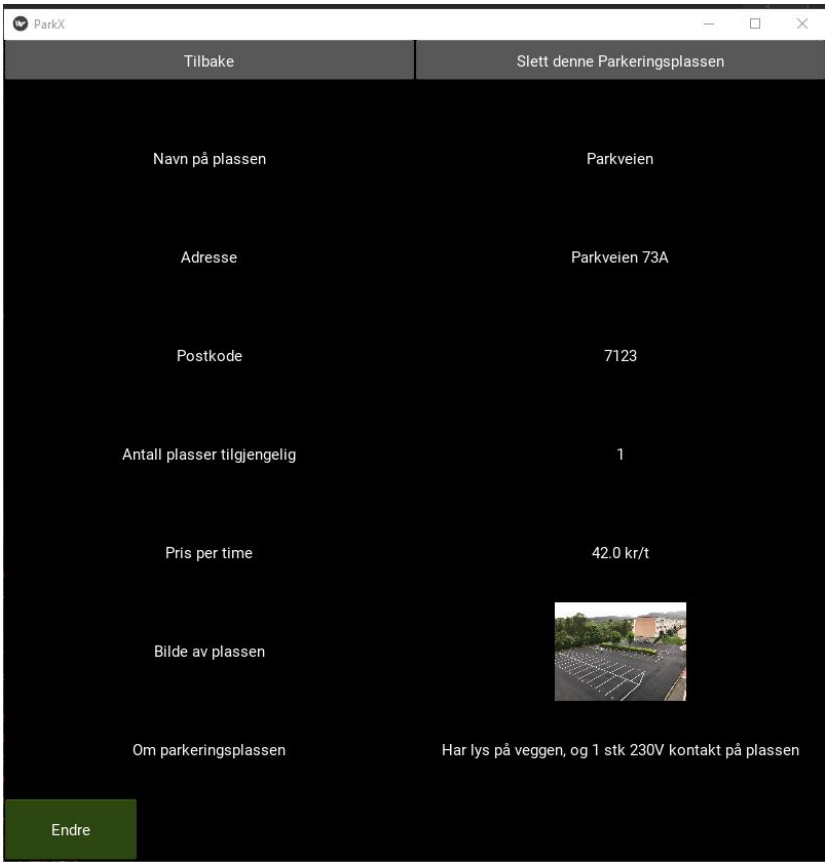
Bruker får også muligheten til å se i detalj. Trykker bruker her, blir en sendt en detaljert visning (se figur 9). Her har bruker også valget å gå tilbake til listen ved å klikke «tilbake». Bruker kan også trykke slett parkeringsplassen, da vil bruker også bli sendt tilbake til parkeringsplass listen. Bruker for også muligheten for å endre parkeringsplassen ved å klikke på «endre». Da vil bruker også bli sendt til et allerede utfylt skjema med den informasjonen som parkeringsplassen hadde fra før (se figur 10). Denne fungerer på akkuratt samme måte som legg til ny parkeringsplass skjemaet og vil også gi en feilmelding pop-up om ikke den angitte informasjon i skjemaet er korrekt.



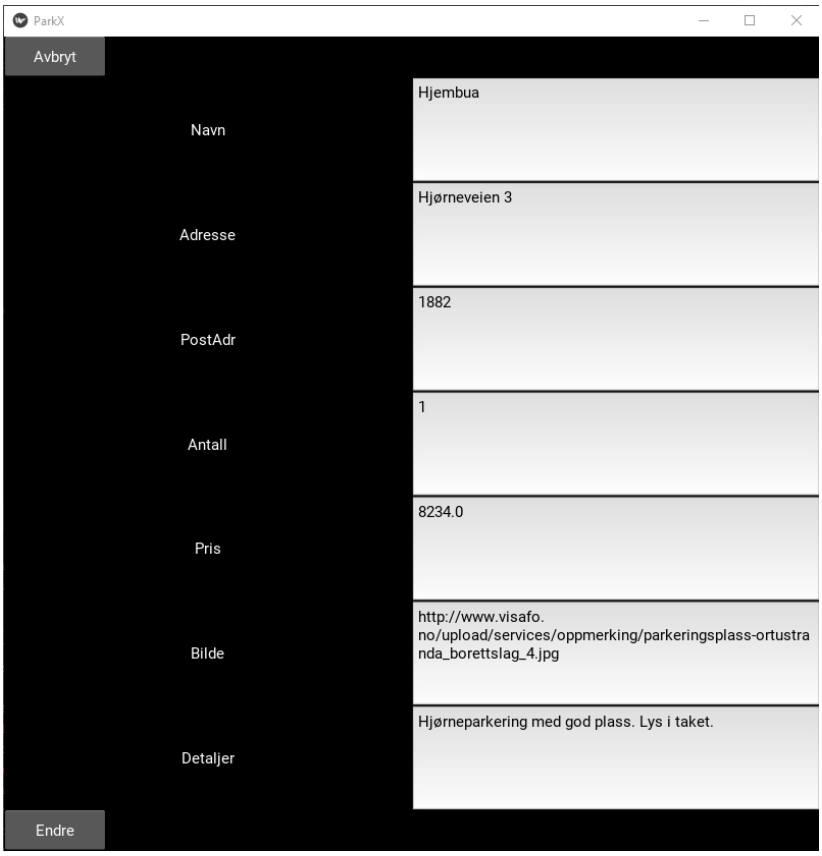
Figur 7: skjema for ny parkeringsplass



Figur 8: feilmelding pop-up om angitt info er riktig



Figur 9: detaljert visning for utleieren

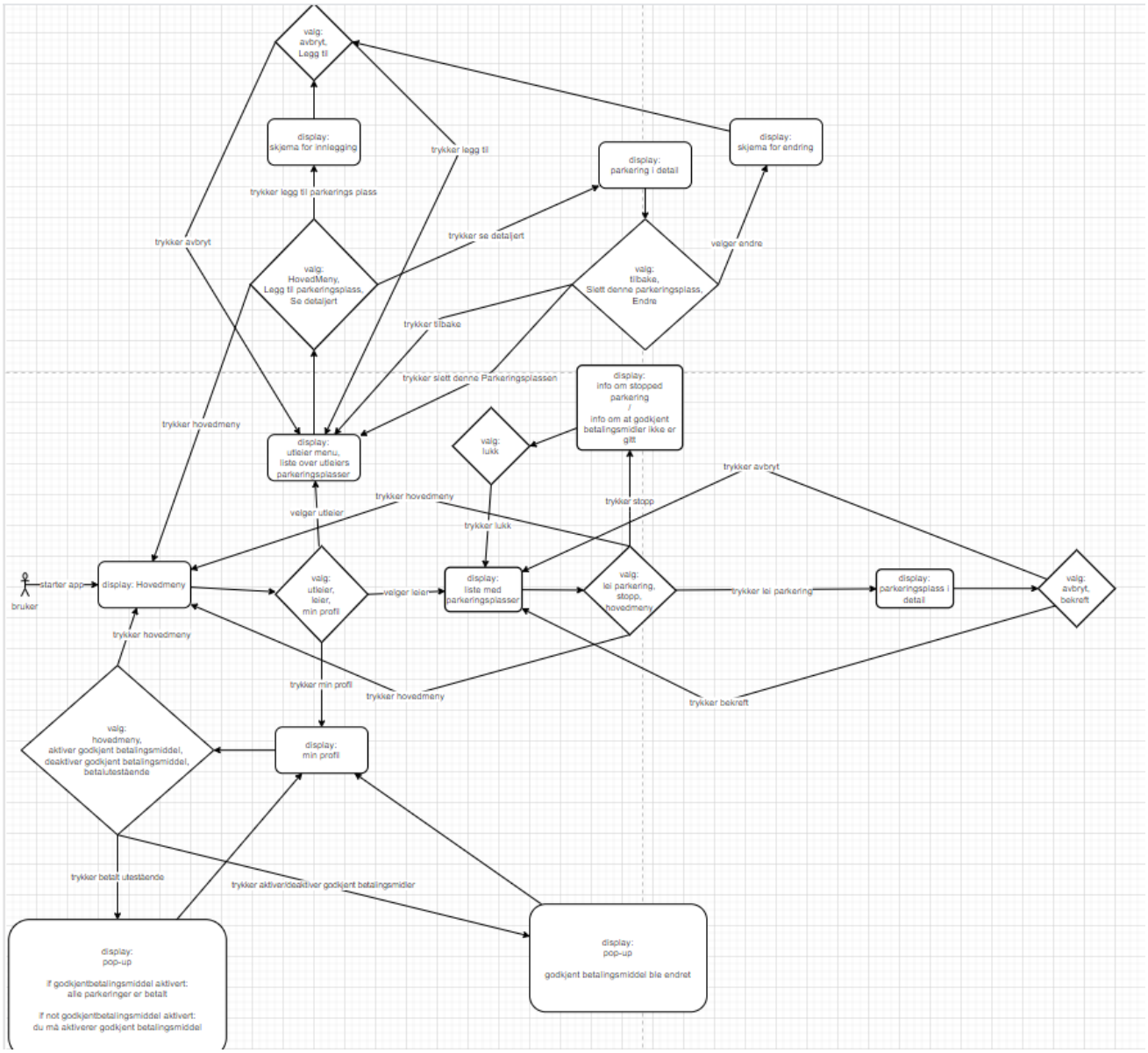


Figur 10: Endre parkeringsplass skjema for utleier

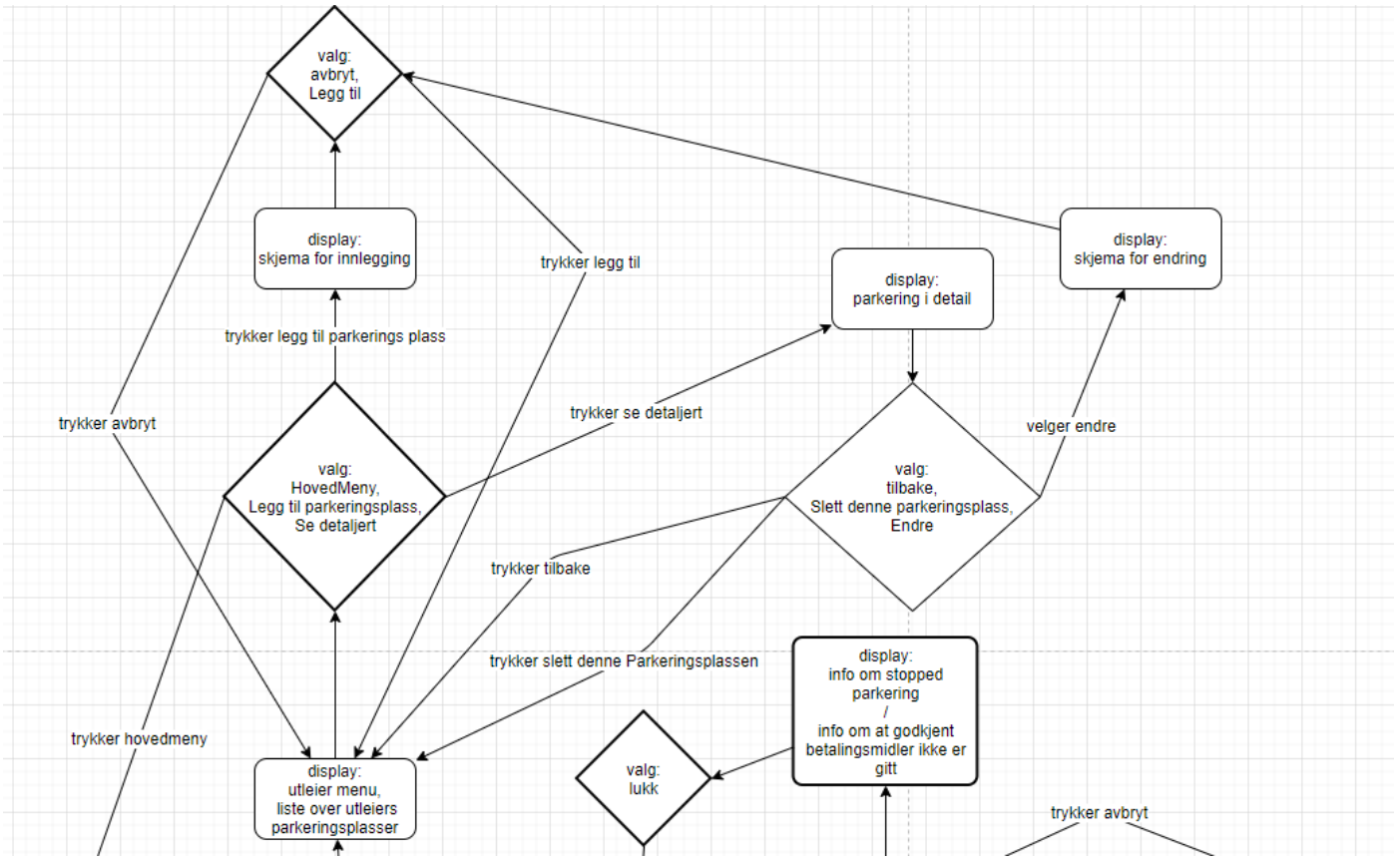
Fra hovedmenyen for også muligheten for å se min profil, ved å klikke på knappen som angir det. Min profil siden er ment for å vise ubetalte parkeringer, samt prisen det kostet (se figur 11). Her har bruker igjen muligheten for å klikke på hovedmeny som sender bruker tilbake til hovedmenyen. Bruker har i tillegg to knapper for aktiverer godkjentbetalingsmidler og deaktiverer godkjentbetalingsmidler, disse styrer også hvilken pop-up som vises etter en parkeringsplass. Betalingsløsningen består av tanken om automatisk betaling, men siden vi tenkte at den automatisk trekk løsningen skulle være styrt av en tredjeparts ordningen er det bare blitt lagt inn knapp som aktiverer de to stadier. Er godkjent betalingsmidler deaktivert vil parkeringer som ikke blir betalt da komme inn i listen som vises i midten av vinduet. Bruker får muligheten for å betale disse utestående parkeringer, ved å klikke «betal utestående». Heretter kan to pop-vises. Den ene sier du må aktiverer godkjente betalingsmidler om denne ikke er aktivert. Den andre vil si alt er blitt betalt.

En større visning av hvordan GUI fungerer utfra bruker handlinger er vist på figur 12, i tillegg til 13, 14 og 15 som viser i detalj.

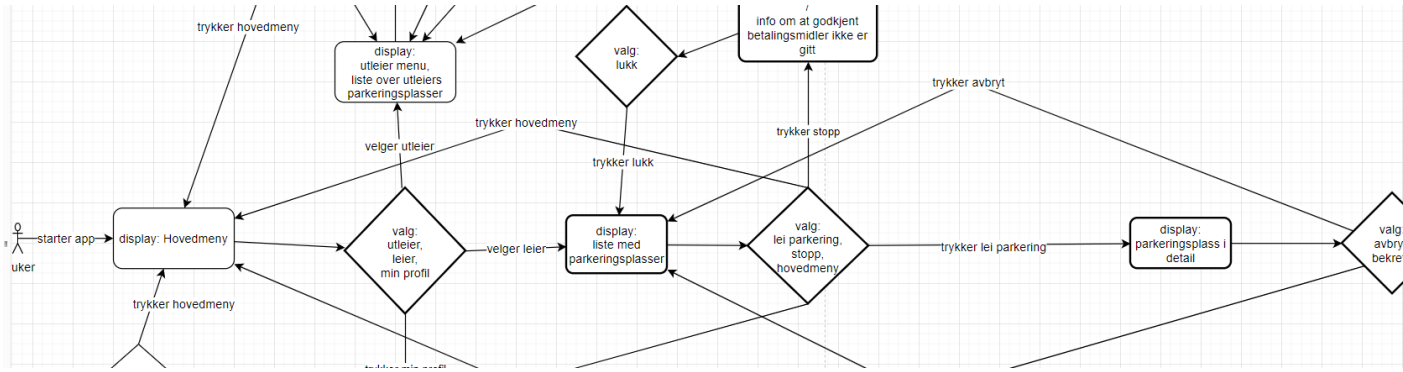
Figur 11: min profil siden med en ubetalt parkering, og to knapper for å fake betaling



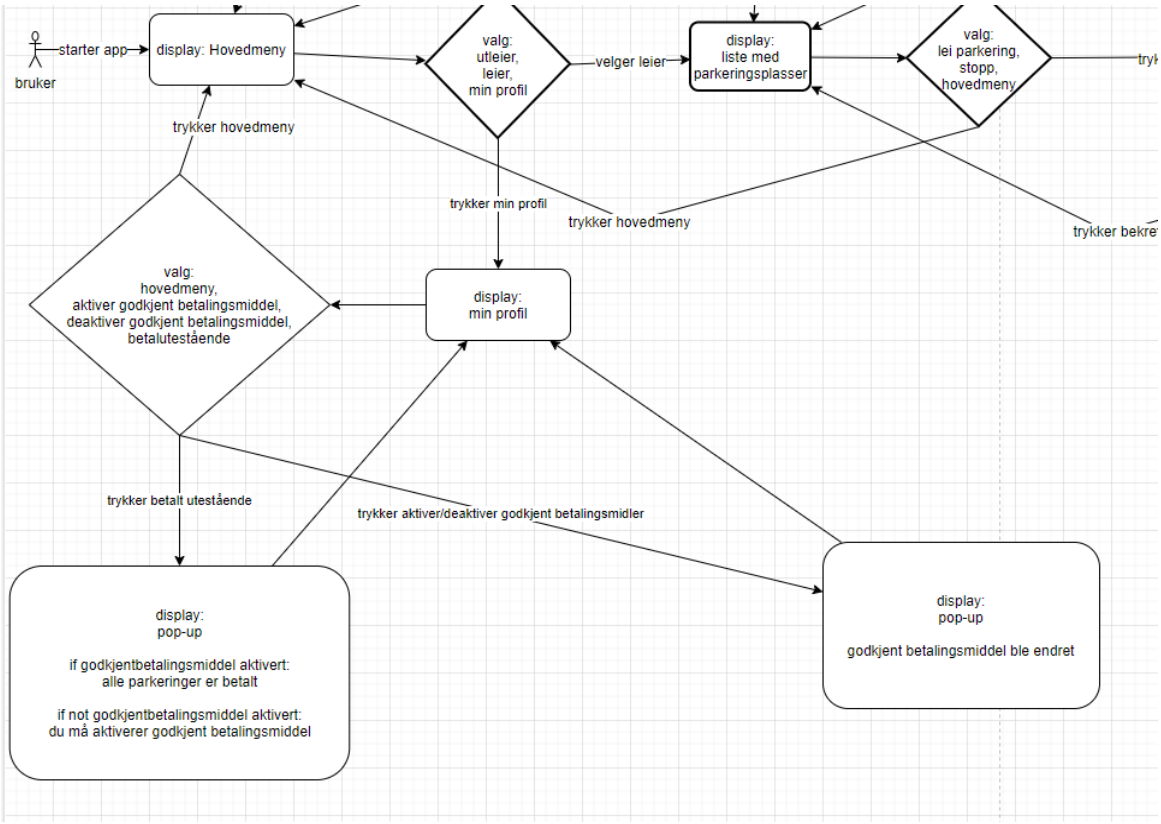
Figur 12: Fullstendig tilstands diagram for hvordan sidene endrer seg. Diagrammet er delt opp i mindre, tydeligere deler under.



Figur 13: øverst del av diagrammet



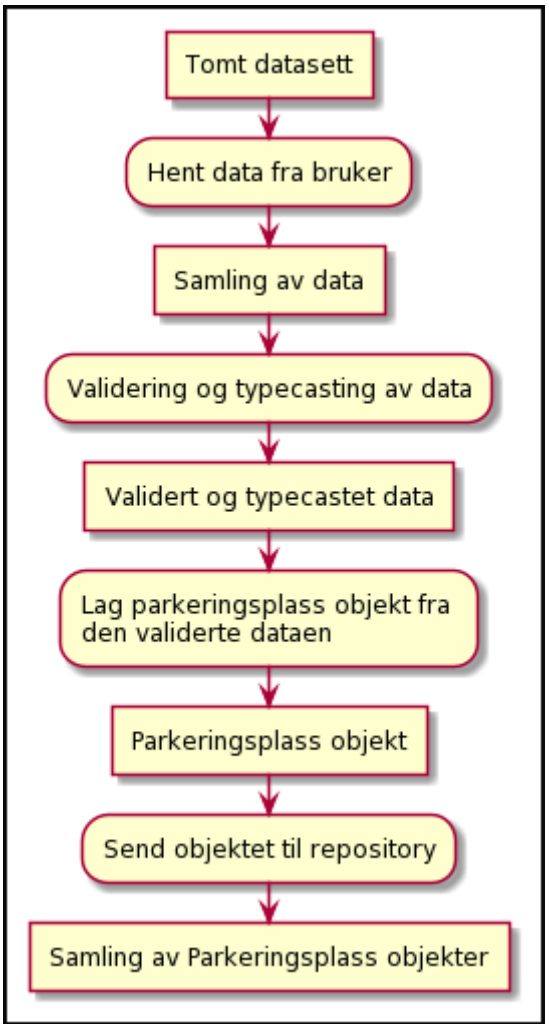
Figur 14: midterst del av diagrammet (zoom inn hvis du ser dette på pc)



Figur 15: nederste del av diagrammet

Innlegging av nye parkeringsplasser:

Innlegging av nye parkeringsplasser har inngangspunkt i innlegging skjemaet i utleier delen av GUI. Når informasjon til den nye parkeringsplassen er fylt inn, blir informasjonen samlet i en dictionary og deretter sendt til ParkingController. Det første ParkingController vil gjøre er å tildele informasjonen en id. Denne vil også være en del av informasjonen som parkeringsplassen lages utfra ParkingController kommer deretter til å gjøre en validering av den input som ble kjørt. Denne validering går bl.a. ut på å teste om informasjon er på riktig format. Eks: antall parkeringsplasser kan bare være et tall. Men valideringen vil også typecaste informasjonen. Dvs. overfører informasjonen den til den type data vi trenger den i. Dette må gjøres siden all informasjon som kommer fra tekstboksene kommer som en streng. Hvis valideringen feiler av eksempelvis, ikke alle felter er blitt fylt ut eller gitt i feil format, vil en feilmelding bli sendt tilbake til Gui. En popup vil da vises som sier informasjon ble skrevet inn feil. Hvis valideringen ikke feiler, blir informasjonen sendt videre til ListRepository. Her vil ListRepository lage en ny parkeringsplass ut fra informasjonen som ble mottatt. Heretter vil parkeringsplassen bli lagret i ListRepository sin liste, hvor alle parkeringsplasser blir lagret.



Figur 16: diagram som viser informasjonsflyt ved innlegging av ny parkeringsplass

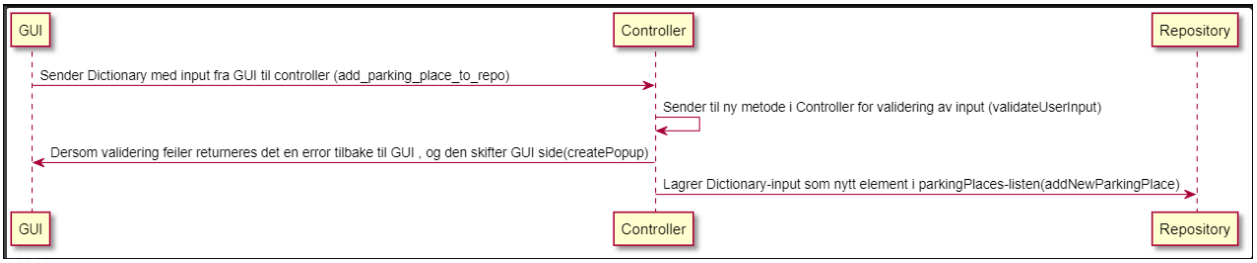


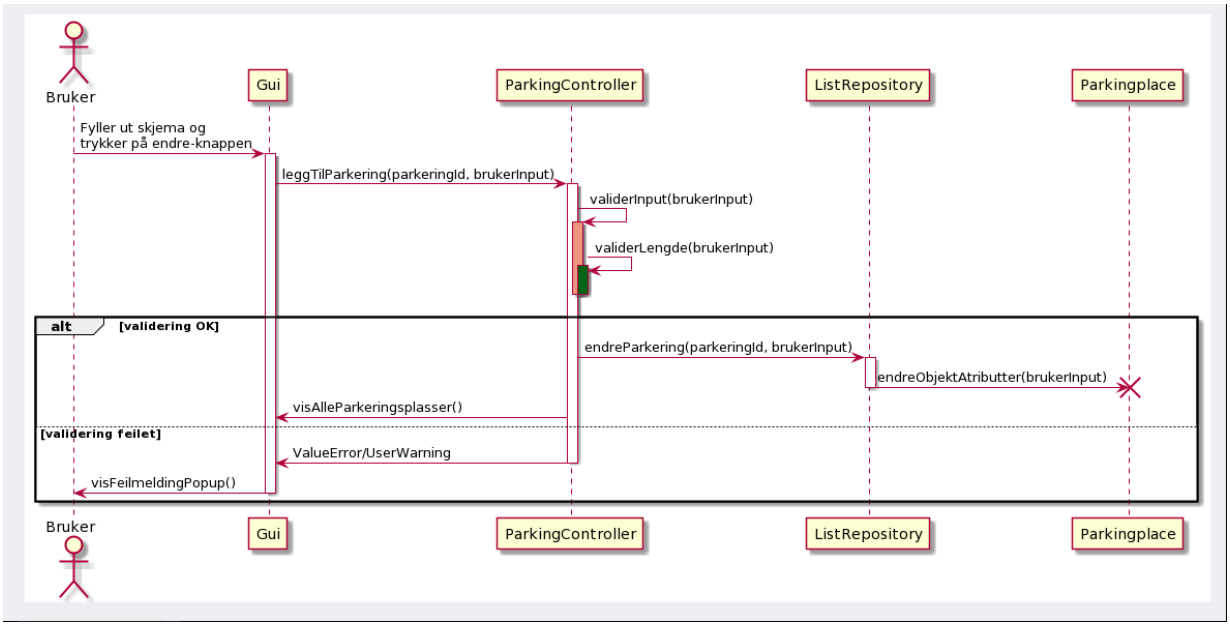
Figure 17: diagram visning av innlegging av ny parkeringsplass

Endre/slette parkeringsplasser:

Endring av parkeringsplasser skjer også i et skjema som ved innlegging av en ny parkeringsplass, men finnes et annet sted. Dette skjema finnes når man trykker endre, i den detaljerte visning av parkeringsplassen for utleier. Skjemaet skiller seg også ut for innlegging av ny parkeringsplass på den måten at all informasjonen som parkeringsplassen hadde fra før står i skjemaets tekstbokser. Når det klikkes endre parkeringsplass er stegene veldig like stegene for innlegging av parkeringsplass. Informasjonen blir sendt til ParkingController for

validering og typecasting, deretter blir informasjonen sendt videre til ListRepository som finner frem den riktige parkeringsplassen. For å finne den riktige parkeringsplass brukes id som parkeringsplassen fikk tildelt, da den først ble laget. Det går ikke an å endre på id'en ved endring av parkeringsplass.

Sletting av parkeringsplass, kan skje hvis bruker klikker på slett parkeringsplass i detaljertvisning av parkeringsplassen for utleier. Parkeringsplassens id vil da bli sendt til ParkingController, som da igjen forteller ListRepository at den skal slette parkeringsplassen med den angitte id.



Figur 18: diagram som viser prosessen i endring av en parkeringsplass (stegene er veldig like det i innlegging av parkeringsplass)

Visning av parkeringsplasser:

Visning av parkeringsplasser skjer i GUI. Over ble det gått gjennom hvordan GUI endrer seg når brukeren beveger seg mellom forskjellige sider. Her vil det bli gått gjennom hvordan GUI ren faktisk får informasjonen som skal vise frem.

Det er to forskjellige måter parkeringsplasser vises frem på, i en liste eller en detaljert parkeringsplass om gangen. Begge stegene er ganske like. Hvis bruker som leier eksempelvis trykker på lei parkering, skal parkeringen vises i detalj før bruker klikker bekreft. Får å vise parkeringen i detalj må GUI få den individuelle parkeringsplassen detaljer. Først og fremst når lei parkeringsknappen trykkes vil en funksjon bli aktivert som kommer til å fjerne alt som

var på siden og deretter bygge den opp på nytt. Fra den forrige siden vil den ha fått en id, denne id er gjemt i selve knappen slik at når knappen trykkes vil den vite hvilken parkeringsplass den skal vise med id'en. GUI sender deretter id'en til en spesiell funksjon i ParkingController som forteller at GUI vil ha informasjonen som tilhører parkeringsplassen med den id. ParkingController vil heretter be ListRepository om å gi den parkeringsplassen med den angitte id, i form av en Parkingplace. Denne returnerer ListRepository til ParkingController som sender den videre til GUI. GUI vil ta imot denne og bygge opp GUI utfra informasjonen som parkeringsplassen har. Dvs. den vil vise navn, adresse og bilde i tillegg til andre felter som hører til parkeringsplassen (se figur 2). Dette skjer ved visning av en parkeringsplass, siden er og metoden er den samme for utleier siden bortsett fra at de to knapper er bygget opp med forskjellige knapper å klikke på (se figur 9).

Ved liste-visning når brukeren skal finne parkeringsplassen som brukeren vil leie, blir flere forskjellige parkeringsplasser vist frem med de viktigste detaljer, som navn, adresse og pris (se figur 3). Når denne siden bygges opp, vil GUI si til ParkingController at den trenger listen med alle parkeringsplasser. ParkingController vil heretter be ListRepository om å få listen med alle parkeringsplasser. Denne listen vil deretter bli returnert til GUI som da vil bygge opp en liten beskrivelse for hver parkeringsplass som finnes i listen. I tillegg vil hver liten beskrivelse også ha en knapp, denne knappen får tildelt id'en til parkeringsplassen den står sammen med. På den måten får funksjonen som blir aktivert ved trykk vite hvilken parkeringsplass den har med å gjøre. Denne logikk gjelder også for endring og sletting av parkeringsplass. Når knapper trykkes og det skal inngå en spesiell handling på en parkeringsplass, følger id'en til parkeringsplassen med på knappen.

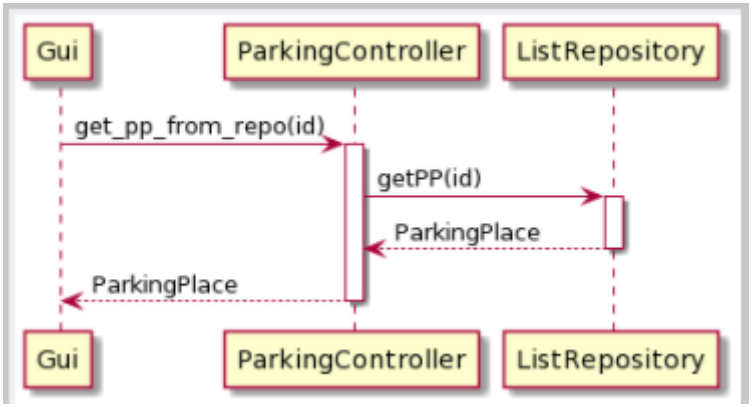


Figure 19: diagram som viser prosessen å hente parkeringsplass informasjon

Leie og stoppe leie av parkeringsplass:

Som beskrevet over ved visning av parkeringsplasser kan vi leie parkeringsplasser. Leie parkeringsplasser skjer i leiers oversikt over alle parkeringsplasser. Leier klikker på lei parkering ved den parkeringsplassen som leier vil leie, hvor etter det byttes til en ny side som viser parkeringsplassen i mer detalj. Oppbygging av selve siden er beskrevet i visning av parkeringsplasser, men allikevel tatt med i diagrammet under for å vise den fulle prosessen. Selve leie prosessen starter når leier klikker på bekreft i den detaljerte visning. Når bruker klikker bekreft vil id'en til den valgte parkeringsplassen bli sendt til ParkingController. ParkingController vil heretter aktivere en funksjon ListRepository med id'en som vil endre statusen til den parkeringsplassen. Parkeringsplassen har også et felt som har oppgave å holde starttidspunktet for leie av parkeringsplassen. Denne vil også bli satt til det nåværende tidspunktet, når endring av status funksjonen blir aktivert. Etter parkeringsplassen har endret status til at den nå blir leiet, vil GUI igjen bytte tilbake til oversikt siden. Men parkeringsplassen som nå blir leiet ligger i den den øverste listen under kalt aktive parkeringsplasser. Denne holder nå en knapp som sier stopp.

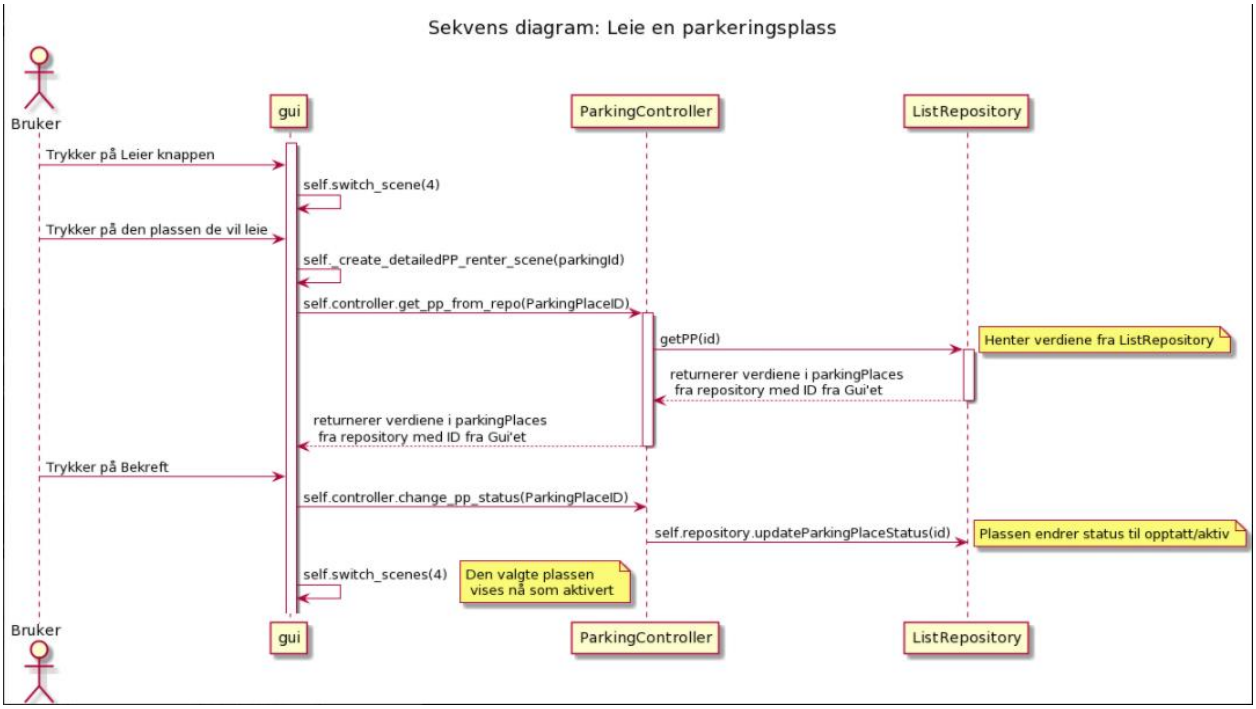


Figure 20: diagram som viser leie prosessen i systemet

Når stopp knappen trykkes vil GUI sende en id'en til ParkingController, som da igjen ber ListRepository endre på statusen til parkeringsplassen med den angitte parkeringsplassen. Denne gang endres statusen til ikke å ikke lengere være i bruk. Heretter vil oversikt siden oppdaterer seg, og en av to forskjellige pop-up vil vises. Godkjente betalingsmidler er ikke aktive eller informasjon om den nå betalte parkeringen. Forklaring til betaling finner du under «lage og betale betalinger». Når disse pop-up lages vil det igjen bli kalt funksjoner i ParkingController og ListRepository for å lage stopptidspunktet og hente parkeringsplassen som parkeringen ble stoppet på. Hver parkeringsplass har en funksjon for å regne ut hvor mye det kostet å leie parkeringsplassen, denne blir aktivert for å kunne regne prisen det kostet å leie parkeringen, men vil også resette parkering starttidspunktet.

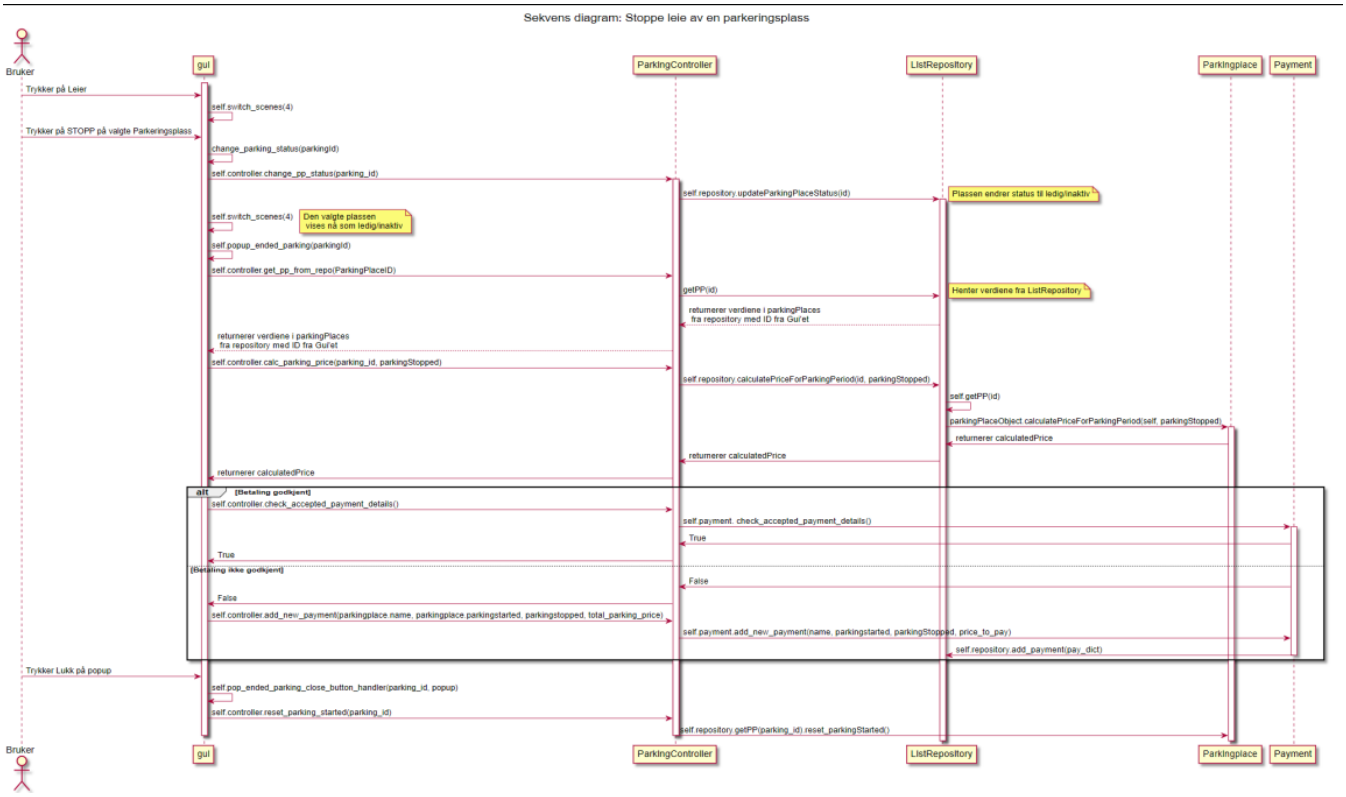


Figure 21: diagram som viser stopp parkering prosessen

Betalinger:

Betalinger (Payments) kan oppfører seg på to forskjellige måter i ParkX, bestemt ut i fra om godkjente betalingsmidler er aktivert eller ikke. Hvis aktivert vil automatisk trekk skje ved trykk på stopp knappen (betalingsprosessen er beskrevet i neste avsnitt). Da vil betalingen ikke bli lagret i selve programmet og en pop-up vil vises med hva den kostet og i hvilket tidsrom du stod på plassen (se figur 4). All denne informasjon regnes ut som beskrevet over ved å hente parkeringsplassens starttidspunkt og pris per time og regne ut ved også å bruke sluttidspunktet.

Hvis godkjente betalingsmidler ikke er aktivert, vil en annen type pop-up vises. Denne vil fortelle automatisk trekk ikke kunne bli gjort fordi godkjente betalingsmidler ikke er aktivert (se figur 5). Nå vil selve betalingen bli lagret. Informasjonen som vises vanligvis om godkjente betalingsmidler er gitt, vil nå i stedet bli sendt til ParkingController. Heretter sender den videre denne informasjonen til Payment, her vil informasjonen bli samlet i et dictionary og bli sendt til ListRepository som lagrer det i en egen liste hvor alle betalinger blir lagret.

For å slå av og på godkjente betalingsmidler må brukeren inn på min profil side. Øverst på denne side ses to knapper aktiver og deaktiver godkjente betalingsmidler. Ved klikk på disse endres programmet til å oppføre seg som godkjente betalingsmidler er gitt eller hvis de ikke var gitt. I tillegg vil det i midten av siden bli vist en liste med de betalinger som ikke har blitt betalt som resultat av at godkjente parkeringsmidler ikke var gitt. Slår bruker på godkjente parkeringsmidler vil de neste parkeringer bli automatisk betalt. Men for å betale forrige ikke betalte parkeringsplasser må bruker klikke på knappen «betal utestående». I tillegg må godkjente betalingsmidler også være på for å kunne betale. Hvis ikke godkjente betalingsmidler er på får man bare en pop-up, som sier dette må bli aktivert før det kan betales. Når betalingen endelig skjer ved klikk på «betal utestående» og godkjente betalingsmidler er aktivert, vil de først bli sendt beskjed til ParkingController. ParkingController vil aktivere en funksjon i Payment som sier alle parkeringsplasser skal bli betalt. Denne metoden sjekker først om betalingsmidlene er aktivert. Hvis aktivert vil metoden sende melding til ListRepository om å tømme listen med betalinger.

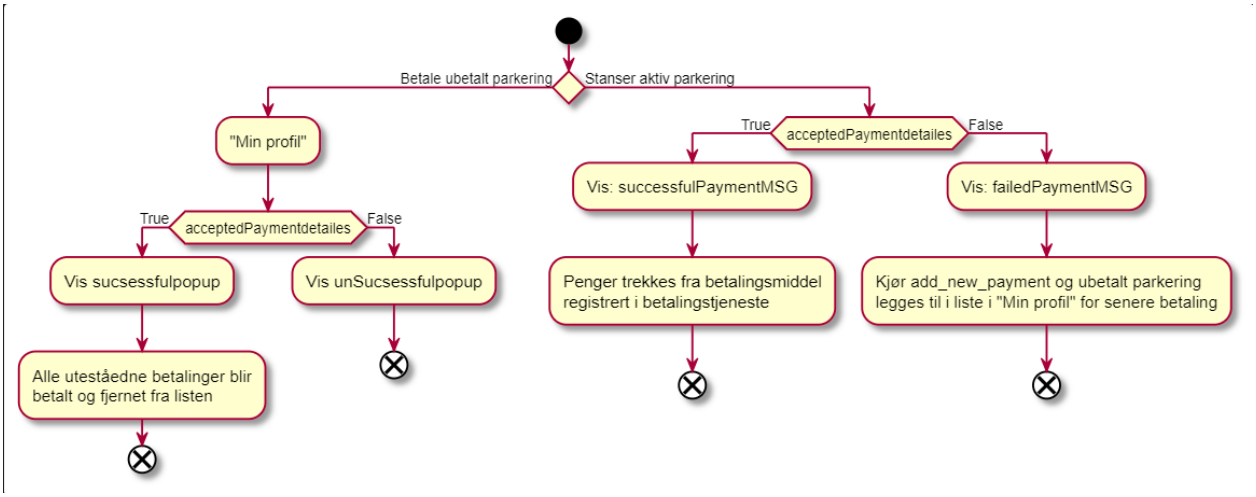


Figure 22: diagram som viser betalingsprosessen

Prototypens rolle i et større system:

Prototypen vår er laget som en MVP (minimum viable product). Dette vil si at prototypen vår er laget med hensikt å være en liten del av det ferdige programmet, men som fremdeles tilfører verdi for den som bruker programmet. Med dette er det også gjort noen forenklinger for å raskest mulig kunne vise funksjonalitet og tilføre verdi for brukeren. Slik programmet er laget nå kommuniserer programmet ikke med noe utenfor selve programmet, og all data lagres lokalt. I det endelige programmet ville systemet hatt en mye større dataflyt inn og ut av programmet. Programmet ville trolig blitt optimalisert for å både kunne bli lastet ned som en app på mobile enheter, og i tillegg kunne brukes som en nettløsning. Brukergrensesnittet på programmet ville og blitt oppdatert og gjort mer brukervennlig, og lagt mer fokus på design. Programmet ville og blitt koblet opp mot en database, slik at all lagring foregikk på en server, og ikke lokalt på enheten. Programmet ville blitt koblet opp mot en tredjepart som håndterer betaling og betalingsinformasjon for oss. Dette er det laget noe kode for i prototypen, for å vise funksjonalitet. Ved å overlate dette til en annen tjeneste ville systemet fått en god løsning på betaling. Dette ville medført at kunden kunne være trygg på at betalingen ble gjort på riktig måte, og at det kunne brukes tjenester der kunde muligens allerede har lagret sine betalingsopplysninger fra før. Dette ville også ført til at det under utviklingen av programmet ikke trengte bli brukt like mye tid på utvikling mot betalingen, noe som ville vært kostnadsbesparende. Programmet ville og koblet opp mot en kart-tjeneste for å enklere kunne vise parkeringsplassene i et kart for brukeren. Her igjen ville det

være fordelaktig da etablerte karttjenester har gode kartløsninger, med forskjellige karttyper, og mulighet for å vise fremkommelighet og trafikk i området.

Valg gjort i prototypen:

- Administrator

  I vår prototype har vi valgt å ikke ha en administrator-rolle. Dette da vi tenker at dette skal være et system der brukerne i størst mulig grad håndterer systemet selv, og at en administrator har minimal nytte ved vanlig bruk. Ting vi tenker administratoren skal kunne gjøre, er og ting vi har valgt å ikke ta med i MVPen, slik som å håndtere klager og fjerne useriøse brukere.

- Brukere

  I vår prototype har vi valgt å kun ha en bruker, og vise frem all funksjonalitet en bruker kan gjøre, fremfor å legge inn et brukersystem. Dette både fordi vi mener det i en MVP ikke tilfører nok verdi i forhold til hvor mye tid det tar. I en reel situasjon kunne det og hendt at man hadde brukt eksisterende bruker-systemer, slik som Facebook eller google og integrert mot deres innloggings-løsning.

- Reservasjon

  I vår MVP har vi valgt å ikke ha muligheten for å reservere en parkering frem i tid. Dette er noe som trolig hadde kommet inn i systemet lenger frem i tid, men i en MVP synes ikke vi det tilfører nok verdi. Vi har derfor valgt at kunder kun kan starte parkeringen i sann-tid, og derfor må betale fra de starter parkeringen.

- Tredjepartsystemer

  I systemet vårt har vi valgt å legge til rette for at det blir brukt tredjepartsløsninger til både innlogging og betaling. Innloggingsdelen er derfor helt utelatt, mens betaling er delvis kodet slik at funksjonaliteten ligger der, og man kun har to knapper i «min profil» for å velge godkjent- eller ikke-godkjent betalingsmiddel. Dette er noe vi hadde sjekket opp mot et betalingssystem, om det som lå inne av opplysninger var godkjent for betaling eller ikke.

- Betaling

  Som beskrevet over, er dette noe vi hadde brukt et tredjepartsystem til. Tanken bak betalingen i systemet er at betalingen hovedsakelig trekkes fra et bankkort man har lagt inn i betalingstjenesten, og at den kun forteller oss om det er godkjente eller ikke-godkjente opplysninger (det de to knappene i min profil representerer). Dersom man stopper en parkering med ikke-godkjente opplysninger, vil man få en feilmelding ved stoppet parkering, og betalingen legges i en liste i min profil. Herfra kan man betalte for alle ubetalte parkeringer når man har lagt inn godkjente betalingsopplysninger igjen. I MVPen er det ikke begrenset hvor mange parkeringer som kan ligge i listen, og man kan starte så mange parkeringer til man vil, men i et videreutvikling av systemet ville det vært noen form for sikkerhet eller auto-fakturaer på disse som hindret for mange ubetalte parkeringer.

- Lagring i MVP

  I systemet vårt har vi kun brukt lagring av dataene i lister i et repository. Dette er en forenkling brukt i prototypen, for å få ordentlig lagring, uten å måtte håndtere databaser og større slike lagringssystemer. På grunn av god lag-deling er det ingen problem å bytte ut repositoryet på et senere tidspunkt mot en database eller lignende for større lagring.

- Flere plasser

  Prototypen vi har laget er i stand til å håndtere flere like plasser, men vi har valgt siden dette er en MVP å ikke gjøre noe ut av denne funksjonaliteten. Dette gjør at man i MVP kun kan opprette en og en plass, og ikke flere like av gangen. Dette hadde blitt implementert senere i systemet.

- Midlertidig deaktivering av plass for eier

  Siden dette er en prototype har vi ikke med muligheten for at en eier av en plass kan deaktivere den når den ønsker bruke den selv. Dette ville og blitt tatt med i en senere versjon av systemet.

- Postkode tall og bokstaver

  For at dette skal være et system som skal kunne brukes i mer enn bare Norge, har vi valgt å lagre postkoder som en string. Dette er for å tilrettelegge for at systemet kan brukes steder der postkode kan inneholde både tall og bokstaver (f.eks. England). Siden dette kun er noe som kun skal leses og vises, og ikke regnes på, har vi gjort dette valget i systemet.

- Lovverk:

  Vi har ikke tatt høyde for hvordan plassene som blir lagt ut på produktet vårt skal kunne passe på at de ikke brukes av noen som ikke har leid dem, eller for hvordan vi skal håndtere biler som blir igjen på plasser etter at utleieperioden har utgått. Siden dette er utenfor prototypen, og muligens noe eier av systemet vil bestemme, er ikke dette omfattet av dokumentasjonen eller prototypen.

- Eksempel parkeringsplasser:

  Siden det er brukere av systemet som skal legge til reelle plasser, har vi i prototypen lagd tre eksempelplasser som er standard i prototypen for at man enkelt skal kunne vise funksjonalitet. Disse blir laget i en egen metode i ListRepository, og metoden kalles ved oppstart av funksjonen fra main. Dette gjør at den enkelt kan fjernes fra systemet når det skal rulles ut til brukere.

Kjente svakheter i prototypen:

Siden dette bare er en prototype og ikke er noe ferdig produkt, er det forventet det er noen svakheter i prototypen. Noen av svakhetene kommer også direkte av at det er en prototype og mangelen på oppkobling til eksterne servere og databaser.

I prototypen valgte vi at å lage en bruker-klasse ikke var viktig akkurat for prototypen. Det var fortsatt mulig å vise frem hvordan programmet kommer til å se ut når det er ferdig. Men dette er en svakhet til prototypen. Ingen bruker klasse betyr altså vi ikke vil kunne teste hvordan systemet ser ut når vi har flere brukere. Det ses også i systemet direkte ved at parkeringsplasser som legges inn er de parkeringsplasser som man kan leie. Dette valget er

med mening siden at bruker klassen ikke får verdi, før vi kan koble det opp servere som gjør vi kan gi forskjellige brukere forskjellige parkeringsplasser. Denne svakhet forsvinner når vi kan koble oss til en server og blir tvunget til å dele opp parkeringsplasser på brukere.

Prototypen har en betalingsløsning i form av en klasse som styrer betalingsdelen i GUI. Denne bruker en True eller False variabel som avgjør om brukeren får betalt for parkeringen eller ikke. Ideelt burde dette muligens vært en mer utvidet løsning, da dagens løsning bryter SRP-prinsippet. En bedre løsning kunne vært å ha en ny controller som styrer betalingene og har variabelen som angir godkjente betalingsopplysninger. Denne kontrolleren kobler sammen et ny repository og GUI, og har ansvaret for å styre lagringer og distribueringen av Payment objekter. Resultat av dette vil være bedre følging av Single Responsibility Principle, på grunn av enhver ny klasse vi lager ender opp med bare en oppgave. PaymentController er ansvarlig for transportering av Payment objekter. Payment objektet i seg selv er ansvarlig for å utføre operasjoner på betalingsobjekter og PaymentRepository vil kun lagre og gi ut Payment objekter.

Prototypen har også noen svakheter angående sletting og endring av parkeringsplasser som er i bruk. Leier man en parkeringsplass, går det også an å slette og endre parkeringsplassen. Dette problemet hører også til i større grad at vi ikke har noen bruker klasse. Uten bruker klassen blir det altså også vanskeligere å si noe om hvilke rettigheter bruker har, i tillegg er det er spørsmål om lov og hvilke endringer utleier kan gjøre, mens parkeringsplassen blir utleid.

Prototypen har også noen mindre svakheter, som evnen til å vise bilder går kun gjennom en URL. Vi bestemte dette ikke var noen MVP å gå i dybden på hvordan å åpne file explorer eller brukerens bilder på mobil. Dette hadde selvfølgelig vært gjennomført på en bedre måte hvis ikke dette bare var en prototype.

Et annet mindre problem er at tiden som blir satt for å regne ut når parkering starter og stopper ikke går på dato. Dvs. prototypen vil altså vise feil om bruker leier mer enn en dag med prototypen. Dette blir ikke noe problem man nødvendigvis legger merke til med mindre prototypen brukes over lengere tid.