Forside IS-304: 2022

Tittel: Utvikling av web-applikasjon for reservering av ressurser

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig	Hallgeir Nilsen
Veileder	Devinder Bahadur Tapa
Oppdragsgiver	Capgemini Norge AS

Studenter:

Etternavn	Fornavn
Faret	Jaran
Hagum	Stig
Hennestad	Andreas
Løvberg	Martin

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeide uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA ✓	NEI___
Kan besvarelsen brukes til undervisningsformål?	JA___	NEI ✓
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA ✓	NEI___

Forord

Vi i gruppe 13 heretter kalt Radioheads har hatt gleden av å utarbeide denne rapporten som den avsluttende delen av vår bachelorgrad i IT og informasjonssystemer ved Universiteter i Agder i Kristiansand. Under RefreshIT i 2021 ble gruppa introdusert til Capgemini Kristiansand og deres spennende oppgavebeskrivelse. Gjennom semesteret har Capgemini og deres ansatte vært tilgjengelige for spørsmål, møter, opplæring, tilgang til ressurser og til og med jobbtilbud. Radioheads vil derfor rette en stor takk for all hjelp og støtte vi har fått fra bedriften.

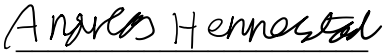
Vi vil også takke fagansvarlig Hallgeir Nilsen, som har satt opp presentasjoner og møter med andre grupper. Dette har åpnet for diskusjon og støtte fra flere som skriver bacheloroppgave. Han har også holdt forelesninger og invitert gjesteforelesere som har kommet og snakket om deres opplevelser om å skrive bacheloroppgave, og relevante arbeidsmetoder. Disse forelesningene har gitt gruppa masse nyttige tips og råd som har hjulpet oss gjennom hele semesteret.

Til slutt vil vi rette en takk til Devinder Bahadur Tapa som har vært gruppas veileder gjennom semesteret, og har vært tilgjengelig for svar på spørsmål vi har hatt i løpet av semesteret og gi tilbakemeldinger på arbeidet vi har gjort.

Alle medlemmer samtykker at de har bidratt til utviklingen av dette prosjektet til beste evne.
Gruppa består av følgende medlemmer:

Jaran Faret

Stig Hagum



Andreas Hennestad

Martin Nyenget Løvberg

Sammendrag

Dette prosjektet er utarbeidet av studiegruppen The Radioheads i samarbeid med Capgemini Kristiansand. Dette samarbeidet ble startet under RefreshIT 2021 i Oktober 2021. Oppgavebeskrivelsen handlet om å utvikle et system hvor ansatte hos Capgemini kunne reservere møterom i de nye kontorene til bedriften. Det skulle også kunne tilpasses andre organisasjoner som ville låne ut andre ressurser til sine ansatte.

Prosjektet har vært en full-stack oppgave hvor vi har tatt en produktbeskrivelse av prosjektet, og gjennom flere prosesser har vi endt opp med et fungerende produkt som oppfyller kravene fra bedriftene.
Gjennom intervju og diskusjon har vi laget brukerhistorier, og videre fått etablert oppbygningen av systemet. Disse brukerhistoriene har vi brukt til å lage systemkrav som har hjulpet oss å komme frem til hvordan systemet skal se ut visuelt.
Til slutt har vi programmert systemet til beste evne og kommet frem til et fungerende produkt. Vi har benyttet Scrumban som arbeidsmetodikk og fokusert på at alle har hatt et høyest og jevnest mulig læringsutbytte.

Det endelige produktet er et resultat av nesten daglig arbeid for hele gruppen, og grundig gjennomgang og demonstrasjon av produktet kan finnes her: https://drive.google.com/file/d/1mDcmca2lBIkNQVFLU5l6lLJWVAbnP9to/view?usp=sharing

Link til kildekoden finnes her: https://github.com/Sthagum/Bachelor2022

Innholdsfortegnelse

Figurliste

1. Introduksjon

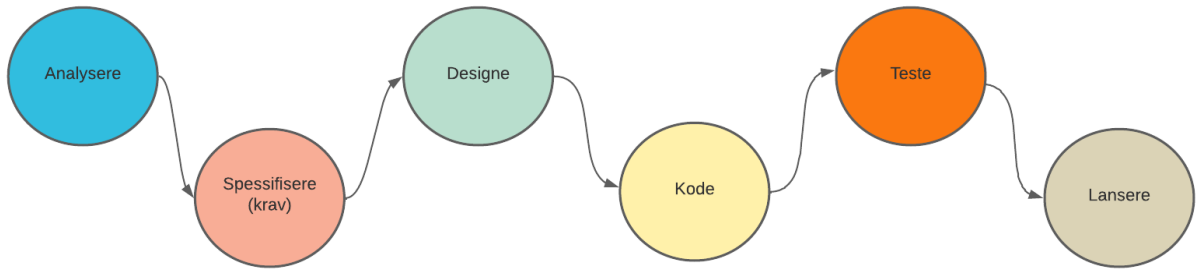
Etter gruppa hadde deltatt på RefreshIT Expo i 2021, var det en bedrift som hadde satt et ekstra godt inntrykk på oss, med en profesjonell og ansvarlig tilnærming til bachelorprosjektet de presenterte. Denne bedriften var Capgemini, en internasjonal konsulentbedrift som har avdelinger og kontorer over hele verden. Vi fikk en god tone under RefreshIT, og samtlige på gruppa var interesserte i en videre dialog om oppgaven. Gruppa fikk innkalling til møte for å grundigere diskutere både oppgaven og oss selv som gruppe. Etter møtet ble vi spurt om vi ønsket å utføre bachelorprosjekt hos dem, noe vi gjerne takket ja til.

The Radioheads er en gruppe på fire som har jobbet sammen siden første semester. Til tross for forskjellige valgfag har vi stor erfaring med hverandres styrker og svakheter, og er veldig komfortable med å jobbe sammen og fordele oppgaver. Gjennom de ulike prosjektene vi har vært gjennom, har vi hovedsakelig jobbet med Scrum, og ønsket å fortsette med noe som var lignende denne metodikken gjennom prosjektet. I løpet av prosjektfasen begynte de nasjonale restriksjonene angående Covid-19 å lette opp, noe som førte til at vi kunne sitte mye mer fysisk og jobbe enn tidligere. Vi har variert med å sitte på campus, hjemme eller på de nye kontorene til Capgemini, som også er en sentral del av oppgaven vår.

For mer enn to år siden ble verden rammet av en global pandemi som gjorde det vanskelig for mange bedrifter å samle sine ansatte på ett sted. Hjemmekontor ble normen, og i mange måneder i strekk var kontorbygg helt tomme, til tross for at bedriftene ofte fortsatte å betale en fast leiesum for lokalene. Ut ifra denne problemstillingen kom ideen til dette bachelorprosjektet. Capgemini ønsket et system som gjorde det mulig å reservere kontorplass de dagene man ønsket å sitte på kontoret, slik at den kontorplassen eller møterommet var ledig hvis ikke. På denne måten kunne bedriften kun betale for de plassene som har blitt brukt, samt holde styr på hvor mange ansatte som møter opp på jobb til enhver dag. Samtidig har Capgemini flyttet inn i nytt kontorbygg hvor huseier ønsker å kombinere Capgeminis ønsker om kontorbooking med booking av grupperom i bygget.

Capgemini ønsker også at systemet er tilpasningsdyktig, slik at denne reserveringen kan gjelde for flere ressurser enn bare kontorplasser eller møterom, for eksempel firmabil eller overtidsmat. De som jobber i kantina kan bruke antallet reserverte plasser til å beregne hvor mye mat de skal forberede, og dermed hindre matsvinn.

Vi har utarbeidet en plan hvor vi først skal analysere og spesifisere krav for å finne ut av hvilke funksjoner systemet vårt skal ha og hvilke oppgaver de skal løse. Deretter skal vi begynne med å designe systemarkitektur for å planlegge hvordan systemet skal se ut og henge sammen. Når alt forarbeidet er klart begynner vi med å kode systemer for å så teste og lansere. Vi har bestemt oss for å være åpne for endringer i planen da det kan oppstå uforutsette hendelser vi må ta tilsyn til.



Figur 1: Arbeidsplan

Arbeidsplanen på figur 1 referer ikke til en rigid arbeidsmetode, da vi arbeider med en agil metodikk. Denne arbeidsplanen er ment som en liste som viser hvilken rekkefølge vi må arbeide for å få best mulig utbytte av prosessene.

2. Analyse

Begynnelsen av prosjektet, før utviklingen kunne begynne, måtte gruppa analysere bruksområder og brukere for å kunne kartlegge hvilke funksjoner vi skal inkludere, og prioritere i systemet. Analysedelen av prosjektet tar for seg hvordan teamet jobbet for å hente nødvendig informasjon. Vi tok utgangspunkt i oppgavebeskrivelsen vi fikk fra kontaktpersonen, og utarbeidet et intervju med huseieren for å finne ut hvilke funksjoner som er mest ønskelige. Med resultatene fra intervjuet kunne vi lage noen brukerhistorier, og sette disse inn i en prioriteringsliste. Etter systemkravene var satt, hadde vi nok informasjon til å iverksette utvikling av systemarkitektur.

2.1 Intervju

Gjennom dette prosjektet har Capgemini fungert som både arbeidsgiver og prosjekteier. Systemet skal tas i bruk av ansatte hos Capgemini, men skal også kunne tilpasses for andre kontorbygg. Vi hadde allerede fått grundig informasjon gjennom oppgavebeskrivelsen, men vi følte vi måtte gjennomføre et intervju med huseier for å få et annet perspektiv enn en standard

bruker. Av erfaring syntes vi at et semi-strukturerte intervju fungerer best for å få mest mulig nøyaktige resultater fra enkeltpersoner, med mulighet til å gå bort fra manus for å stille oppfølgingsspørsmål. Vår erfaring stemmer med en artikkel fra Scribbr som forklarer at semi-strukturerte intervju gir god mulighet til å holde seg til et rammeverk, men fortsatt ha muligheten til å grave dypere inn i enkelte spørsmål og svar. (George, 2022)

Det transskriberte intervjuet ligger under vedlegg. Resultatet fra intervjuet tilfredsstilte de funksjonalitetene som vi selv hadde diskutert oss frem til, noe som bekreftet at vi var på riktig spor i henhold til oppbyggingen av prosjektet. Intervjuet ga oss også bedre oversikt over nøyaktig hvilke funksjoner som må være i programmet og hvordan han som huseier har sett for seg at de forskjellige funksjonene er koblet sammen. Dette er f.eks. hvordan han har sett for seg at kantinen skal bli implementert i systemet og hvordan de skal kommunisere med resten av systemet.

2.2 Brukerhistorier

En brukerhistorie er en kort og simpel beskrivelse av en funksjon fortalt fra perspektivet av en person som skal bruke det nye systemet og hvorfor personen ønsker denne funksjonen. De utarbeides basert på systemkravene. Brukerhistorier er noe som brukes i agile prosjekter, og målet er å ende opp med en prosjekt-backlog som er problemer vi må løse for å oppnå målet vårt. De største godene av å bruke brukerhistorier er at man tenker over hvem som skal bruke systemet, og ser hva de skal bruke det til slik at vi kan komme på funksjoner som løser det de vil gjøre. De bidrar også til å skape diskusjon innad i gruppa når vi utarbeider dem sammen over hvilke funksjoner vi trenger og hvorfor. Brukerhistoriene hjelper oss også med estimeringen senere. (MountainGoat Software, 2022)

Brukerhistorier kan utformes på forskjellige måter, men vi har valgt å følge oppskriften: «Hvem, hva og hvorfor». Det vil si at vi vil utforme brukerhistorien basert på **hvem** som skal bruke systemet, **hva** brukeren vil gjøre, og **hvorfor** brukeren vil gjøre dette. Først har vi utarbeidet veldig omfattende brukerhistorier som vi kan kalle «epics», disse deler vi så opp i mindre med spesifikke brukerhistorier som har oppgaver som er lettere å estimere tidsbruk på og gjennomføre i løpet av en sprint. (Capgemini, 2022)

I tabellene under har vi samlet alle bruker historiene under sine tilhørende epics:

Epic: Som bruker av systemet, ønsker jeg å registrere meg i systemet for å ta i bruk funksjonene det tilbyr.
Brukerhistorie
Som bruker ønsker jeg å ha et eget profilbilde for å lett se at jeg er logget inn på min profil.
Som ny bruker av systemet ønsker jeg å kunne legge inn min relevante info for å kunne oppdrette en bruker.
Som bruker ønsker jeg å kunne gjenopprette glemt brukernavn eller passord dersom jeg skulle glemme det.

Epic: Som huseier ønsker jeg at bygget skal være så bærekraftig og økonomisk gunstig som mulig.
Brukerhistorie
Som huseier ønsker jeg at brukere skal registrere om de vil spise i kantina eller ikke slik at kantina kan forberede passende mengde mat for god bærekraft, økonomi og minst mulig matsvinn.

Epic: Som administrator ønsker jeg å ha kontroll og oversikt over brukere.
Brukerhistorie
Som administrator ønsker jeg å se når en bruker ble opprettet eller endret for å ha god oversikt.
Som administrator ønsker jeg å knytte brukere opp mot en bedrift slik at de ikke kan booke plass i kontorene som tilhører andre bedrifter.
Som administrator ønsker jeg å kunne opprette, slette og endre brukere for å kunne håndtere brukere og rettigheter.

Epic: Som bruker ønsker jeg å kunne booke møterom for hele teamet mitt slik at vi kan samarbeide.
Brukerhistorie
Som bruker vil jeg kunne se antall plasser på møterom jeg booker for å unngå for få eller mange plasser.
Som bruker vil jeg se om rommet har utstyr for teams møte for å være sikker på at det har det vi trenger.
Som bruker vil jeg se når rommet er ledig for å finne ledige rom.
Som bruker vil jeg se rommene jeg har tilgang til for å ikke booke rommene til andre firma.
Som bruker vil jeg ha en oversikt over ledige rom som jeg kan booke for å kjapt finne noe.
Som bruker vil jeg avbestille rom dersom vi ikke trenger det lenger.
Som bruker vil jeg bestille mat for at kantinen for bygget bedre kan estimere mengden med mat som må lages hver dag.
Som bruker vil jeg at bookingen skal legges inn i timeplanen til deltakerne
Som bruker som booket rommet vil jeg sende varsler til deltakerne med tid og sted for å informere om hvor og når møtet er.
Som bruker vil jeg få en påminnelse før møtet for å rekke møte.

Epic: Som en ansatt som jobber i bygget ønsker jeg å booke kontorplass slik at jeg kan jobbe på kontoret
Brukerhistorie
Som bruker ønsker jeg å få en oversikt over alle tilgjengelige plasser slik at jeg vet hvilke jeg kan velge mellom.
Som bruker ønsker jeg informasjon om hvilke hjelpemidler som er tilgjengelig på plassen slik at jeg kan velge plass basert på hvilke hjelpemidler jeg trenger.

Som bruker ønsker jeg å kunne reservere den kontorplassen jeg ønsker meg.
Som bruker ønsker jeg å kunne frasi meg kontorplassen dersom jeg ikke trenger den lenger.
Som bruker ønsker jeg å kunne registrere om jeg skal ha mat i kantina slik at kantina kan klargjøre en estimert mengde mat og hindre matsvinn.

Epic: Som en ansatt som jobber i bygget ønsker jeg å kunne se mine egne bookinger slik at jeg får en personlig oversikt.
Brukerhistorie
Som bruker ønsker jeg å se hvor jeg har booket plass for å huske hvor jeg skal være.
Som bruker ønsker jeg å se når jeg har booket plass for å huske når jeg skal være der.

Epic: Som avdelingsleder ønsker jeg at mine ansatte skal kunne registrere om de møter opp på kontoret slik at jeg har en oversikt over hvor mange som møter opp.
Brukerhistorie
Som avdelingsleder ønsker jeg en visuell oversikt over hvilke kontorplasser som er i bruk.
Som avdelingsleder ønsker jeg en visuell oversikt over hvilke ansatte som har reservert kontorplasser.

Figur 2: Brukerhistorier

2.3 Systemkrav og estimering

Ved å etablere brukerhistorier som nevnt i avsnittet over, kan vi ta for oss hvilke funksjoner vi skal implementere i programmet. Brukerhistoriene gir oss mange ulike funksjoner å velge mellom, og ettersom vi arbeidet med en agil tilnærming, måtte vi etablere hvilke systemkrav som var viktigst, og hvilke som var mindre viktige. En god metodikk som gruppa har erfaring med er MoSCoW-metoden. Denne metodikken tar for seg hver funksjon, hvor gruppa diskuterer om funksjonen faller innenfor kategoriene ‘Must have’, ‘Should have’, ‘Could have’ og ‘Won’t have’ (Brush, 2020). Systemkrav som er helt nødvendige for at systemet skal fungere, blir

registrert som ‘Must have’. ‘Should have’ inkluderer systemkrav som ikke er livsnødvendige for funksjonaliteten, men som burde være med for at systemet skal være behagelig å bruke. Systemkrav med ‘Could have’ er funksjonaliteter som ikke nødvendigvis er så høyt oppe på prioriteringslista, men kan forbedre systemet dersom det er tid til det. Systemkrav som vi gjerne skulle hatt med, men ikke har tid til, faller under ‘Won’t have’.

Estimering er en stor del av gjennomførelsen av prosjekt. Av erfaring er det også noe av det vanskeligste å gjennomføre. Gjennom semestrene har vi vært innom flere metoder for estimering, men Capgemini introduserte oss til en metodikk som heter ‘planning poker’ (Planning Poker, 2022). Ved hjelp av denne metodikken kunne alle på gruppa komme med sin mening om hvor lang tid en spesifikk oppgave tok, og begrunne sine argumenter. Dermed fikk alle på gruppa et felles perspektiv for oppgaven og kunne lettere bli enige om en estimering.

Nedenfor har vi utarbeidet en tabell med systemkrav, prioritering og estimering. Dersom vi skulle ha med alle systemkravene hadde vi endt opp med en arbeidsmengde på 132 timer, noe som videre understreker viktigheten av tidsestimering og prioritering. Dette er kun et estimat, og med tanke på at vi selv ikke har lang erfaring med systemutvikling prosjekter vil vi alltid ha i bakhodet at ting kan ta lengere eller kortere tid enn det vi har kommet frem til i estimatene våre. Derfor passer det bra at vi har en agil metodikk som lar oss tilpasse oss endringer i planer underveis.

Systemkrav	Prioritering	Brukerhistorie	Estimering (timer)
Oversikt over ledige rom	Must have	Som bruker vil jeg ha en oversikt over ledige rom som jeg kan booke for å kjapt finne noe.	4
Registrering med navn, bilde og rolle	Must have	Som ny bruker av systemet ønsker jeg å kunne legge inn min relevante info for å kunne oppdrette en bruker.	5
Møterom har oversikt over antall plasser	Must have	Som bruker vil jeg kunne se antall plasser på møterom jeg booker for å unngå for få eller mange ledige plasser.	5
Avbestilling av rom funksjon	Must have	Som bruker vil jeg avbestille rom dersom vi ikke trenger det lenger.	3
Bruker må kunne booke kontorplass som er ledig.	Must have	Som bruker ønsker jeg å kunne reservere den kontorplassen jeg ønsker meg.	3

Bruker må kunne fjerne reservasjon av kontorplass.	Must have	Som bruker ønsker jeg å kunne frasi meg kontorplassen dersom jeg ikke trenger den lenger.	3
Filopplastning	Could have	Som bruker ønsker jeg å ha et eget profilbilde for å lett se at jeg er logget inn på min profil.	5
Registreringsdato og endringsdato på bruker	Could have	Som administrator ønsker jeg å se når en bruker ble opprettet eller endret for å ha god oversikt.	4
Administrator rollen kan redigere brukere	Could have	Som administrator ønsker jeg å kunne opprette, slette og endre brukere for å kunne håndtere brukere og rettigheter.	5
Visuell oversikt over reserverte kontorplasser.	Could have	Som avdelingsleder ønsker jeg en visuell oversikt over hvilke kontorplasser som er i bruk.	7
Visuell oversikt over hvilke ansasatte som har reserverte kontorplasser.	Could have	Som avdelingsleder ønsker jeg en visuell oversikt over hvilke ansatte som har reservert kontorplasser.	7
Registrere ønske om mat	Should have	Som bruker ønsker jeg å kunne bestille mat opp til møterommene dersom jeg har gjester som ønsker dette.	1
Glemt passord funksjon	Should have	Som bruker ønsker jeg å kunne gjenopprette glemt brukernavn eller passord dersom jeg skulle glemme det.	6
Registrere ønske om mat	Should have	Som huseier ønsker jeg at brukere skal registrere om de vil spise i kantina eller ikke slik at kantina kan forberede passende mengde mat for god bærekraft, økonomi og minst mulig matsvinn.	7
Koble sammen bruker og bedrift. For å gi adgang til bedrifts plasser	Should have	Som administrator ønsker jeg å knytte brukere opp mot en bedrift slik at de ikke kan booke plass i kontorene som tilhører andre bedrifter.	8
Møterom har oversikt over teknisk utstyr	Should have	Som bruker vil jeg se om rommet har utstyr for teams møte for å være sikker på at det har det vi trenger.	5

Funksjon for å se om grupperom er opptatt eller ledig	Should have	Som bruker vil jeg se når rommet er ledig for å finne ledige rom.	3
Oversikt over bedriftens tilgjengelige møterom	Should have	Som bruker vil jeg se rommene jeg har tilgang til for å ikke booke rommene til andre firma.	4
Oversikt over alle ledige plasser som kan bookes.	Should Have	Som bruker ønsker jeg å få en oversikt over alle tilgjengelige plasser slik at jeg vet hvilke jeg kan velge mellom.	5
Oversikt over tilgjengelige verktøy på kontorplass.	Should have	Som bruker ønsker jeg informasjon om hvilke hjelpemidler som er tilgjengelig på plassen slik at jeg kan velge plass basert på hvilke hjelpemidler jeg trenger.	2
Bruker må kunne registrere om bruker vil ha mat ved reservasjon av plass.	Should have	Som bruker ønsker jeg å kunne registrere om jeg skal ha mat i kantina slik at kantina kan klargjøre en estimert mengde mat og hindre matsvinn.	4
Oversikt for bruker over egne bookinger.	Should have	Som bruker ønsker jeg å se hvor jeg har booket plass for å huske hvor jeg skal være.	8
Oversikt for bruker over tid på egne bookinger.	Should have	Som bruker ønsker jeg å se når jeg har booket plass for å huske når jeg skal være der.	8
Automatisk overføring av booking til timeplan	Will not have	Som bruker vil jeg at bookingen skal legges inn i timeplanen til deltakerne	10
Automatisk varsling av deltakere registrert på møterom	Will not have	Som bruker som booket rommet vil jeg sende varsler til deltakerne med tid og sted for å informere om hvor og når møtet er.	7
Automatisk påminnelse om møte	Will not have	Som bruker vil jeg få en påminnelse før møtet for å rekke møte.	3
		Sum:	**132**

Figur 3: Systemkrav med estimerings- og prioriteringsliste.

2.4 Risikoanalyse

Det er viktig å kartlegge og beskrive hva som er forskjellige risikoer som kan oppstå under prosjektet, i hvilken grad påvirker de arbeidet og hvor stor sannsynlighet er det for at de oppstår.. Skulle disse hendelsene oppstå kan det være til hinder for å oppnå målene våre, derfor må vi lage en plan på hva vi skal gjøre skulle hendelsen oppstå. Vi har utarbeidet en matrise som gir et overordnet blikk på potensielle hendelser og hvilke effekter de kan ha på prosjektet.

Risiko	Sannsynlighet	Alvorlighet	Anbefalte handlinger
Gruppemedlem blir demotivert.	Liten	Middels	Hold det sosialt, og ha det trivelig under arbeidet. Ta gode pauser, og legge til rette for fritid.
Medlem blir syk og må jobbe hjemmefra	Stor	Liten	Legge til rette for at det syke medlemmet kan jobbe hjemmefra og fortsatt delta effektivt. Unngå mer smitte innad i gruppa.
Medlem blir syk og utilgjengelig.	Middels	Middels	Medlem får slappe av under sykdommen, men bør delta på statusrapporter så mye som det er mulig for å holde seg oppdatert.
Konflikt innad i gruppen.	Liten	Stor	Sette seg ned i plenum og la alle synspunkter få komme frem for å få til en god løsning.
Manglende kunnskap for å få gjennomført utviklingen.	Liten	Stor	Be oppdragsgiver eller andre om hjelp til å få løst problemet, og sette av ekstra tid til å lære det som trengs.

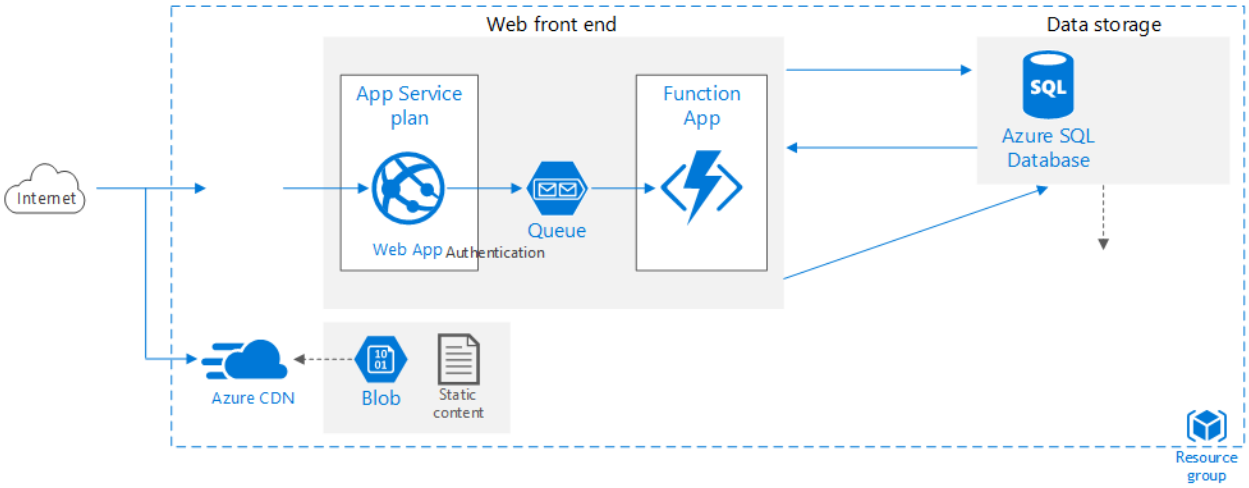
I vårt arbeid som gruppe er det ingen tvil om at sykdom er den største trusselen mot å nå målene våre. Vi har nettopp kommet ut av langvarig pandemi og flere har et immunforsvar som igjen må tilpasse seg normalen, og sjansen for andre sykdommer som influensa er dermed større.

3. Design

Under dette kapitlet tar vi for oss prosessene vi gjennomførte for å komme frem til det endelige utseende av produktet. Dette omhandler blant annet systemarkitektur, navigasjonskart, ER- og klassediagrammer, wireframes og prototype. Designfasen handlet om å bruke dataen vi har samlet inn, og ta avgjørelser om hvordan programmet skal se ut, hvordan arkitekturen er bygget opp, og hvordan klassene og databasetabellene skal bygges opp og jobbes sammen.

3.1 Systemarkitektur

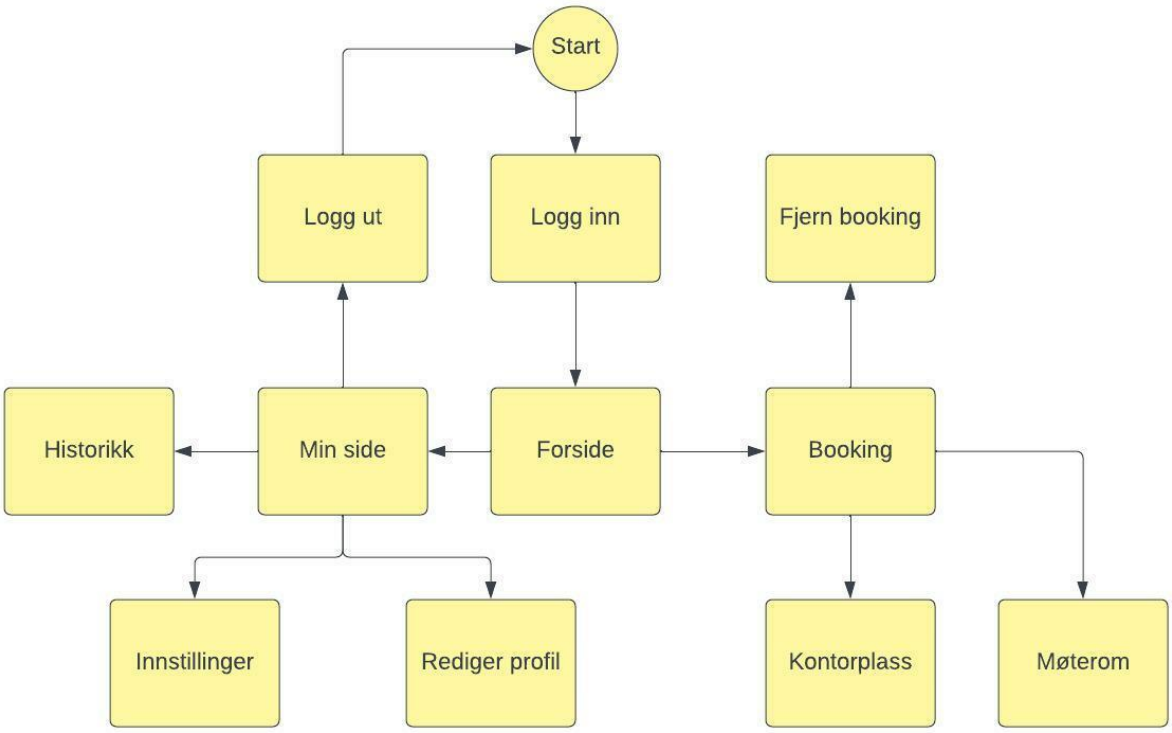
På et møte med kontaktperson gikk vi gjennom bruken av Microsoft Visio, et modelleringsprogram for diagrammer og andre kart. Verktøyet tilbyr ikoner som representerer andre Microsoft-programmer som kan samarbeide sammen gjennom Azure. Ved hjelp av dette fikk vi utviklet en modell for systemarkitekturen. Valgene vi tok var basert på konkrete krav og ønsker fra produkteier, som forklarte sammenhengen mellom programmene underveis i møtet. Figur 4 viser den endelige modellen vi skal basere programmet vårt på. Modellen er en forenklet versjon av Microsoft sin egen modell for en grunnleggende web-applikasjon *(Microsoft, 2022)*. Med Visio kunne vi redigere vekk komponenter i denne arkitekturen som vi ikke trengte å legge vekt på, slik at modellen ble oversiktlig i henhold til vårt prosjekt.



Figur 4: Systemarkitektur

3.2 Navigasjonsdiagram

For å forstå hvordan brukeren skal navigere seg igjennom systemet har vi utviklet et navigasjonsdiagram. Det viser hvordan brukeren beveger seg fra en side til neste for å utføre forskjellige handlinger. Hver side er representert med en firkantet boks i diagrammet. Pilene representerer lenker som sender deg fra en side til en annen. Slik kan vi planlegge hvordan sidene skal henge sammen for en best mulig brukeropplevelse. (British Broadcasting Corporation, 2022)



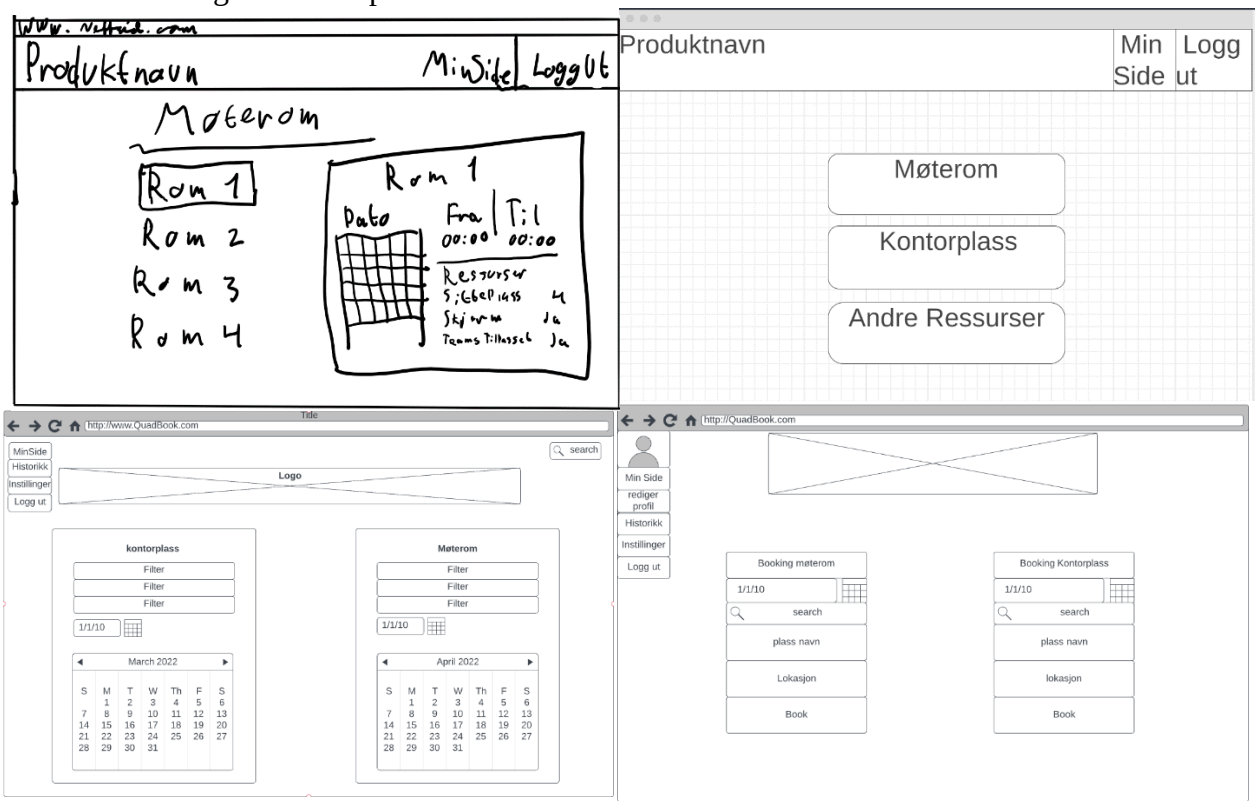
Figur 5: Navigasjonskart

3.3 Wireframes

Wireframes er simple visuelle modeller som grunnleggende forklarer hvordan systemet skal se ut. Disse kan variere mellom low-fidelity sketsjer på papir, til high-fidelity modeller som inkluderer farger og design. Etter vi fikk laget navigasjonsdiagram, hadde vi en oversikt over hvilke sider som skulle være med i programmet, noe som gjorde utviklingen av wireframes lettere. Hvert medlem på gruppa laget sine egne wireframes som representerte hvordan de hadde sett for seg systemet. Etterpå diskuterte vi de ulike modellene og ble enige i fellesskap om

hvordan det endelige designet på systemet skulle se ut. Etter wireframes var ferdig laget, kunne vi begynne og utvikle en prototype, som skal være en nøyaktig representasjon over hvordan det endelige systemet skal se ut. Figur 6 viser noen av våre wireframes, resten ligger under vedlegg. (Babich, 2020)

Enkelte på gruppa ønsket at alle funksjonene på programmet var tilgjengelig fra første og samme side, og at sider som 'min side' skal være tilgjengelige gjennom en oppgavelinje som vises på toppen av alle sidene i programmet. Gruppa ble enige om at denne oppgavelinjen skulle bli implementert i systemet vårt. Det var litt mer uenigheter med hvordan forsiden skulle vare. Resultatet ble at alle ressursene som kunne reserveres var fremstilt i en egen boks på forsiden, som skissert i figur 6 eksempel 3



Figur 6: Wireframes Eksempel 1, 2, 3 og 4

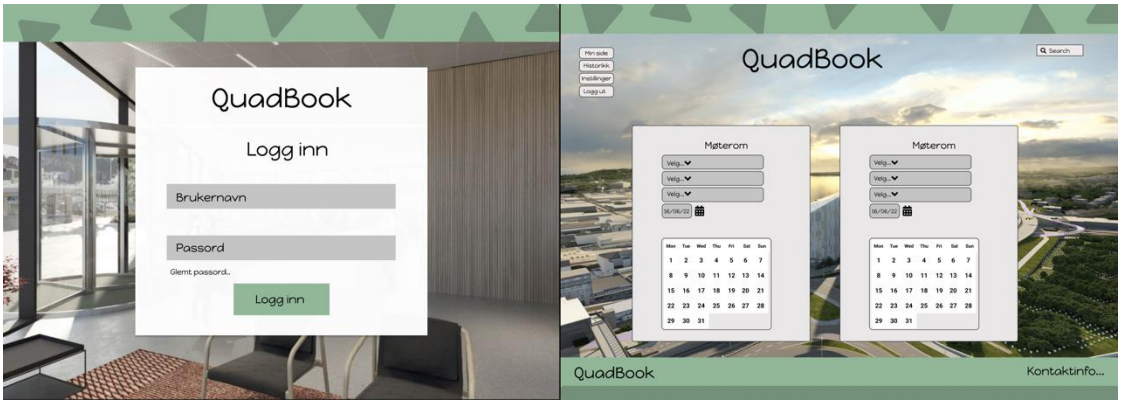
3.4 Prototype

En prototype er en modell som skal visuelt representere det ferdige produktet (Studio by UXPin, n.d.). Dette inkluderer alle fargene og designet som skal være på front-end delen av vårt prosjekt. En prototype hjelper utviklere med å teste og avdekke feil, mangler og forbedringer før vi begynner med utviklingen, slik at utviklingen går så smertefritt som mulig. Prototypen hjelper i

tillegg gruppen med å se hvordan systemet eventuelt skal se ut og kan jobbe ut ifra en felles forståelse om dette. Denne prototypen er laget ved hjelp av Figma, og er utarbeidet av hele gruppa i felleskap, basert på beslutningene fra wireframes. (Figma, u.d.)

Figur 7 viser et eksempel på hvordan siden skal se ut. Fargene vi har valgt er basert på logoen og hjemmesiden til Quadrum, ettersom vi lager dette systemet til dette bygningsprosjektet i første omgang. Grønnfargene til Quadrum er gjenkjennelig for ansatte som jobber i disse byggene, og basert på Benyons designprinsipp, 'familiaritet' vil denne fargen gi inntrykk av at systemet er en del av den digitale infrastrukturen til organisasjonen. Benyons designprinsippene som omhandler «Affordance» og «Consistency» som handler om å lage en lett gjenkjennelig og konsistent nettside hvor bruker er i sentrum, har også vært sentrale under designfasen. Ved å følge disse prinsippene, har systemet oppnådd et mer brukervennlig grensesnitt der interaktive elementer som knapper og menyer følger et oppsett som er gjenkjennelig og lett å bruke. (Benyon, 2017)

Oppdatering 03.05.2022: Omtrent to uker før innleveringsfrist oppdaget gruppa at hjemmesiden til Quadrum har blitt endret. Forandringen er minimal, men dreier seg om temafargen, som har endret seg fra den lysegrønne fargen du ser på figur 7, til en mørkere og tydeligere farge. Denne endringen oppdaget vi etter prototypen var ferdigstilt, men vi fikk endret fargen i det endelige systemet.



Figur 7: Prototype

3.5 Produktnavn

Navnet på produktet er noe som ikke hadde den høyeste prioriteringen, men det var likevel noe som gruppa ønsket å ta seriøst, da et originalt og kreativt navn gir prosjektet vårt et mer personlig preg. Navnet 'BookIT' var omkring de mest diskuterte av navnene, hvor 'IT' er en dobbeltbetydning som IT-entusiaster kjenner igjen. Det endelige navnet vi landet på ble 'Quadbook', som er en blanding av navnet på byggene, og hovedfunksjonen av applikasjonen.

3.6 Entitet-relasjonsdiagram

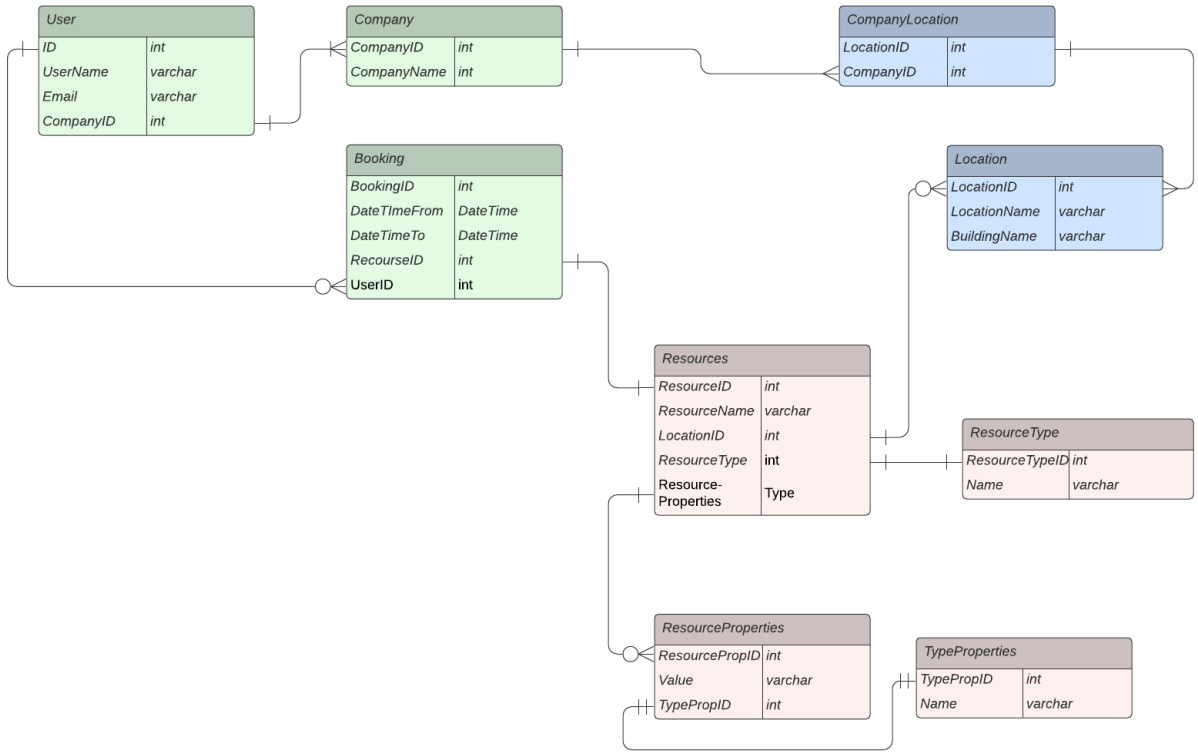
For å planlegge og lage en visuell representasjon av databasen har vi laget en entitet-relasjonsmodell. Den består av entitetene i databasen, entitetenes attributter og relasjonene imellom dem. I vårt tilfelle skal vi lage en relasjonsdatabase. Det betyr at entitetene representerer tabellene i databasen og relasjonen representerer fremmednøkler. (Lucidchart, 2022)

Det er viktig at tabellene og kolonnene har presise og simple navn som gjør det enkelt å forstå hva som skal hvor i databasen. Relasjonene må også ha en sterk relasjon og alle vage relasjoner bør fjernes. For å klassifisere lignende tabeller har vi tatt i bruk farger for å gruppere dem. (Nishadha, 2021)

Fordelen med å lage dette er at vi får en god plan på hva vi trenger i databasen og hvordan den ser ut. Da slipper vi å designe databasen samtidig som vi utfører spørringene som oppretter den, og vi får en god forståelse for videre arbeid i design fasen og kan kontrollere at databasen dekker de behovene vi skal dekke fra brukerhistoriene.

Figur 8 henviser til diagrammet vi satt igjen med. Vi visste at selv om Capgemini hovedsakelig ønsker å kunne reservere møterom, skulle systemet kunne tilpasse seg andre bedrifter, med andre ressurser i andre bygg. Av erfaring vet vi også at det er lurt å modellere så abstrakt som mulig. Det ferdige diagrammet viser hvordan vi løste dette problemet med å kunne tilpasse til andre bygg. Ved å referere til reservasjonsobjektene som 'ressurser' kan bedriftene selv bestemme hvilken ressurs-type dette skal representere. Tabellene 'ResourceProperties' og 'TypeProperties' refererer til hvilke ekstra utstyr disse ressursene har, for eksempel utstyr til digitale møter på møterom, ekstra skjermer på kontorplasser eller antall seter i en firmabil. Slik kan vi fokusere systemet imot møterom for Capgemini, men også lett tilpasse systemet for andre bedrifter.

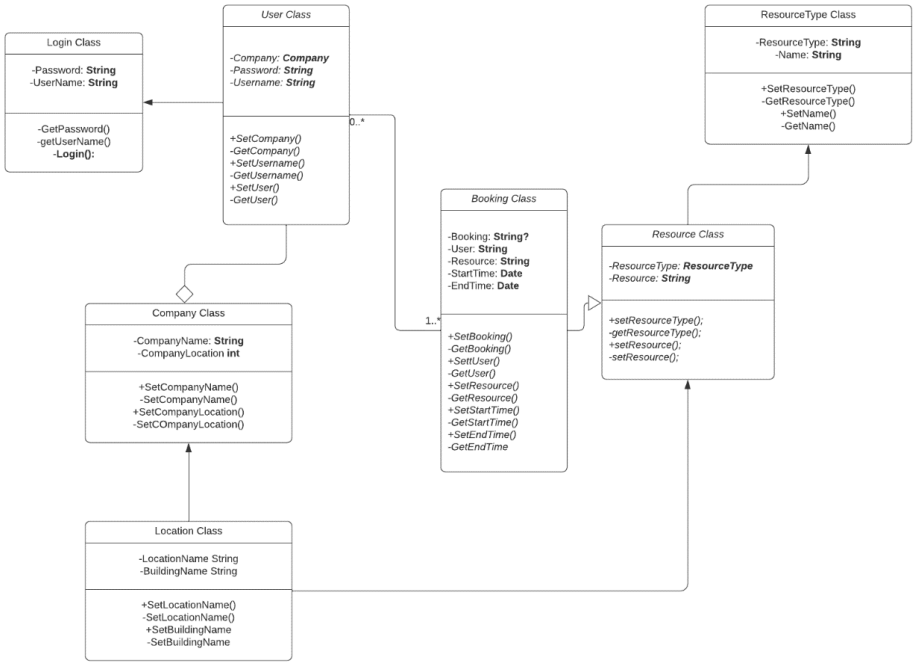
Figur 8 viser ERD diagrammet vi utviklet i samarbeid med bedriften. De ønsket en database som var åpen for å kunne legge til flere tjenester i utlånssystemet og kobles opp mot flere bedrifter, typer ressurser og bygg.



Figur 8 ER-Diagram

3.7 Klassediagram

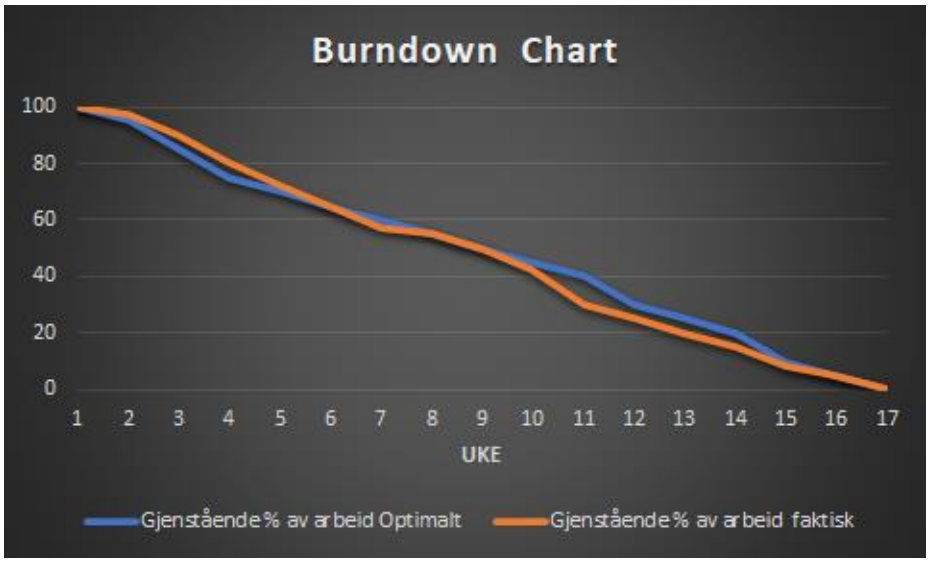
Etter å ha utarbeidet Entitet relasjonsmodellen som viser lagring av data og relasjonene imellom dem, så begynte vi på et klassediagram som viser organiseringen av klassene og metoder i systemet. Klassene kan sees på som abstrakte representasjoner av objekter fra den virkelige verden. Relasjonene viser hvordan disse klassene henger sammen i systemet. Metodene definere oppførselen til de forskjellige klassene. (Indika, 2011)



Figur 9: Klassediagram

3.8 Burndown Chart

Burndown-chartet er en oversikt over antatt gjenstående arbeid som blir sammenlignet med hva vi så for oss var optimalt i starten. Formålet med dette er å overvåke prosjekt-progresjonen for å sikre jevn fremgang med et ferdig produkt på slutten av prosjektet.



Figur 10: Burndown Chart

4. Språk og verktøy

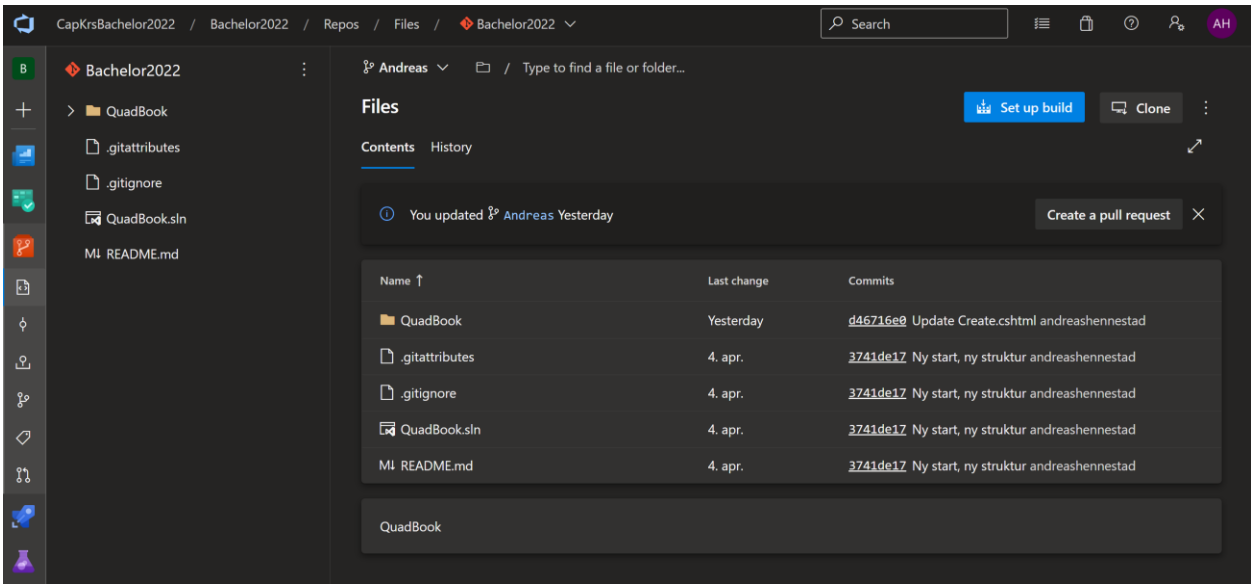
Dette kapittelet omhandler de ulike verktøyene og teknologiene vi har tatt i bruk under dette prosjektet. Gjennom semestrene på Universitetet i Agder har gruppemedlemmene vært innom forskjellige teknologier og programmer. Gruppemedlemmene har alle ulik erfaring innen utviklingsspråk, utviklingsmiljø, samarbeidsplattformer, metodikker og teknikker. Av denne grunn er gruppa svært tilpasningsdyktige, og var åpen til å ta i bruk teknologier som bedriften var vant med å bruke i deres prosjekter. Capgemini var selv veldig åpne på hvilke teknologier vi skulle bruke, men hadde selvfølgelig anbefalinger. Som resultat kom vi frem til å bruke Azure DevOps som samarbeidsplattform, C# (C Sharp) som programmeringsspråk, og ASP.NET for rammeverk for prosjektet. Ettersom dette var alle nye programmer for oss å sette seg inn i, ble vi grundig innført i disse teknologiene av ansatte i Capgemini som bruker disse programmene hver dag.

4.1 Azure DevOps

Azure DevOps er en samarbeidsplattform for utviklere. Det er en del av samlingen teknologier Microsofts .NET og fungerer derfor bra sammen med de andre verktøyene vi skal bruke. I løpet av studietiden har vi brukt programmer som har likhetstrekk med DevOps, som for eksempel Jira og Trello. Azure DevOps tilbyr alle disse programmenes funksjonalitet på ett og samme sted. Utviklere hos Capgemini bruker dette systemet daglig og vi fikk en grundig og kjapp gjennomgang på hvordan vi skulle bruke dette opp mot vårt prosjekt. Deretter fikk vi utdelt et eget miljø for vårt prosjekt som vi kunne bli kjent med på egenhånd.

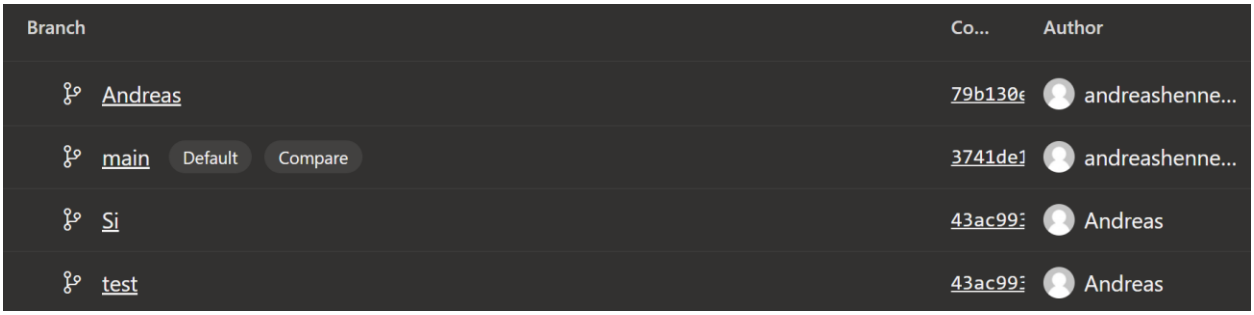
DevOps tilbyr en oversikt som kalles ‘boards’, som gjør det enklere å fordele og strukturere arbeidsoppgaver. I kapittel 2, Analyse, finner man det vi kaller for brukerhistorier og systemkrav. Hvert av disse systemkravene blir skrevet opp på en virtuell tavle under kategoriene ‘ny’, ‘påbegynt’ og ‘fullført’, slik at alle på gruppa for en oversikt over hvilke arbeidsoppgaver som kan gjøres og hvilke som er tatt av andre gruppemedlemmer.

4.2 Versjonskontroll



Figur 11: Skjermbilde fra Azure DevOps

Azure DevOps tilbyr bruk av versjonskontroll i systemet Git, som er kompatibelt med programmene vi bruker som den integrerte versjonskontrollen i Visual Studio og Github Desktop. Her kan vi klone prosjektet og jobbe på en egen sikker gren av prosjektet, som vi senere kan dytte til hoved-grenen. Vi valgte å strukturere disse grenene etter en anbefaling fra vår kontaktperson, visualisert på figur 11.



Figur 12: Oppbygningen av grenene Azure DevOps

Her har vi en hoved-gren som er det fungerende prosjektet. Hovedgrenen har en annen identisk test-gren som vi alle dytter vår fungerende kode til. Når vi er sikre på at test-grenen fungerer optimalt, kan vi dytte det til hoved-grenen. Hvert medlem i gruppa lager en egen gren fra test-grenen hvor de lager en funksjon og dytter tilbake til. På denne måten sørger vi for at alle kan jobbe på prosjektet samtidig og at hovedgrenen som inneholder det endelige fungerende

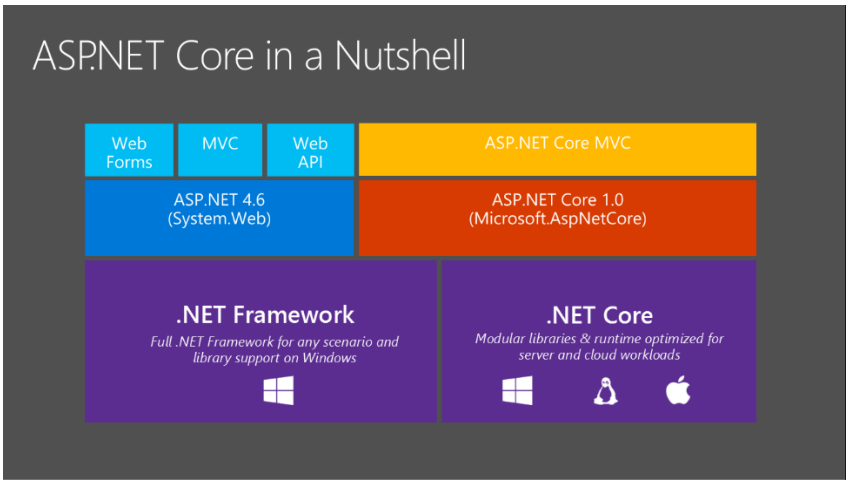
prosjektet, ikke blir ødelagt av uferdig og rotete kode. Denne arbeidsmetoden er blitt anbefalt av Capgemini, samtidig som det er denne metodikken vi har brukt på tidligere prosjekter. Det var derfor et veldig naturlig valg for oss.

4.3 C#

C# er et objektorientert programmeringsspråk utviklet av Microsoft i henhold til ASP.NET. ASP.NET tillater bruk av flere språk, men ettersom C# er tilrettelagt for utvikling innen .NET, har vi valgt å programmere systemet i dette språket. Bedriften anbefalte oss å jobbe med C# da de har god kompetanse på dette selv på kontorene i Kristiansand. C# var et nytt språk for oss på gruppa, men gjennom årene på studiet har vi lært oss logikken bak de fleste programmerings-metodikker, og vi behersker å tilpasse oss flere programmeringsspråk. Brikkene falt på plass ganske kjapt da vi har god erfaring med Java og andre objektorienterte programmeringsspråk (Microsoft, 2022).

4.4 ASP.NET

ASP.NET er ett rammeverk, som skal gi mulighetene til å utvikle dynamiske webapplikasjoner. Det er en del av en større samlinger av teknologier for systemutvikling som inngår i Microsofts .Net pakke. I rammeverket finner vi Blazor som er et nettverk med åpen kildekode som gjør det mulig å lage webapplikasjoner med C#, som et alternativ til for eksempel Javascript. (Microsoft, 2022)



Figur 13: Illustrasjon fra Microsoft ASP.NET

ASP.NET er rammeverk som er innlemmet i Visual Studio og som lar oss bruke Entity Framework Core som er ett rammeverk vi bruker for å sette opp databasen til prosjektet.

4.5 Utviklermiljø: Visual Studio

Utviklermiljøet vi valgte for prosjektet var Visual Studio, som ikke må forvirres med Visual Studio Code. Visual Studio Code er et cross-plattform-utviklerverktøy som støtter nedlastning av utvidelser spesifikt for stråket og programmet man lager. Visual Studio derimot er mer enn bare et utviklermiljø, da det kommer med mange funksjoner, maler og støtteprogramvare for alle slags typer systemer (Williams, 2021). Blant annet er det mye lettere å laste ned pakker som Entity Framework (kap. 4.6), prosjektoppdretter og tilrettelegging for testing og feilsøking. I tillegg er programmet laget for å fungere bra med nettopp ASP.NET, Azure DevOps og C#.

Visual Studio passet best for oss da det gjorde det mye lettere å ta i bruk de verktøyene vi trengte i tillegg til at vi slipper å installere utvidelser selv. Kontaktpersonen vår i Capgemini har også god erfaring med dette utviklermiljøet.

4.6 Database

ASP.Net tilbyr en tjeneste som kalles Entity Framework, som gjør samspillet mellom kode og database lettere å behandle (Microsoft, 2022). Ut ifra hvordan en utvikler er vandt til å arbeide, kan man velge om man vil lage databasemodellen først, og la Entity Framework oppdrette objektene i koden (Code first), eller selv lage objektene og la programmet oppdrette databasen og koblingene (Database first).

4.7 Model View Controller

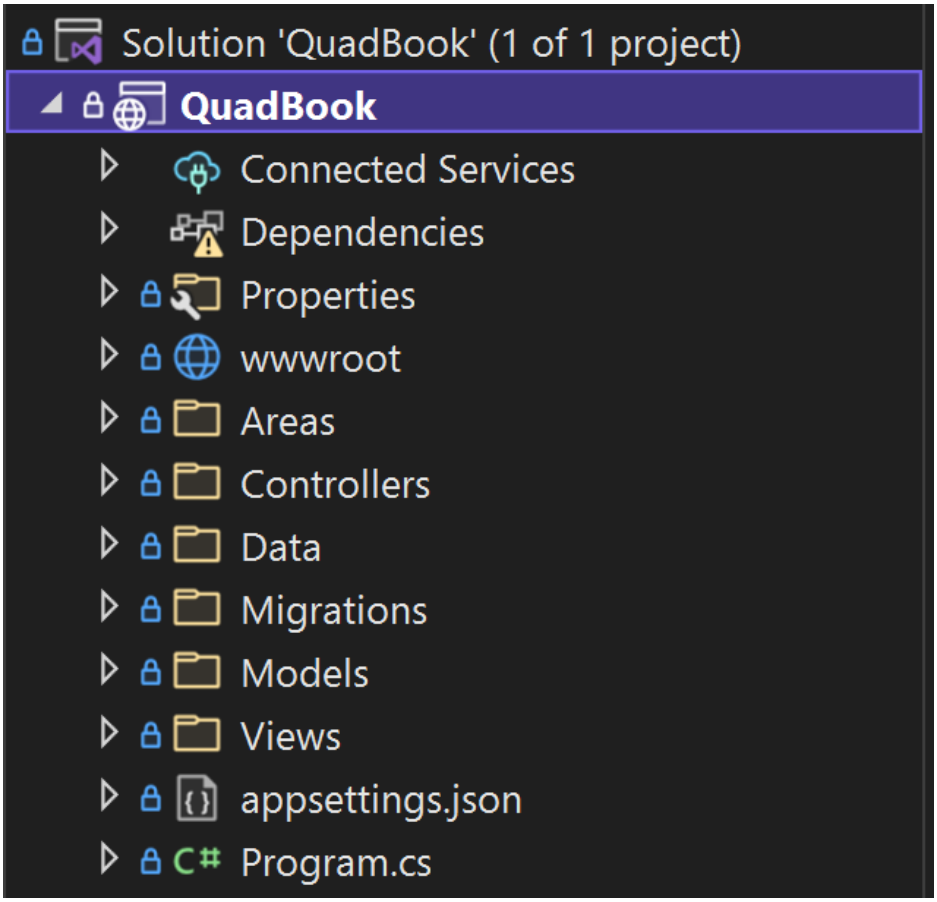
Vi tok i bruk et design-mønster som kalles ‘Model View Controller’ (MVC) som først ble definert av den norske informatikeren Trygve Reenskaug i 1979. (Den norske dataforeningen, 2017) Hensikten med et slikt design-mønster er å fordele komponentene i systemet inn i tre ulike deler som representerer hvilken funksjonalitet filen utfører. Klassene som representerer objekter faller under Model, klassene som behandler all funksjonalitet mellom objektene er Controllers, og klassene som behandler det brukeren skal se, er Views. (Codeacademy Team, 2022)

Vi måtte velge mellom å bruke MVC eller Razor pages som er et annet alternativ. Vi tok i bruk MVC da det er veldig my dokumentasjon om det på nett, og det krever hakket mindre kunnskap for å bruke.

5. Utviklingen av systemet

I dette kapittelet forklares utviklingsprosessen av systemet. Dette innebærer hvilke valg som ble tatt, litt om arbeidsmetoder og tiltak for hvordan vi endte opp med det endelige produktet.

Vi startet med å sette opp et web-applikasjons-prosjekt basert på MVC og Entity Framework Core som genererte utgangspunktet for prosjektet vårt, strukturen så da slik ut som i figur 13 (Codeacademy Team, 2022). Utgangspunktet inneholdt derfor allerede forhåndsgenererte views og deler av controllerne vi skulle bruke. Utviklingen av prosjektet gikk derfor ut på å opprette modeller for databasen, utvide logikken i controllers for å manipulere dataen og hente ut relevante data og presenter dem med views. Vi har også brukt ASP.NET Core sin Identity-innlogging for å ha en innlogging som er sikker.



Figur 14: Filstruktur

5.1 Sikring av kvalitet under utviklingen.

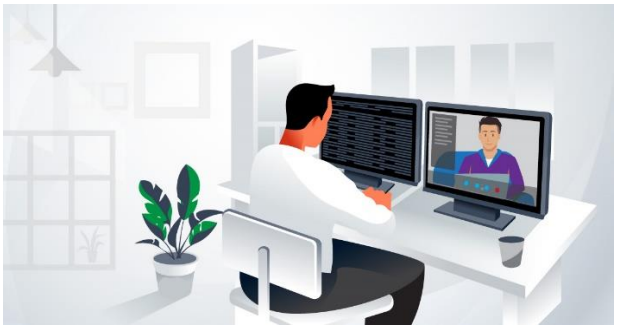
Vi har gjennom hele utvklingsdelen vært opptatt av å både sikre og kontrollere kvalitet. Vi har jobbet forebyggende mot feil og sikret kvalitet ved å benytte en rekke metodikker som parkoding. I selve koden har vi vært opptatt av at andre skal kunne lese den og kommentarer og god navngivning har derfor vært prioritert. Versjonskontrollen har også vært viktig når det gjelde sikring av kvalitet da god og riktig bruk av versjonskontroll har gjort det mulig å kombinere sammen kode utviklet av forskjellige medlemmer uten at det skaper problemer.

For å kontrollere kvaliteten av det vi har laget er god nok har vi hver gang vi ble ferdig med en oppgave i backloggen gått over det i plenum og manuelt testet for potensielle feil.

5.2 Parkoding

I utviklingen har vi benyttet parkoding for å sikre kvalitet, forebygge feil og sørge for fremgang. (Gillis, 2021)Vi har i tidligere programmeringsfag blitt anbefalt å ta i bruk parkoding og har hatt god erfaring med dette tidligere. Gruppen valgte parrkoding for å kunne utnytte evner og synspunkter fra forskjellige medlemmer for å kunne løse de mer avanserte og kritiske oppgaver på en best mulig måte. Målet med dette var å hindre at fremgangen til enkelt medlemmer ble hindret av et problem som ikke var løst samtidig som vi ønsket å hindre feil som senere ville krevd lang tid å fikse senere. Parkoding har vist seg å være veldig effektivt og sosialt i arbeidet med utviklingen. Det har hjulpet oss å løse problemer som vi mest sannsynlig ikke ville klart å løse på egenhånd. Vi oppdaget at det var enkelt å få til både ved å sitte i samme rom (figur 14) og bruke skjermdelingsverktøy over nett (figur 15). Et negativt aspekt med parkoding er at det er ressurskrevende å ha to utviklere som fokuserer på en og samme ting samtidig, men læringsutbyttet og resultatene gjorde opp for tapt tid.

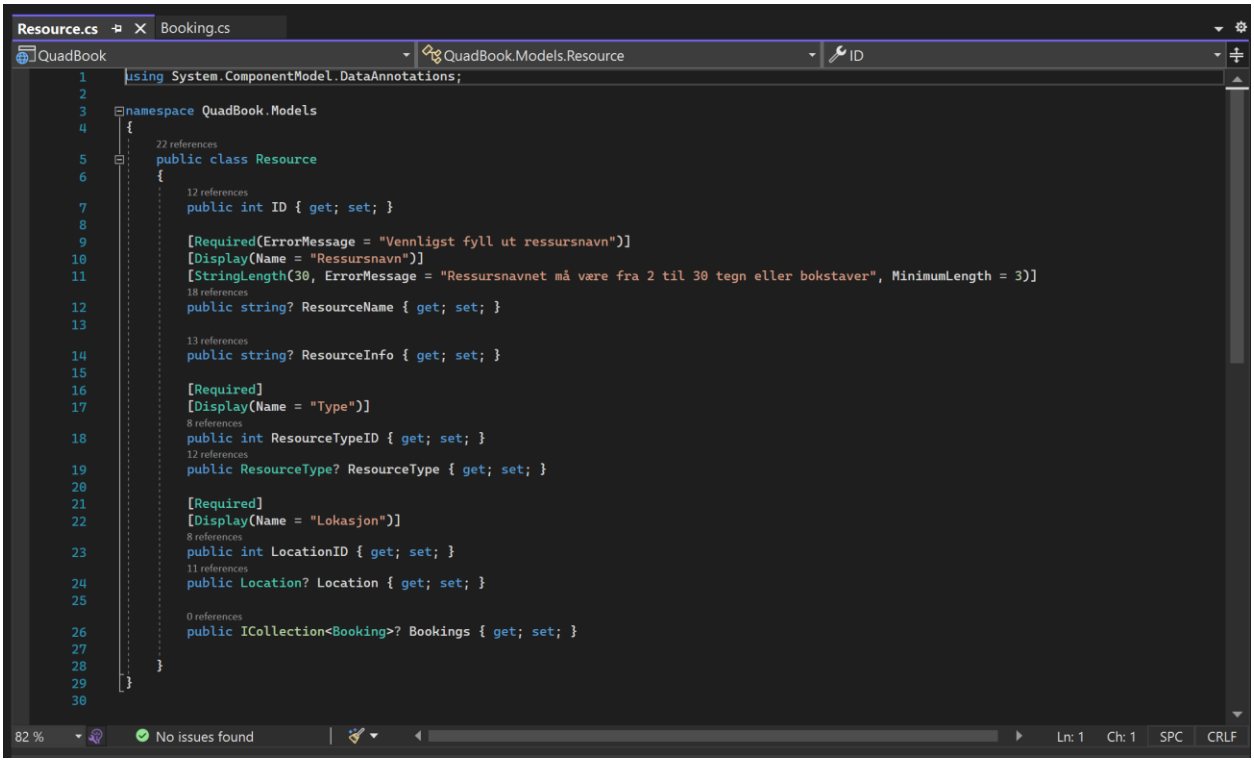
Figur 15: Parkoding i samme rom



Figur 16: Parkoding over nett

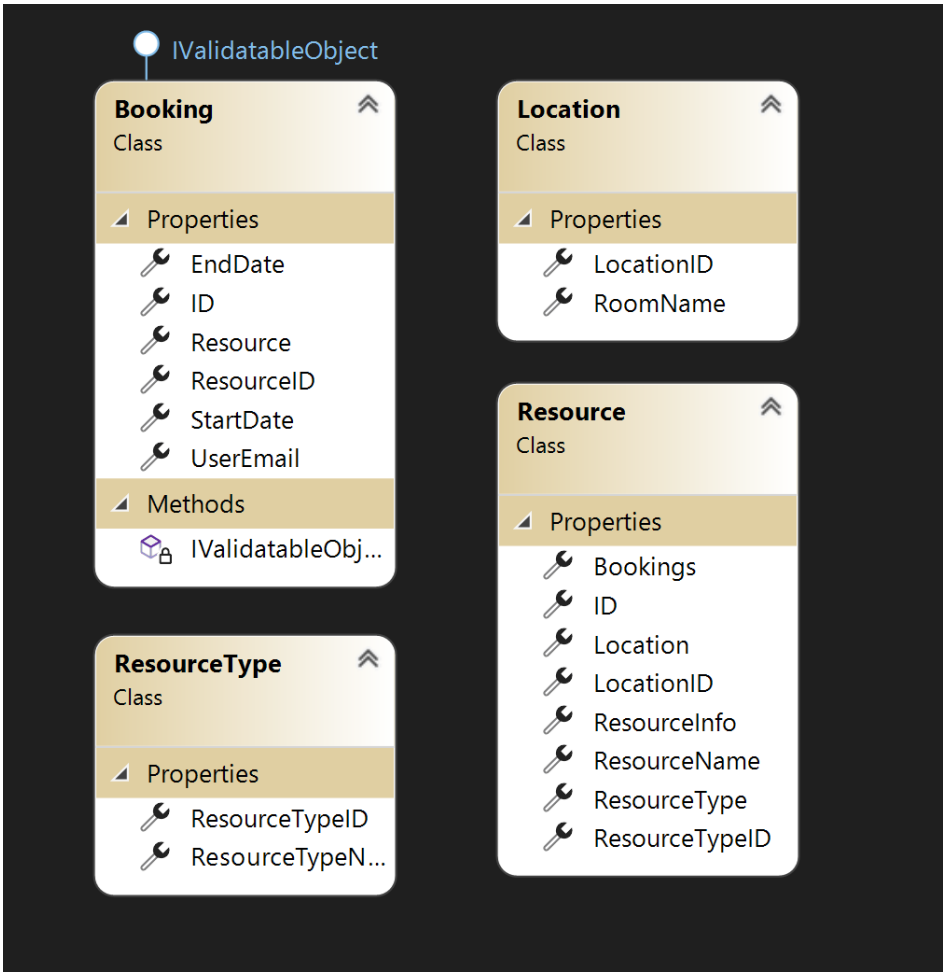
5.3 Entity Framework / Models

Som vist i tidligere kapittelet, hadde gruppa på dette stadiet laget både et klassediagram (kap. 3.7) og entitets-relasjonsmodell (kap. 3.6). Dette gav oss valget mellom å bruke Entity Framework med Code First eller Database First. Vi ble anbefalt og kom til en enighet om å bruke Code First ettersom det ville spare oss for en god del tid og vi ville slippe å håndtere SQL spørringer. Vi lagde klasser i C# i modellene basert på klassediagrammet, som ble lagret under Models. Videre satte vi Entity Framework i gang, som automatisk genererte controllere og views for klassene og satte opp databasen.

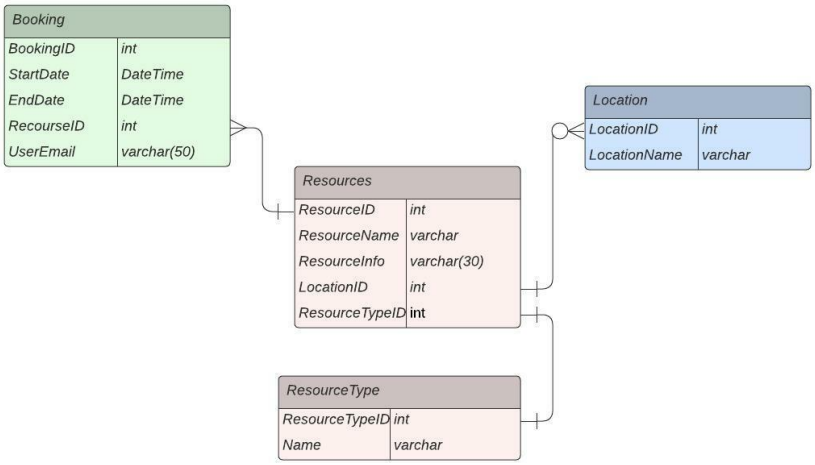


Figur 17: Koden til Resource-klassen

Vi opplevde da problemer med måten databasen var satt opp og så at vi ikke fikk den funksjonaliteten vi ønsket da det i kombinasjon med MVC ikke fungerte veldig bra. Vi måtte derfor omstrukturere databasen slik at den ble simplere og enklere å forholde seg til og sørge for at kvaliteten på systemet blir sikret. Vi reviderte databasen og endte da opp med et klasse- og ER-diagram som vist i figur 17 og 18.



Figur 18: Nytt Klassediagram



Figur 19: Nytt ER-diagram

5.4 Endringer i klassene i databasen

Vi bestemte oss for å fjerne blant annet tabellene for egenskapene til attributtene og heller gjøre dette om til et informasjonsfelt i ressurs tabellen. Etter å ha omstrukturert databasen ble det mindre kode å forholde seg til og systemet ble lettere å bruke enn ved den første database modellen, og vi ble veldig fornøyd med resultatet.

5.5 Controllers og Views

Etter å ha skapt det vi trengte ved bruk av modellene våre kunne vi begynne å legge til funksjoner i logikken som ikke kunne vi bli generert gjennom modellene. Det gjelder funksjoner som å for eksempel ikke kunne booke en ressurs på samme tidspunkt som en annen bruker. Mye av koden i disse var allerede generert. Controller filene håndterer logikken i systemet og View filene håndterer det visuelle. Disse er koblet opp mot hverandre slik at når siden vises, kan det visuelle i viewet ta i bruk logikken i controller.

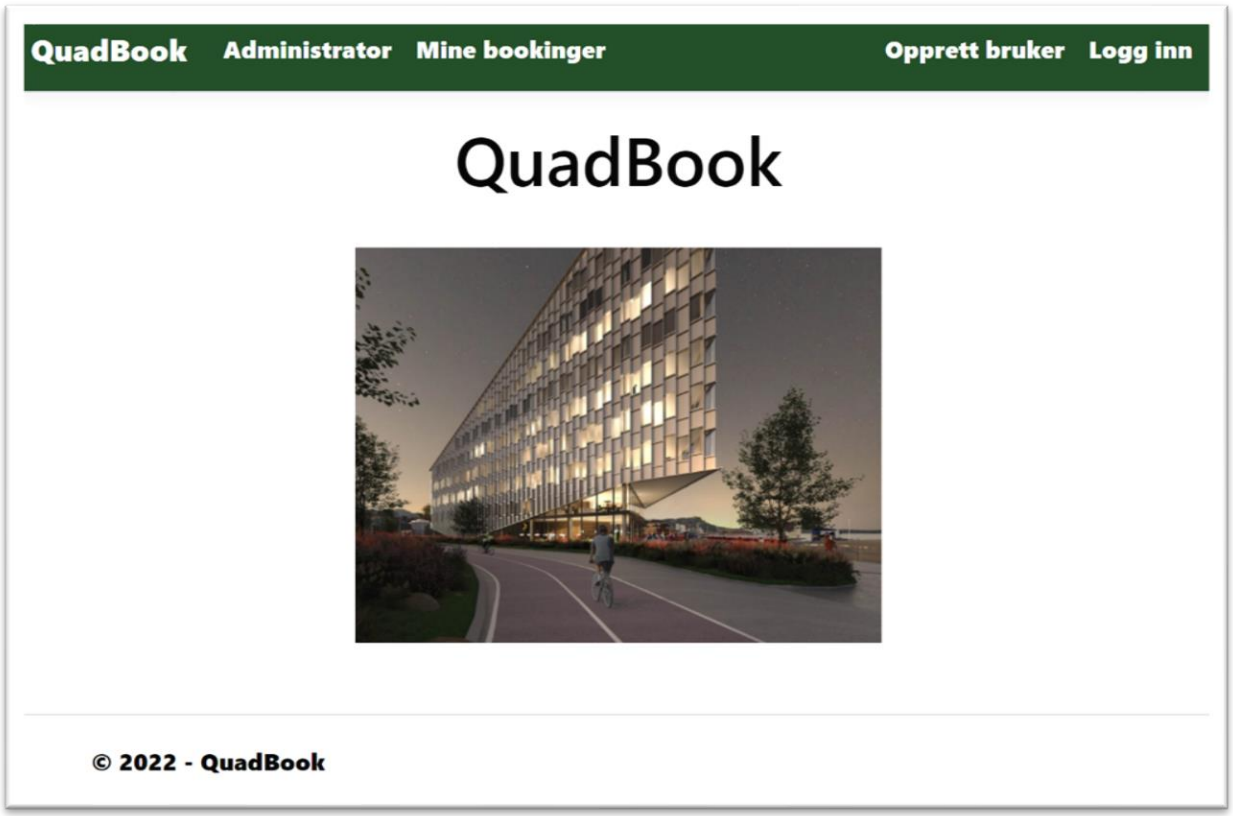
5.6 ASP.NET Core Identity

For å gjøre det enkelt og sikkert å sette opp et brukere for systemet vårt tok vi i bruk ASP.NET Core Identity som er et programmeringsgrensesnitt (API) i ASP.Net som inneholder ferdig utformet bruker og innloggingshåndtering. Vi kan da enkelt legge til brukere og definere forskjellige roller for disse. Rollene kan kobles opp mot autorisering som trengs for å bruke forskjellige deler av koden. Vi kan for eksempel legge til en administrator-rolle og definere at spesifikke deler av systemet kun kan brukes av administrator rollen, slik som håndtering av

ressurser og bookinger. Ved å ha gjort det på denne måten er det også lett å koble innloggingstjenesten opp mot enkeltpålogging (SSO) slik at brukere kan logges inn med tjenester som Azure og Google.

5.7 Visuelt

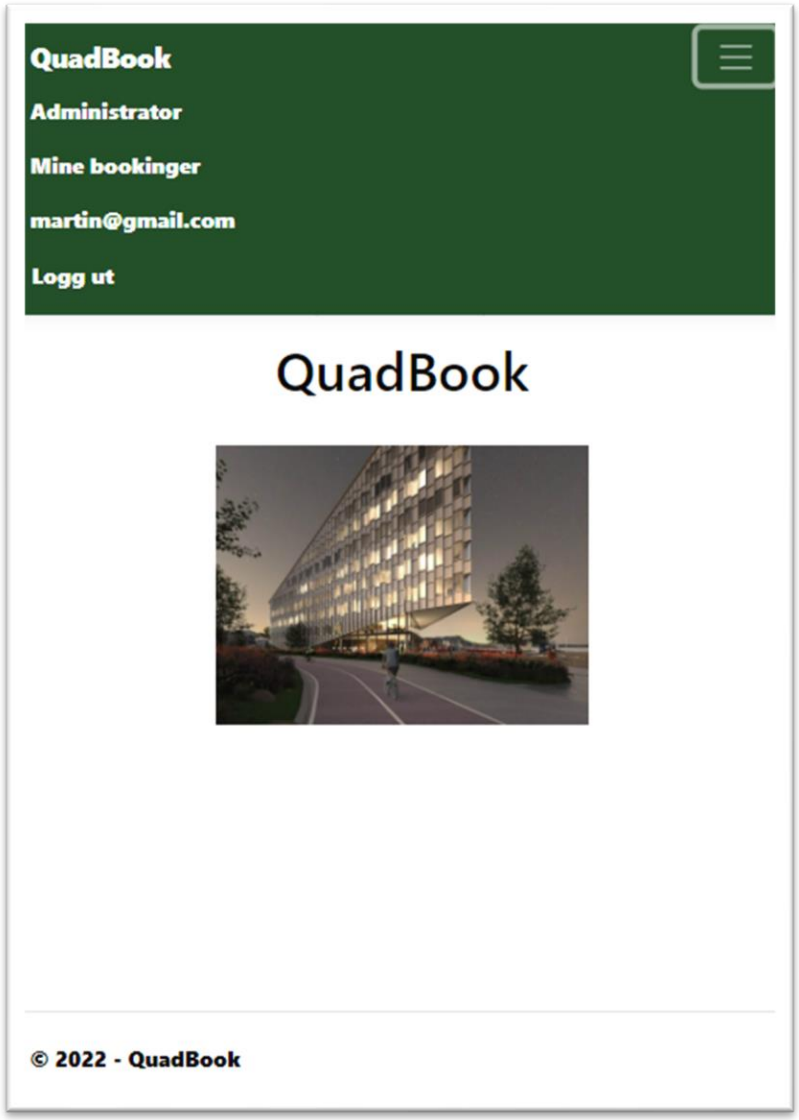
Vi ønsket å holde noe av samme fargetema som Quadrum sine egne nettsider på samme måte som vi gjorde i Prototypen. Rammeverket hadde generert mye for oss og vi kunne derfor tilpasse disse slik at vi fikk de fargene og teksttypen vi ønsker gjennom å endre på prosjektets CSS fil. Vi har holdt utformingen av det visuelle familiært ved å se på hvordan andre lignende kjente nettsider har utformet plassering av for eksempel knapper og informasjon, slik at brukere ikke bruker unødvendig tid til å bli vant til systemet.



Figur 20: Startsiden av systemet

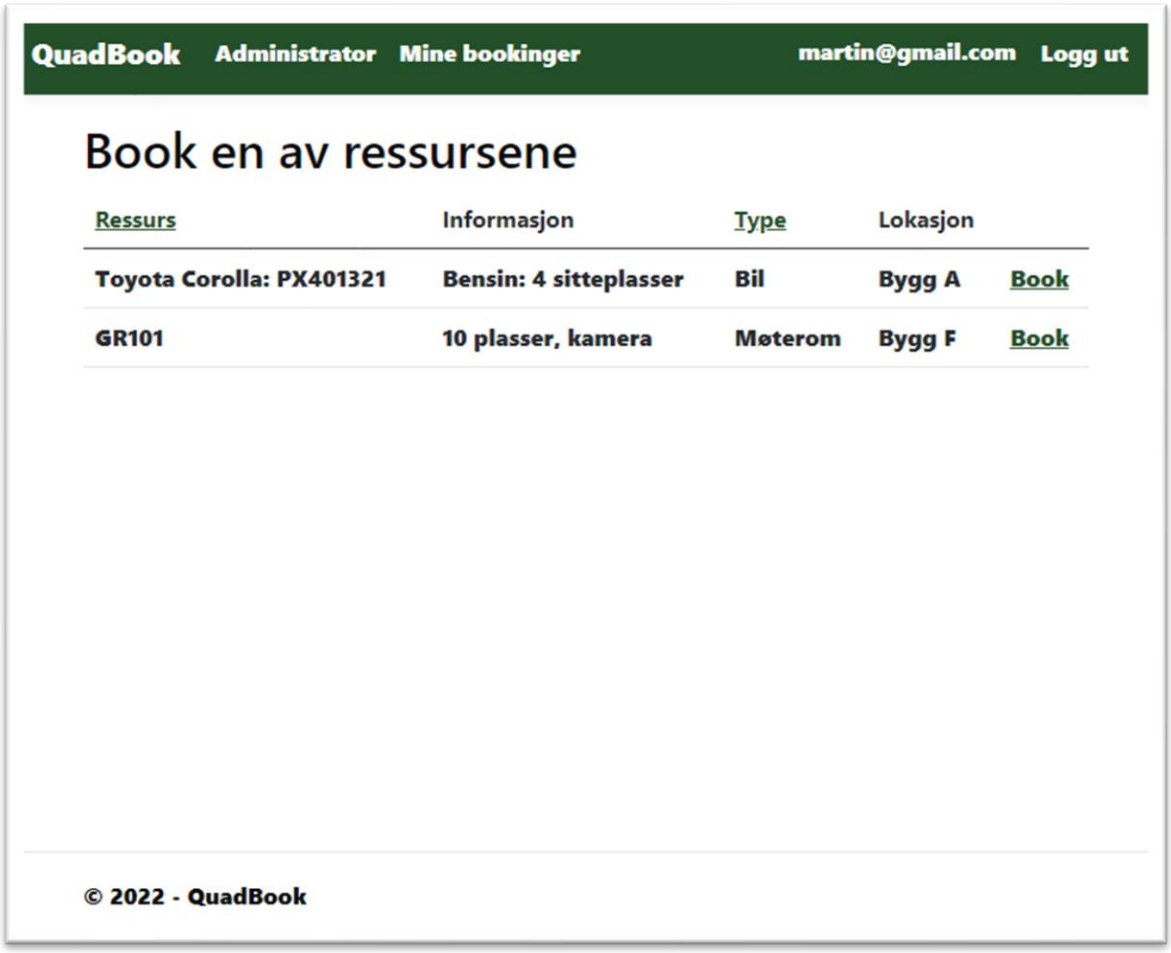
Systemet har blitt utviklet som en webapplikasjon som helst skal brukes med en datamaskin eller bærbar PC. Det kan også brukes på en mobiltelefon da det meste skalerer seg selv til å passe vindustørrelsen, navigasjonsbaren i toppen vil for eksempel krympe sammen til en drop down

meny på en mobiltelefon. Det kan nok tilpasses enda bedre for mobiltelefoner skulle det være ønskelig.

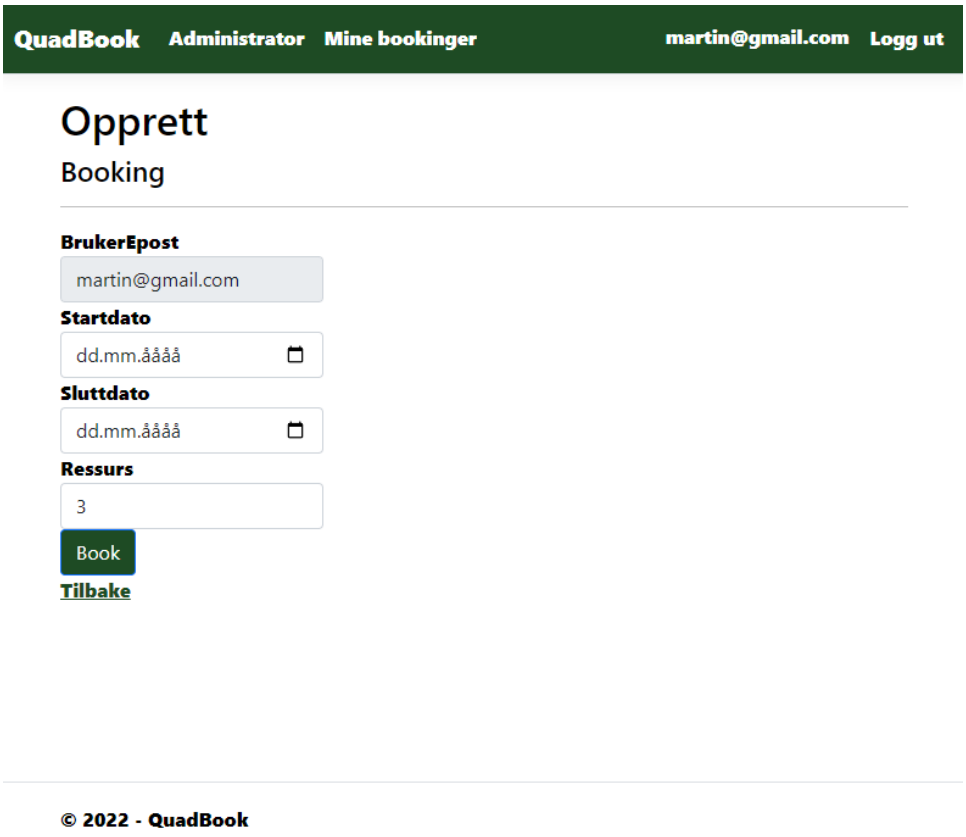


Figur 21: Drop-down meny på mobil enhet

Da vi etter designfasen oppdaget at rammeverket kunne generere mye av koden til det visuelle for oss innså vi at vi kunne spare mye tid på å beholde mye av det som ble generert, og at det til og med passet formålet til systemet veldig bra. Vi gikk derfor litt bort fra designet når det kommer til å velge datoer i fra prototypen.



Figur 22: Oversikt over ressurser



Figur 23: Oppdretting av ny ressurs

6 Prosjektstyringsmetodikk

For å gjennomføre et systemutviklings prosjekt er det en fordel å velge en metodikk som skal styre gjennomførelsen av prosjektet. Det finnes både rigide og agile metoder å velge mellom, og det er derfor viktig å finne den metodikken som passer vårt mål best.

Fossefall (Waterfall) metoden er en veldig tradisjonell og rigid metode å styre et prosjekt på. Det setter krav til klare mål, at produkteier vet nøyaktig hva de vil ha og at prosjektet er forutsigbart. Det heter fossefall da det kan sammenlignes med en foss som renner nedover mot målet, og når du har gjennomført en del er det ingen vei tilbake, på samme måte som det er vanskelig å svømme opp en foss. Det er derfor heller ikke store muligheter for å gjennomføre endringer underveis. Fossefall metoden passer derfor ikke for oss da vi har et behov for å kunne være åpne for forandringer og at de andre partene skal ha mye innsyn i prosjektet. (Teamwork.com, 2022)

Som et alternativ til rigide metodikker som Fossefall finnes det vi kaller agile metoder som Scrum, Kanban og Lean, og blant disse er det nok flere metodikker som kunne passet oss. Agile

metoder åpner for flere endringer underveis dersom det skulle være behov for det, og det lar oss gjøre endringer i utviklingen så lenge vi fortsatt oppnår det oppdragsgiver ønsker. Med agilitet kommer frihet, og med frihet kommer ansvar. Det er derfor viktig med agile metoder å sørge for at alle deltakere er motiverte til å bidra og sette seg mål, og at prosjektet klarer å takle endringer underveis. Agile metoder har vist seg å være en god arbeidsmetodikk for gruppen, og har vært et godt hjelpemiddel til å sikre kvalitet i arbeidet.

6.1 Scrumban

Scrum er en agil utviklings metodologi som egner seg spesielt bra innenfor utvikling av programvare i prosjekter. Scrum er spesielt egnet for prosjekter der kunden ønsker innsyn i prosessen og kan følge med gjennom hele prosessen. Som passer bra for oss som skal ha innsyn i prosjektet fra både Capgemini og Universitetet. Scrum er tilpasningsdyktig, fleksibelt, raskt å gjennomføre og regnes som et smidig og effektivt rammeverk å jobbe med. (Digite, 2022)

Scrum-metodikken handler om å fordele arbeidstiden inn i mindre sprinter, hvor hver sprint skal levere et ferdig produkt eller funksjonalitet (Scrum.org, 2022). Etter hver sprint, begynner man med første fase og gjør fasene på nytt. Dette gir oss muligheter angående tidsestimering, prioritering og utvikling. Dette delkapittelet vil ta for seg hvordan scrumban har blitt implementert og hvordan gruppen har brukt scrumban gjennom hele prosjektet.

Scrum utføres slavisk, og alt skal følges nøyaktig etter teorien, men etter flere gruppearbeid i løpet av studietiden har vi kommet frem til at denne prosessen bytter ut mye tid med lite resultater. Derfor har vi tilpasset systemet litt til våre preferanser, samt kombinert det med en metodikk som kalles Kanban. Derfor har gruppa en mye løsere tilnærming til Scrum enn det teorien krever, med mindre tid på møter, og mer tid på faktisk arbeid. Kanban åpner for mer frihet rundt valg av oppgaver i backloggen og passer derfor mye bedre for oss som har mindre erfaring med systemutvikling og har behov for mer fleksibilitet. (Radigan, 2022). Gjennom dette prosjektet har gruppen valgt å bruke scrumban som arbeidsmetodikk, som er en kombinasjon av Scrum og Kanban. En annen faktor som har spilt inn på bruken av Scrumban er at Capgemini allerede bruker Scrumban i arbeidshverdagen og har god erfaring med dette. De har brukt dette i flere år og har derfor luket ut vanlige feil, samt optimalisert det når det gjelder å bruke det i større prosjekter.

6.2 Gjennomføring av sprinter

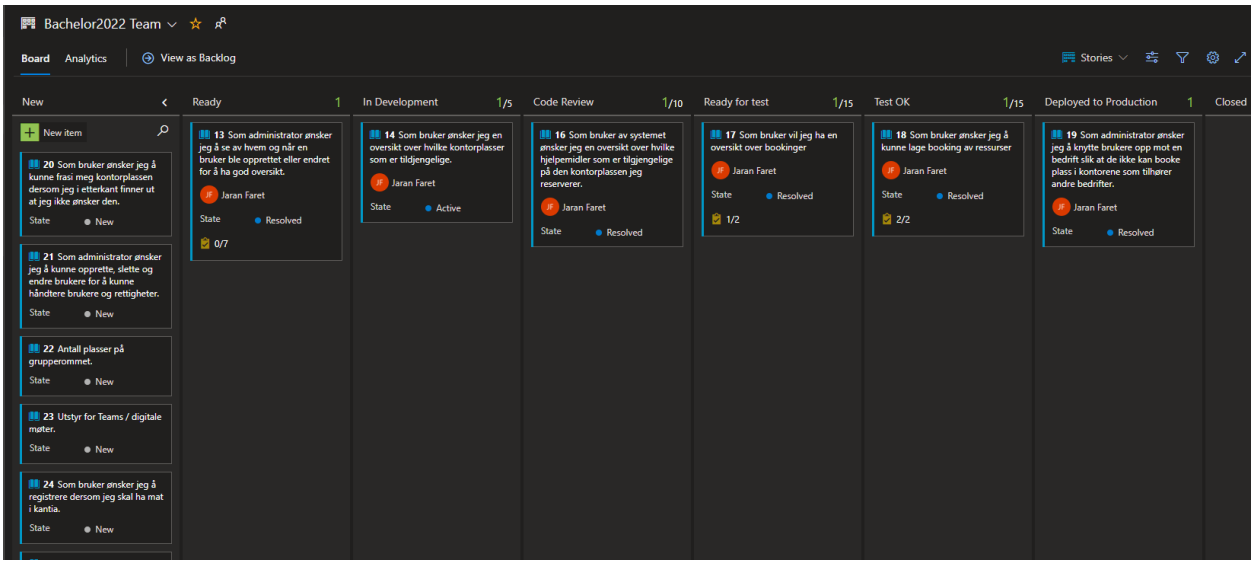
Scrum gjennomføres vanligvis ved å dele prosjektet opp i flere sprinter. Disse sprintene starter med en pre-sprint der man går gjennom arbeidet gruppen skal gjennomføre i den valgte tidsperioden. Deretter har man «daily scrum» der alle kort forklarer hvordan de ligger an med sine arbeidsoppgaver, her kan man også forhøre seg med medarbeiderne sine dersom der noe man skulle trenge hjelp med eller lurer på. På slutten av sprinten har man et sprint-review møte der man går gjennom sprinten, hvilke oppgaver som ble gjennomført, hvilke som ikke ble ferdigstilt og en gjennomgang av gjennomføringen. Her ser man også på hva som fungerte i hva som ikke fungerte helt som man ønsket.

6.3Sprints I Azure Devops

Ettersom vi fikk opplæring fra Capgemini i Azure DevOps, ble vi forklart hvordan de jobber med dette. Figur 24 er et skjermbilde fra hvordan dette brettet så ut, hvor hver kolonne har en egen hensikt for kvalistetssikring i systemet. Vi fikk ikke like mye bruk for alle kolonnene i systemet, men vi ble bedre kjent med hvordan denne metodikken fungerer i arbeidslivet. Måten vi blandet dette med Scrum var at vi diskuterte i sprint-planlegging hvilke systemkrav som var viktigst, basert på prioriteringslisten, og satt en ‘Active’-status på de systemkravene vi mente var overkommelige for hver sprint.

Hvert systemkrav vi satt opp i figur 3 ble satt inn som en oppgave på dette brettet, og i begynnelsen av hver sprint gikk vi gjennom hvilke av disse oppgavene som hadde høyest prioritering og hvor mange som var overkommelige i løpet av perioden vi hadde satt av til sprinten. Disse fikk da statusen ‘Active’. Da systemkravet var satt under ‘Active’ kunne et gruppemedlem overføre dette til ‘In Development’ som betydde at den personen foreløpig jobbet på den oppgaven. Dette er visualisert på Azure DevOps, hvor vi har satt opp samtlige systemkrav

i en slik modell, som vist på figur 19.



Figur 24: skjermbilde fra Azure DevOps

De resterende kolonnene er mer aktuelle i større team hvor de har egne folk til å teste, lese gjennom og rette på, noe vi selv hadde ansvaret for i dette prosjektet. I den siste kolonnen blir oppgavene lagt dersom de er helt fullstendig ferdige og klare til å pushest til main-grenen i systemet. Vi arbeidet med utgangspunkt at fullførte oppgaver ikke skulle pushest til hovedgrenen en og en, men at vi ventet til en bestemt dag hvor vi pushet alle nye endringer. Dette er samme metode som større programvarebedrifter benytter, for eksemepel Microsoft (Microsoft, 2022)

6.4 Pre-Sprint

Da vi fikk utdelt denne oppgaven var planen at bedriften skulle flytte inn det nye bygget tidlig 2022, og gruppa kunne møte opp få tildelt et møterom. Før den tid fikk vi råd fra kontaktpersonen vår om å lese oss opp på dokumentasjon om Azure DevOps, Visual Studio og C#. Frem mot innflyttingen arbeidet gruppa for det meste selvstendig med å lese og forstå disse teknologiene. Det var stort sett ikke mer å ta tak i før bedriften fikk flytte inn.

6.5 Sprint 1

Da bedriften fikk flyttet inn på de nye kontorene, var det bare å begynne på prosjektet. Første sprint gikk med på å planlegge prosessene vi måtte gjennom for å sikre kvalitet på produktet. Vi ble enige om hva vi skulle gjøre for å samle inn data, hvilke modeller vi skulle lage for å komme frem til systemarkitekturen og hvordan vi skulle finne på designet til produktet.

6.5.1 Sprint Planning

Første sprinten gikk ut på hvordan vi skulle angripe dette prosjektet. Gruppemedlemmene skulle fortsette med individuelt arbeid med opplæring i teknologier i første omgang. Samtidig skulle vi forsøke å utføre et intervju med huseier til de nye byggene. Capgemini Kristiansand fungerer for oss som både oppdragsgiver og produkteier, og vi hadde allerede fått en grundig forklaring på hvordan de ønsket at systemet skulle være, men vi ønsket på dette tidspunktet et annet perspektiv til informasjonssanking. Videre i sprinten skulle vi bryte ned prosjektforklaringen i mindre systemkrav og utarbeide en modell for systemkrav. Gjennom et tips fra kontaktpersonen, satt vi som mål og bruke planning poker til å estimere tidsbruket for utviklingen av systemkravene. Det siste målet var å utvikle en prioriteringsliste for systemkravene for å finne ut av hvilke systemkrav vi skulle begynne på først.

6.5.2 Daily stand up

På Radioheads er det to medlemmer som har deltidsjobb, men arbeidstidene er vanligvis på ettermiddagen og kveldstid. Av den grunn var hele morgenen og formiddagen ledig til å jobbe sammen på. Vi ble enige om at så langt det lar seg gjøre, skal vi møtes hver ukedag, enten på kontoret til Capgemini, digitalt på Discord eller på universitetet. På denne måten fikk samtlige på gruppa med seg hva som ble gjort fra dag til dag, og det ble et fordelt arbeidsutbytte. Dersom et gruppemedlem ikke kunne møte opp, gav de beskjed, og måtte deretter fortelle gruppa hva vedkommende hadde gjort siden sist på deres neste oppmøte. Vi mente det var et lurt valg å møtes så ofte, spesielt i begynnelsen da det var viktig at hele gruppa hadde oversikt over valgene som ble tatt.

6.5.3 Sprint Review

Første Review-møte ble holdt på kontoret sammen med kontaktpersonen. Vi viste arbeidet som hadde blitt gjort til nå, og hva planen var videre. I løpet av møtet fikk vi blant annet tips om estimeringen, og at det etter hans erfaring alltid tar lenger tid enn man tenker det tar. Derfor anbefalte han å ta den originale estimeringen vi hadde blitt enige om, og gange det med 3.14(PI) for en mer nøyaktig estimering. Videre viste gruppa systemkravene som hadde blitt brutt ned til mindre oppgaver.

6.5.4 Sprint Retrospective

Etter endt sprint hadde gruppa et møte for å diskutere hva som hadde gått bra og hva som kunne gått bedre. Dette er et viktig møte for å følge Scrum da det gjør det lettere å vite hva som var vanskelig, hva vi skulle prioritere og hvordan vi skulle angripe neste sprint.
Det viste seg at innflyttingen til Capgemini tok litt lenger tid enn planlagt, og det grupperommet vi hadde blitt lovet på de nye kontorene var ikke møblert. Derfor ble det for det meste jobbing på universitetet.
Et mål vi hadde satt oss var å få gjennomført et intervju med huseier. Det viste seg at også han var svært opptatt med innflyttingen av hele bygget, og vi fikk ikke satt opp et intervju før i midten av neste sprint. Dette hindret datainnsamlingen litt, men vi hadde fortsatt oppgavebeskrivelsen fra produkteier, og kunne bruke denne dataen til å utvikle mindre systemkrav. Vi lagde brukerhistorier og systemkrav basert på oppgavebeskrivelsen, og brukte Planning Poker for å estimere tidsbruket på hvert systemkrav. Vi kunne også begynne på en estimeringstabell ved hjelp av MoSCoW-metoden, men denne ville ikke være helt fullført før vi får huseier sitt perspektiv på produktet.

Vi merket tidlig i prosessen at det var høyere entusiasme for å begynne selve programmeringen av programmet, da de fleste på gruppa har høy interesse for utvikling, og ønsket å teste ut nye ferdigheter i C#. Det viste seg at det ikke var den beste prioriteringen å begynne på koden før vi hadde etablert hvordan systemet skal henge sammen. Derfor ble vi enige om at hovedfokuset på neste sprint skulle basere seg på utviklingen av navigasjonskart, klassediagram og entitets-relasjonstabell.

6.6 Sprint 2

6.6.1 sprint planning

I sprint 2 begynner selve arbeidet for fullt i prosjektet. I denne sprinten skal gruppen begynne med arbeidet rundt klassediagram, lo-fi wireframes og gjennomføre et intervju med byggeier som i prosjektet er vår kunde. Gjennom dette intervjuet forventer gruppen også å få et enda klarere bilde av systemet og hvordan kunden ønsker utformingen og være, samt hvilke funksjoner som er forventet i systemet. Det er også planlagt et møte med kontaktpersonen vår i Capgemini der hovedfokuset vill være å se på systemarkitekturen og hvordan det skal bygges opp. Etter dette intervjuet skal vi på nytt se på systemkravene, estimeringslisten og

prioriteringslisten med mer data fra et annet perspektiv. I løpet av denne sprinten har vi blitt lovet en leksjon i hvordan man effektivt kan bruke Azure DevOps, noe som betyr at vi ganske snart kan begynne på utviklingen av prosjektet.

6.6.2 Daily Stand-up

Vi valgte å fortsette med like mye oppmøte som på forrige sprint, da dette viste gode resultater. I utgangspunktet skulle gruppa møtes hver dag på skolen, fortelle hva som har blitt gjort siden sist, og planlegge hva som skal gjøres denne dagen. Dersom det ikke passer for noen å møte opp en dag, sier vedkommende i fra om dette snarest mulig, og får noen oppgaver å gjøre hjemmefra. Gruppa har bevisst valgt å ikke ha noen form for straff eller konsekvens av å komme for sent eller glemme å si ifra, da tidligere prosjekter ikke har vist et behov for dette. Alle gruppemedlemmene tar oppmøtet alvorlig og gjør sitt beste for å sørge for å møte opp, eller si ifra tidligst mulig dersom man ikke kan møte opp. I det tilfellet ville resterende gruppemedlemmer utlevere noen arbeidsoppgaver som er passende.

6.6.3 Sprint Review

Etter sprint 2 gjennomførte gruppen et sprint-review møte med kontaktpersonen i Capgemini der vi sammen gikk gjennom de forskjellige gjøremålene gruppen selv satt i starten av sprinten. Her ble det satt mye fokus på klassediagrammene ettersom de er viktige for å ha god systemarkitektur senere i prosjektet. I dette møtet hadde vi også videre opplæring i Azure DevOps og hvordan man kan bruke det effektivt i et prosjekt.

6.6.4 Sprint Retrospective

Etter sprint 2 var ferdig, hadde vi enda et møte for å diskutere denne og neste sprint. Planen hadde gått bedre denne sprinten. Vi fikk utført intervju med huseier tidlig i sprinten, og hadde god tid til å se tilbake på systemkravene og tilpasse for et annet perspektiv. Videre hadde vi møte med kontaktperson for å diskutere utviklingen av entitetsmodellen. I løpet av dette møtet fikk vi ferdigstilt den modellen som er vist i Figur 8, som fungerte som et godt grunnlag til utviklingen av klassediagram. Etter at gruppa hadde gjennomgått bruken av Azure DevOps, førte vi inn alle systemkravene inn i programmet. Herfra kunne vi nøyere planlegge hvilke systemkrav som vi skulle fokusere på hver neste sprint.

Videre hadde gruppa et møte hvor hvert medlem individuelt lagde lav-fidelitet wireframes om hvordan systemet skulle se ut. Herfra ble vi enige om hvilke ideer som var mest passende og

lagde grunnlaget for prototypen. Siste del av sprinten gikk ut på å sette opp mappestrukturen til prosjektet og sette opp versjonskontrollen på Azure DevOps.

6.7 Sprint 3

6.7.1 Sprint Planning

I denne sprinten er det planlagt å komme skikkelig i gang på utviklingen av systemet. I begynnelsen av sprinten gikk gruppen gjennom hvilke systemkrav som var overkommelige i første omgang, og valgte ut noen oppgaver som vi mente var et godt grunnlag til videre utvikling av systemet. Samtidig skal vi utarbeide en prototype som skal symbolisere grensesnittet til systemet. Dersom denne prototypen blir fullført og godkjent av kontaktpersonen, er alt forarbeid til prosjektet fullført, og gruppen kan sette hovedfokuset mot programmering og utviklingen.

6.7.2 Daily Standup

Som nevnt tidligere har gruppa drevet med parkoding under utviklingen av systemet. Derfor ble vi enige om at vi ikke trengte å møte sammen hver eneste dag. Det var ikke like stort behov lenger for at alle satt sammen og diskuterte alle beslutninger som ble tatt. Vi valgte å jobbe mer digitalt enkelte av dagene, men bestemte oss for et fysisk møte minst to ganger i en hverdagsuke for å presentere hva vi hadde gjort og hvordan det hadde gått. Noe gruppen ønsker å sørge for er et likest mulig læringsutbytte og forståelse av systemet. Under disse møtene ble det altså satt av en del tid til å forklare koden og hva den gjør slik at gruppa har en felles forståelse for systemet.

6.7.3 Sprint Review

I denne sprinten har vi satt mye fokus på kodingen ettersom vi nå endelig har begynt med det. Her har gruppen kommet godt i gang med å sette opp databasen og mappestrukturen for systemet. Dette ble vi ferdig med og har derfor nådd hovedmålet vi satt oss for sprint 3. I denne sprinten har vi også begynt å utvikle en prototype for systemet som gir oss og kunden et bedre innsyn i hvordan vi ser for oss systemet. Her har vi også kommet godt i gang, men mangler fortsatt litt før den er helt ferdig. Dette var forventet og planen er å ferdigstille prototypen i løpet av neste sprint.

6.7.4 Sprint Retrospective

Retrospective-møtet ble holdt som vanlig etter sprinten var ferdig. Tredje sprint hadde ikke gått så bra som vi planla, da det ble vanskeligere å planlegge separat arbeid. Vi merket at arbeidsmengden og effektiviteten gikk ned når vi ikke lenger hadde eksternt press fra resten av gruppa. Samtlige gruppemedlemmer merket at dette var ikke bevisst, men at det ble en naturlig økning i arbeidsmengden av at alle satt i samme rom og arbeidet. Det ble vi enige om å gjøre videre resten av prosjektet. Ellers hadde vi som mål å utarbeide en prototype, noe som ble gjort i Figma. Prosessen av denne har gått fint og vi sitter igjen med en simpel men representativ prototype. Vi skulle også fullføre utviklingen av de systemkravene vi planla i begynnelsen av sprinten. Kontaktpersonen vår ved Capgemini gav et tips om å velge veldig få oppgaver til å begynne med helt i begynnelsen av prosjektet for å få et perspektiv om hvor mye innsats som kreves for hvert systemkrav. Mengden oppgaver vi satt opp for den første sprinten var en behagelig mengde i første omgang og gav et godt bilde på hvor mye arbeidsmengde som krevde for hvert systemkrav.

6.8 Sprint 4

6.8.1 Sprint Planning

I sprint 4 har gruppen satt som mål å ferdigstille både koden og prototypen, samt rette mye tid mot skriving i rapporten. Kodedelen er det gruppen ser på som det viktigste å ferdigstille i denne sprinten ettersom det er svært tidskrevende og ikke noe vi ønsker å stresse med i den avsluttende fasen av prosjektet. Skriving er også viktig for gruppen å ligge godt an på for å slippe store skippertak i slutten av prosjektet.

6.8.2 Daily Standup

Gruppen har fortsatt med parkoding ettersom dette har funket bra frem til nå og vi ikke ser noe poeng i å endre på dette. Der vi har gjort endringer siden sprint 3 er på hvor regelmessig gruppen samles fysisk for å jobbe. I denne sprinten har vi satt mer fokus på å møte fysisk da vi så dette var noe gruppen kunne bli bedre på. Det gjør også at alle gruppemedlemmene har mer kontroll over hva de andre jobber med og det er lettere å hjelpe hverandre hvis det skulle trengs. Det gjør det også mye lettere når vi nå skal bruke mye tid på å skrive og begynne å ferdigstille rapporten.

6.8.3 Sprint Review

Dette sprint review møtet var et konkluderende møte der gruppen sammen med kontaktpersonen i Capgemini, har gått gjennom prosjektet og lagt en plan for siste den siste sprinten. Ettersom dette er et av de siste møtene, ble mye tid brukt på å se på systemet og hvor langt vi har kommet på det. Vi snakket også om potensielle funksjoner som kan legges til i fremtiden ettersom Capgemini vurderer selv om de ønsker å jobbe videre på vårt system. Capgemini har gjennom hele prosjektet vært klar på at dette er en mulighet og at vi derfor må sette opp systemet på en måte som gjør det enkelt for andre å jobbe videre med. Annet enn det reflekterte vi over hvordan prosjektet har gått fra Capgemini sin side og hvordan samarbeidet har gått.

6.8.4 Sprint Retrospective

I denne sprinten har gruppen samlet jobbet veldig bra. Samarbeidet har fungert akkurat som vi ønsket med tanke på hvor ofte vi har fysiske møter. Dette har ført til at alle har hatt kontroll på hva som blir gjort og ikke gjort i prosjektet som igjen har ført til bedre arbeidsstruktur og oversikt for hele gruppen. Når det kommer til koden som har blitt så si ferdigstilt i denne sprinten, er gruppen fornøyd med hvordan systemet er i dag med tanke på at de funksjonene vi ønsket å ha ferdigstilt. Videre har vi også satt opp funksjoner som kan implementeres i systemet for å gjøre det enda bedre. Skrivingen har også gått bra, men her trengs det også mer arbeid. Dette var forventet og blir det som er hovedfokuset når vi nå går inn i den siste sprinten for prosjektet.

6.9 Sprint 5

6.9.1 Sprint Planning

Sprint 5 er den siste sprinten i prosjektet og går derfor bare på å ferdigstille prosjektet. Dette involverer å skrive ferdig rapporten, ferdigstille koden og gjennomgå hele rapporten sammen for å luke ut mulige feil. Det er i denne sprinten alt må leses gjennom flere ganger, ettersom vi ikke kan gjøre noen endringer i rapporten etter dette. I denne sprinten skal vi også ha et siste møte med kontaktpersonen vår i Capgemini og veileder for å gå over og vurdere prosjektet sammen med dem.

6.9.2 Daily Standup

Som nevnt i sprint 4 har gruppen gjort et bevisst valg for å møtes mer fysisk for å jobbe. Dette er noe som bare forsterkes i den siste sprinten der det er planlagt å møtes hver dag for å jobbe.

Dette fører med seg den fordelen at vi har full kontroll på hva som blir jobbet med og hva som trenger mer tid for å fullføres. I denne delen av prosjektet er dette det viktigste som kommer med daglige møter ettersom slutten nærmer seg med stormskritt.

6.9.3 Sprint Review

Dette siste sprint review møtet hadde vi på Capgemini sine kontorer sammen med vår kontaktperson og veilederen vår i IS-304. Sammen med veilederen og kontaktpersonen gikk vi gjennom hele prosjektet og de forskjellige utfordringene som oppsto, samt tilstanden til systemet og hva som eventuelt kan bygges videre på. Kontaktpersonen vår har mest interesse i systemet og det mer tekniske med oppgaven ettersom det er denne delen han har hjulpet oss med og har hovedansvaret for. Applikasjonen har kommet til et punkt der vi ser oss fornøyd med hva som har blitt laget, selv om den ikke er helt ferdig. Dette var forventet ettersom å lage en slik applikasjon krever mer tid enn vi hadde på dette prosjektet.

Når det kommer til rapporten og alt rundt den, ble dette gått mer nøye gått igjennom med veilederen vår ettersom han på dette området har mye kunnskap og erfaring som kan hjelpe oss med å ferdigstille rapporten. Her gikk mye tid til å se at vi har fått med alt som skal være med i rapporten og om noe eventuelt trengte å skrives om.

6.9.4 Sprint Retrospective

Det som fungerte bra i denne sprinten var hvordan gruppen møttes ofte for å jobbe sammen med oppgaven. Dette førte til mer diskusjon om innholdet og med det, en mer utfyllende rapport som reflekterer hvordan hele gruppen tenker. Ikke uventet var det mye ekstra tid utenom det vanlige som måtte legges ned på slutten for å skrive ferdig. Som sagt var dette forventet ettersom det alltid dukker opp ting på slutten av et så stort prosjekt som må gjøres.

7 Refleksjon

Siste delkapittel av rapporten brukes til å reflektere rundt hele utviklingsprosessen, og resultater. Denne oppgaven har gitt oss et læringsutbytte som har vært svært nyttig for fremtidig utdanning. Vi har fulgt bedriften gjennom prosessen om å flytte hele bemanningen inn i nye kontor. Gjennom tre år på Universitetet i Agder har vi hatt fag som berører alle prosessene vi har benyttet i løpet av prosjektet, men for første gang har vi hatt et prosjekt som berører hele systemutviklingsprosessen.

Med en simpel oppgaveforklaring fra Capgemini har vi hentet inn data, utarbeidet systemkrav basert på dataen, tegnet en systemarkitektur med relevante modeller, designet et grensesnitt som er behagelig for brukere, og programmert systemet, samt dokumentere det vi har gjort etter vår beste evne.

Hele prosessen har vært svert lærerik og ikke minst verdifull for oss som studenter. Vi har fått muligheten til å få en ordentlig praktisk oppgave i samarbeid med en anerkjent profesjonell bedrift med svært god kunnskap innenfor feltet. Vi har fått direkte erfaring med nye moderne utviklingsverktøy og ikke minst fått muligheten til å utvide vårt eget nettverk.

Samarbeidet med både skolen og bedriften har også vært gunstig. Kontaktpersonen vår har hatt fire travle måneder hvor han har vært ansvarlig for innflyttingen i det nye bygget, men har vist vilje og innsats i å være tilgjengelige for oss der vi hadde behov for det. Samarbeidet førte i løpet av semesteret til at tre på gruppen hadde jobbintervju, der ett gruppemedlem begynner som systemutvikler hos Capgemini Kristiansand etter endt studietid.

Det å være en gruppe med bare studenter med lite erfaring fra bransjen er å få muligheten til å på egenhånd drive med prosjektstyring. Vi har lært mye om dette temaet og hvordan vi skal drive med prosjektstyring, og vi har fått en god forståelse for hvorfor de forskjellige delene av en systemutviklingsprosess er nødvendige og hvordan de henger sammen.

Læringsutbyttet av prosjektprosessen har vært enorm og gruppen sitter nå med nye kunnskaper om metoder for å utvikle en web-applikasjon, verktøy og rammeverk.

7.1 Utfordringer

Som forventet har vi også møtt på en del utfordringer i løpet av dette prosjektet, både i planleggingen og under utviklingen. Vi ønsker å belyse de forskjellige utfordringene vi møtte underveis og hva vi lærte av dem.

7.1.1 Arbeidsfordeling

Vi hadde i starten av prosjektet litt problemer med å fordele hvem som skulle gjøre hva i prosjektet. Å fordele ansvarsområder var ikke et problem, men å faktisk skulle fordele oppgaver praksis viste seg å være vanskeligere enn tidligere antatt. Vi tilpasset oss ganske raskt situasjonen ved å fordele ut oppgavene under starten av hver sprint slik Scrumban legger til rette for.

7.1.2 Tidsestimering

Vi fikk tidlig beskjed av oppdragsgiver at tidsestimering er noe som er vanskelig selv for erfarne systemutviklere. Det ble da spesielt vanskelig for oss som hadde mindre erfaring med verktøyene og som ikke har vært med på så mange utviklingsprosjekter tidligere. Vi bestemte oss derfor under estimeringen å være veldig åpne for at den faktiske tidsbruken kunne avvike en del fra det vi hadde estimert. Spesielt møtte vi på avvik under utviklingen av selve webapplikasjonen og planleggingen av rapport skrivingen. Uforutsette feil i koden kunne gjerne ta lang tid å fikse. Vi har derfor opparbeidet mye kunnskap om hva som kan skape avvik i tidsestimeringen og hva vi må ta i betraktning neste gang vi skal gjøre det.

7.1.3 Utvikling

Ettersom vi måtte lære oss nye moderne verktøy til å bruke i utviklingen av web-applikasjonen møtte vi på en ganske bratt læringskurve. Vi hadde satt av god tid i starten av semesteret til å lære oss disse verktøyene, men selve læringsprosessen ble ikke gjennomført like effektivt som vi selv hadde ønsket. Vi fikk løst problemene vi møtte på med verktøyene med å ta kontakt med oppdragsgiver og fikk god hjelp til å løse feil i koden som kunne tatt mye tid å løse helt på egenhånd.

7.2 Videre utvikling

Målet med prosjektet vårt var å utvikle grunnmuren for ett booking system som kunne brukes i kontorbyggene til oppdragsgiver. Det har vi fått til ved å lage en web-applikasjon hvor du enkelt kan legge til nye type ressurser. Designmønsteret vi har brukt er også populært og lett

gjenkjennelig. En forbedring vi kunne gjort i utviklingen kunne vært å sørge for at det er lettere å gjenbruke kode i andre deler av systemet, det var en tilbakemeldingene vi fikk fra oppdragsgiver og også en lærdom vi vil ta med oss videre.

8. Konklusjon

Helt fra begynnelsen av prosjektet har gruppen vært klar over at dette er et svært prosjekt. Produktet kom ikke til å bli helt ferdigstilt i løpet av den relativt korte tiden vi hadde til å jobbe med det, og målet vårt var å utvikle en grunnmur. Tidsestimering og arbeidsfordeling er en utfordring i alle slags prosjekt, og dette er ingen unntak. Det er litt av grunnen til at estimeringsmetodikker og prioriteringer er så viktige. Det vi sitter igjen med i slutten av prosjektet er et system som oppfyller alle 'Must Have' systemkrav, samt en del under 'Should Have'. Arbeidet vårt har sørget for at en fortsettelse på utviklingen av dette programmet er mulig, da man bare trenger å ta utgangspunkt i denne rapporten for å forstå hva neste steg i utviklingen er, og designmønsteret er gjenkjennelig. Systemkravene og prioriteringslisten er satt opp med utgangspunkt i hele systemet og ikke bare de systemkravene som er mest viktige og nødvendige for vårt omfang.

Selve oppgaven vi skulle løse var et system for å kunne reservere kontorplasser og møterom på kontoret til en organisasjon. Utfordringen med oppgaven var hovedsakelig at dette systemet skulle kunne brukes av andre bedrifter for å låne ut deres ressurser. Måten vi løste dette var å lage et system som ikke hovedsakelig reserverer kontorplasser eller møterom, men organisasjonene selv skal kunne bestemme hva deres ansatte skal kunne reservere. Ved å tenke mer abstrakt på denne oppgavebeskrivelsen fikk vi et bedre perspektiv på oppgaven, og vi endte med modellen av systemet som kan sees i. Det endelige systemet oppfølger begge disse kravene.

Semesteret har bestått av mange prosesser, metodikker og arbeidsoppgaver som er kjente for gruppen, men har aldri blitt satt sammen i ett, stort prosjekt før. Det viktigste målet for hele gruppen, ved siden av et produkt av kvalitet, har vært et jevnt læringsutbytte blant samtlige gruppemedlemmer. Daglige møter og aktiv diskusjon har ført til at alle på gruppen Radioheads har en felles forståelse av prosjektet.

9. Referanseliste

(n.d.). Retrieved from Figma: https://www.figma.com

Babich, N. (2020, November 24). *Everything You Need to Know About Wireframe Design and Prototypes*. Retrieved from xd.adobe.com: https://xd.adobe.com/ideas/process/wireframing/wireframe-design-definition/

Benyon, D. (2017). *Designing User Experience.* Pearson.

British Broadcasting Corporation. (2022). *Designing solutions - CCEA*. Retrieved from Design Solutions: https://www.bbc.co.uk/bitesize/guides/zcsky4j/revision/7

Brush, K. (2020, April). *MoSCoW method*. Retrieved from TechTarget: https://searchsoftwarequality.techtarget.com/definition/MoSCoW-method#:~:text=The%20MoSCoW%20method%20is%20a,make%20the%20acronym%20more%20pronounceable.

Capgemini. (2022). *INVEST in User Stories*. Retrieved from capgemini.io: https://capgemini.github.io/agile/invest-in-user-stories/

Codeacademy Team. (2022). *MVC, Model, View, Controller*. Retrieved from Codeacademy.org: https://www.codecademy.com/article/mvc

Den norske dataforeningen. (2017, Mars 3). *Sofaprat med Trygve Reenskaug*. Retrieved from dataforeningen.no: https://www.dataforeningen.no/arrangement/mot-en-av-norges-fremste-it-forskere-sofaprat-med-trygve-reenskaug/

Digite. (2022). *What Is Scrum Methodology & Scrum Project Management.* Retrieved from Digite: https://www.digite.com/agile/scrum-methodology/

George, T. (2022, January 27). *Semi-Structured Interview | Definition, Guide & Examples*. Retrieved from Scribbr: https://www.scribbr.com/methodology/semi-structured-interview/

Gillis, A. S. (2021, Juni). *pair programming*. Retrieved from Techtarget: https://www.techtarget.com/searchsoftwarequality/definition/Pair-programming#:~:text=Pair%20programming%20is%20an%20Agile,code%20and%20test%20user%20stories.

Indika. (2011). *Difference between ER diagram and Class Diagram.* Retrieved from DifferenceBetween: https://www.differencebetween.com/difference-between-er-diagram-and-vs-class-diagram/

Lucidchart. (2022). *What is an Entity Relationship Diagram (ERD)?* Retrieved from Lucidchart.com: https://www.lucidchart.com/pages/er-diagrams

MDN. (2022). *MVC.* Retrieved from MDN: https://developer.mozilla.org/en-US/docs/Glossary/MVC

Microsoft . (2022). *Visual Studio.* Retrieved from Microsoft: https://visualstudio.microsoft.com/

Microsoft. (2022). *ASP.net.* Retrieved from Microsoft.com: https://dotnet.microsoft.com/en-us/apps/aspnet

Microsoft. (2022). *Basic Web Application*. Retrieved from Microsoft Learn: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/basic-web-app?tabs=cli

Microsoft. (2022). *Entity Framework documentation*. Retrieved from microsoft: https://docs.microsoft.com/en-us/ef/

Microsoft. (2022). *Security Update Guide FAQs*. Retrieved from Microsoft.com: https://www.microsoft.com/en-us/msrc/faqs-security-update-guide

MountainGoat Software. (2022). *User Stories*. Retrieved from MountaingGoatSoftware.com: https://www.mountaingoatsoftware.com/agile/user-stories

Nishadha. (2021). *Er-Diagrams-Tutorial*. Retrieved from Creately: https://creately.com/blog/diagrams/er-diagrams-tutorial/

Planning Poker. (2022). *Planning Poker*. Retrieved from PÅlanning Poker: https://www.planningpoker.com/

Quadrum. (2022). *Forside*. Retrieved from Quadrum.no: https://quadrum.no/

Radigan, D. (2022). *What is Canban*. Retrieved from Atlassin Aglie Coach: https://www.atlassian.com/agile/kanban

Scrum.org. (2022). *What is Scrum?* Retrieved from Scrum.org: https://www.scrum.org/resources/what-is-scrum

Studio by UXPin. (n.d.). *What Is a Prototype: A Guide to Functional UX*. Retrieved from uxpin.com: https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/#:~:text=A%20prototype%20is%20%E2%80%9CA%20simulation,to%20engineering%20teams%20for%20development.

Teamwork.com. (2022). *Teamwork.com.* Retrieved from Project- managment-methologies: https://www.teamwork.com/project-management-guide/project-management-methodologies/

tubik. (2017, 06 23). *Color Theory: Brief Guide For Designers.* Retrieved from uxplanet: https://uxplanet.org/color-theory-brief-guide-for-designers-76e11c57eaa

Wikipedia. (2022, 05 4). *Pair programming*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Pair_programming

Williams, E. (2021, 03 26). *Visual Studio VS. Visual Studio Code*. Retrieved from codeguru: https://www.codeguru.com/visual-studio/visual-studio-vs-visual-studio-code/

Vedlegg

Uttalelse fra Capgemini Kristiansand

Uttalelse fra Capgemini

Capgemini er et konsulentselskap med over 300 000 medarbeidere i over 50 land. I Norge er vi cirka 1500 ansatte og etablerte vårt kontor i Kristiansand høsten 2021. I forbindelse med inngåelse av kontrakt for leie av lokaler i Quadrum ble vi også gjort oppmerksom på hvordan huseiere møter nye behov i forbindelse med utleie av moderne lokaler i en post-Covid sammenheng.

Trenden er at bedrifter som leier kontorlokaler ønsker større fleksibilitet i forhold til kontorarealer og fellesfunksjoner i kontorbyggene. Slike fellesfunksjoner kan bl.a. inkludere løsninger for leie av bil, felles møterom, felles kontorarealer mv. Ut fra dette behovet oppsto en forespørsel om et bestillingssystem for fellesfunksjoner på kryss av leietaker i et bygg.

Capgemini tok med seg forespørselen til «RefreshIT». «RefreshIT» er et arrangement i regi av Universitetet i Agder, og en mulighet for bedrifter å presentere seg for bachelorstudenter. Her inviteres bachelorstudenter inn til prosjekter hvor de kan arbeide sammen med bedrifter i forbindelse med bacheloroppgaven. Capgemini møtte her mange interessante studentgrupper, og hadde også intervjuer med flere av disse i etterkant.

Valget vårt falt på gruppen bestående av Andreas Hennestad, Jaran Faret, Stig Hagum og Martin Løvberg. Gruppen ble valgt fordi de fremsto faglig dyktige, strukturerte og viste en entusiasme for både Capgemini og prosjektet. Det var også en gruppe som hadde arbeidet mye sammen før, og hadde en god fordeling ift. interesser og kompetanseområder. Det var også en klar rollefordeling i gruppen.

Gjennom prosjektet har Capgemini bistått med både tekniske og strategiske ressurser, og vi har i tillegg kommet med innspill fra huseier (Quadrum AS). Studentgruppen har også hatt intervjuer med både Capgemini-ansatte og ansatte hos huseier. Videre har Capgemini støttet prosjektgruppen i forbindelse med arkitekturvalg, utviklerverktøy og metodikk, men gruppen har utvist stor grad av selvstendighet og evne til å ta disse valgene selv. Gruppen har sånn sett bekreftet de initielle inntrykkene fra intervjuene og har gjennom hele prosjektet opptrådt svært profesjonelle og entusiastiske. De har også vist en stor grad av læringsvillighet i forhold til nye metoder og verktøy som har blitt introdusert for dem gjennom prosjektet.

Prosjektet har vært gjennomført etter smidig metodikk («Scrumban») hvor Azure DevOps har blitt brukt for både kildekodekontroll og prosjektstyring. Gruppen har hatt tydelige roller og gjennomført prosjektet glimrende fra vårt ståsted. Vi i Capgemini har hatt regelmessig kontakt med prosjektgruppen og fulgt fremdriften gjennom sprint reviews og andre demoer. Gruppen har utvist stort engasjement og forpliktelse i prosjektet, samtidig som de har vært bevisst på tidsbruk og levert oppgaver i henhold til fastsatte estimater. Gruppen har jobbet delvis i Capgemini sine lokaler, og delvis off-site. På den måten har de også demonstrert sine evner til samarbeid både internt i gruppen, og med ansatte hos Capgemini.

Capgemini setter veldig stor pris på det arbeidet studentgruppen har lagt ned i prosjektet, og vi har også fått positive tilbakemeldinger fra huseier på de intervjuer som har blitt utført. Vi er veldig stolte av å være en del av dette bachelorprosjektet og ønsker å utvise takknemlighet overfor gruppen for godt samarbeid.

Kristiansand, 13. mai 2022

Med vennlig hilsen

Øyvind Mjølund

Managing Consultant

Egenvurdering

Radioheads har arbeidet sammen i grupper gjennom 3 år med utdanning, og kjenner hverandre godt. Det har vært et hovedfokus gjennom årene at gruppen skal fungere som en enhet, og ikke fire personer som samarbeider mot et felles mål. Vi har alltid tatt hensyn for å sørge for at alle medlemmer får et jevnt læringsutbytte, og dette prosjektet har ikke vært annerledes. Hvert gruppemedlem har hatt fokusområder hvor vedkommende har arbeidet mest på, men gruppen har, så godt det har latt seg gjøre, sørget for at alle medlemmer er med på alle områder at prosessen slik at læringsutbyttet blir så jevnt som mulig.

Jaran Faret

Min rolle gjennom prosjektet har vært Scrum Master, og hatt hovedansvaret for å strukturere rapporten. Rollen som Scrum master innebærer å kalle inn til møter, holde oversikt over resten av gruppa sitt arbeid, og fordele videre oppgaver. Jeg har vært scrum master i tidligere prosjekt, men ikke på et så stort prosjekt som dette. Det har vært krevende å holde styr på alt arbeidet som har blitt gjort, men ved hjelp av systemkrav, estimeringsliste og prioriteringsliste, har jeg levert

et arbeid både jeg og resten av gruppa er tilfredsstilt med. I tillegg har jeg stått ansvarlig for strukturen og mye av skrivingen på rapporten, og tatt avgjørelser om hvordan kapitlene og delkapitlene i rapporten er satt opp, og hva de handler om.
Jeg har også bidratt aktivt med datainnsamling, utvikling av design, og programmeringen av produktet.

Stig Hagum

Min rolle i prosjektet har hatt fokus på utvikling og rapport. Før vi kunne begynne på utviklingen var jeg med på å hente inn nødvendig informasjon for å lage systemkrav. Som utviklingsansvarlige har jeg vært med på å bestemme hvilke endringer som har vært nødvendige for at systemet skal bli ferdig og funksjonelt. Dette har involvert noen prioriteringsendringer som krevede at vi fokuserte mer på kjernefunksjoner og droppet mindre viktige funksjoner. Jeg har også vært med ellers der det har vært behov.

Andreas Hennestad

Jeg har i dette prosjektet hatt som en av to hatt ansvaret for utviklingen av selve webapplikasjonen og hatt ansvar versjonskontroll av koden i Azure Devops. Det å være to personer med ansvar for utviklingen har vært svært positivt for både diskusjon og refleksjon rundt valg vi har tatt i koden. Jeg har i løpet av denne bachelor oppgaven fått bekreftet at utvikling er noe jeg liker å jobbe med og ønsker å fortsette med i arbeidslivet. Jeg vil gjerne takke de andre medlemmene på gruppen og er stolt over det vi som et team har klart å oppnå sammen.

Martin Løvberg

Min rolle i prosjektet har hovedsakelig gått på designdelen og prototypen, samt rapporten. Ved å ha hovedansvaret for de forskjellige wireframene og protoypen har jeg med dette også hatt hovedansvaret for hvilken retning designet av systemet skal ta, med tanke på hvilke prinsipper som skal følges og hva som skal settes fokus på. Jeg har også sammen med de andre gruppemedlemmene jobbet jevnt med rapporten gjennom hele prosjektet og ellers bistått der det trengs, eventuelt der jeg har viktig kunnskap eller meninger som har vært med i betraktningen.

Gruppekontrakt

Vi, gruppe nr. 13 som består av følgende medlemmer: Martin Nyenget Løvberg, Stig Hagum, Jaran Faret og Andreas Hennestad

Skal jobbe sammen om følgende prosjekt: Bachelor prosjekt vår 2022

Bakgrunn og motivasjon

Oppgaven vi valgte er noe som virket interessant for alle gruppemedlemmene og samtidig passet oss og våre kunnskaper.

Bestemmelser om fremmøte og samarbeid

Jeg forplikter meg overfor gruppen min, til at jeg vil:

- Følge forelesninger, og delta i gruppearbeid, samlinger og øvelser, be om/ta imot veiledning, gjøre selvstudium av litteratur og gjøre designforskning; delta aktivt i gruppearbeid med feltstudier.
- Respektere kravet om minst 80% fremmøte og også oppgi grunn hvis jeg er forhindret.
- Overholde frister og levere oppgaver som publiseres på Canvas, innen fristene.
- Sørge for at arbeid fordeles jevnt mellom gruppemedlemmer, men samtidig utnytte hver enkelt students spesielle ferdigheter og bakgrunn.
- Være tilgjengelig på melding for å kunne svare på spørsmål og oppklare uklarheten innad i gruppa.

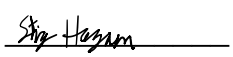
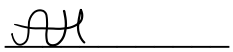
Andre bestemmelser

Gruppeleder leder gruppemøtene og sørger for at alle er med i arbeidet. Er det noe som må diskuteres, skal gruppeleder styre diskusjonen, og sørger for at alle blir hørt. Etter en diskusjon av en lengde som passer til temaets viktighet, avveid mot behovet for å ta en beslutning og komme videre i prosjektet, tas det en omforent beslutning. Er det uenighet, foretas det en avstemming, der gruppeleder har dobbeltstemme ved stemmelikhet. Etter at beslutninger er fattet skal alle forholde seg lojalt til disse.

Gruppeleder er gruppens kontaktperson utad, og hovedansvarlig for fremdrift og for at frister for innlevering m v overholdes. Gruppeleder er også hovedansvarlig for å holde kontakt med og arrangere møter med eksterne partnere, og med intervjuobjekter/samtalepartnere.

Gruppeleder eller viseleder fører logg/kort møtereferat, om hva vi ble enige om, hva gjorde vi og hva gjør vi neste gang. Av referatet skal fremgå: fremmøte, forfall (meldt på forhånd, med grunn), fravær (ikke møtt/uten oppgitt grunn).
Gruppeleder eller viseleder har ansvar for å innkalle til møter.

Kristiansand, 04.01.2022

Jaran Faret	jaran@faret.com	+47 988 42 255	
Stig Hagum	stig.hagum@lyse.net	+47 971 01 505	
Andreas Hennestad	a.hennestad@gmail.com	+47 984 64 411	
Martin Løvberg	martin.lovberg@gmail.com	+47 413 07 122	

Transskribert Intervju

Huseier og ansvarlig for innflytting i Quadrum har sagt ja til å svare på noen spørsmål i et semi-strukturert intervju. Dette ville gi oss et annet perspektiv til programmet vi lager. Før intervjuet fikk vi tillatelse til å ta opp samtalen, slik at transskriberingen ble lettere i etterkant. Etter fullført transkribering har opptaket blitt slettet. Under ser du informasjonen vi fikk fra intervjuet.

Hvilke funksjoner ser du for deg som absolutt nødvendige i dette systemet?

-En mulighet for å kunne reservere et møterom i første etasje, med tilhørende informasjon om utstyr og sitteplasser. Reservasjonen skal kunne inneholde informasjon om hvem som booker, hvilket firma vedkommende tilhører og hvor lenge rommet er reservert.

Hvilke flere funksjoner kunne du tenke deg med i et slikt system?

-Det burde være en superbruker, som kan fikse og rette opp i feil som oppstår. Folk på møterommene skal kunne bestille mat og kaffe/vann dersom dette er ønskelig.

Har du noe formening om hvordan systemet skal brukes, mobil applikasjon eller webapplikasjon?

-Først og fremst en nettside, kan eventuelt lages en applikasjon i senere tid. Det er bestilt inn

små skjermer som skal henges opp utenfor møterommene og disse skal kunne vise om rommet er ledig eller ikke.

Har du erfaring med lignende programmer? Hvis ja, hvilke program og hva synes du er bra med de?

-Det er bra med lignende programmer som har en bra oversikt over de tilgjengelige møterommene slik at brukere ikke er i noe tvil om hva uvedkommende gjør.

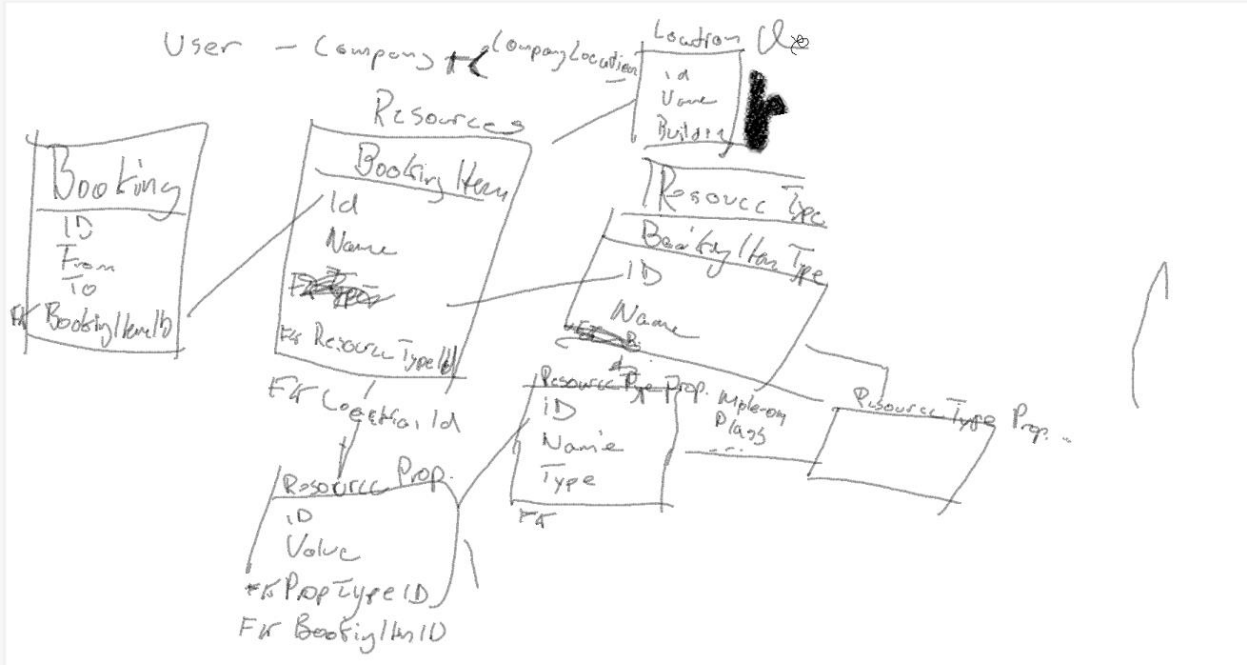
Er det aktuelt at et program som dette er koblet til kantinen og gir de bedre estimat for hvor mye mat som trenger å bli forberedt?

-Dersom noe slikt informasjon blir lagret hadde muligheten for å sende en mail med slik informasjon til kantina, vært veldig aktuelt for effektiviteten i kantina og bærekraft generelt.

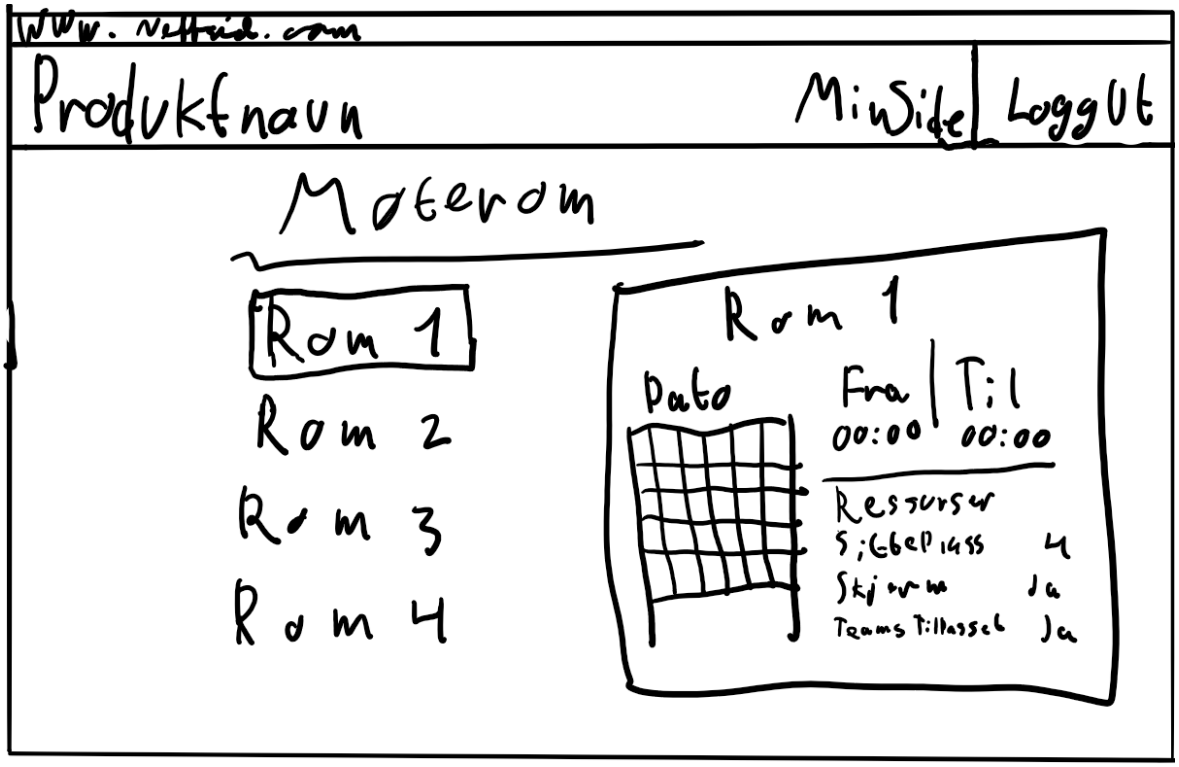
Er det noe mer du ønsker å si som du føler ikke har blitt nevnt allerede?

-En bruker kan kunne ha muligheten til å varsle til en admin dersom noe ikke fungerer som det skal, slik at admin kan fikse dette i rimelig tid.

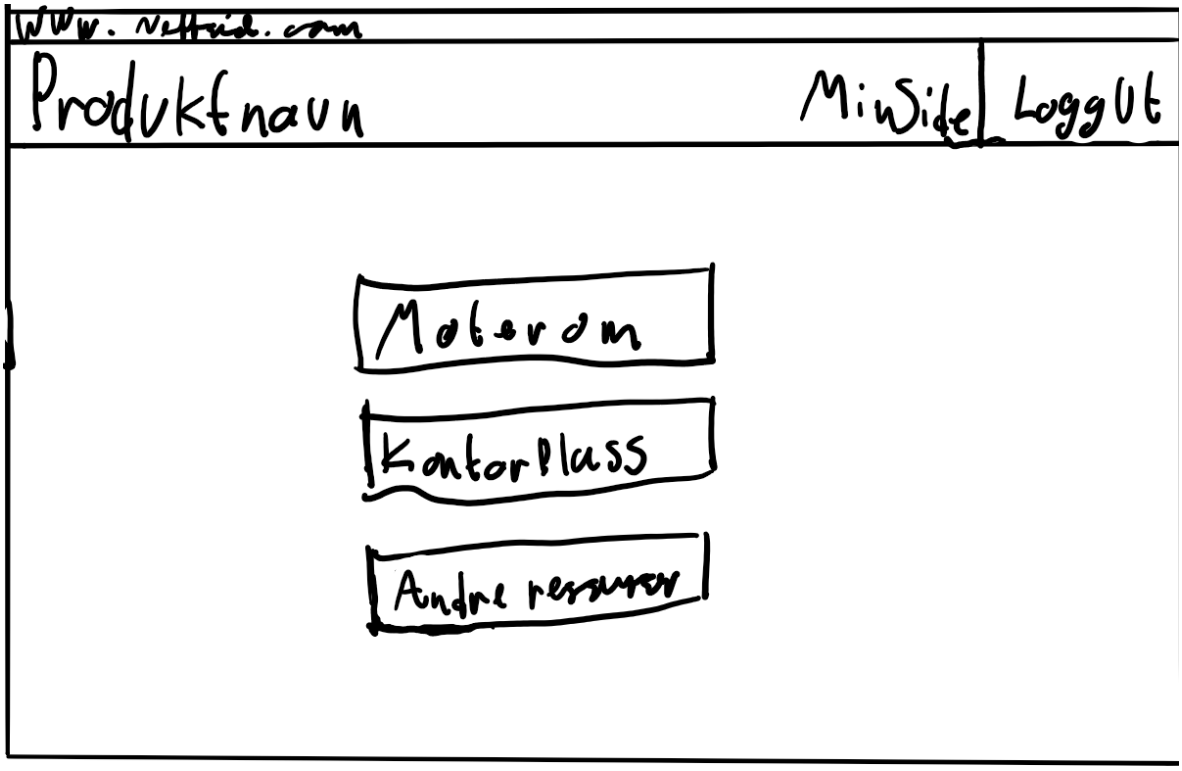
Modeller



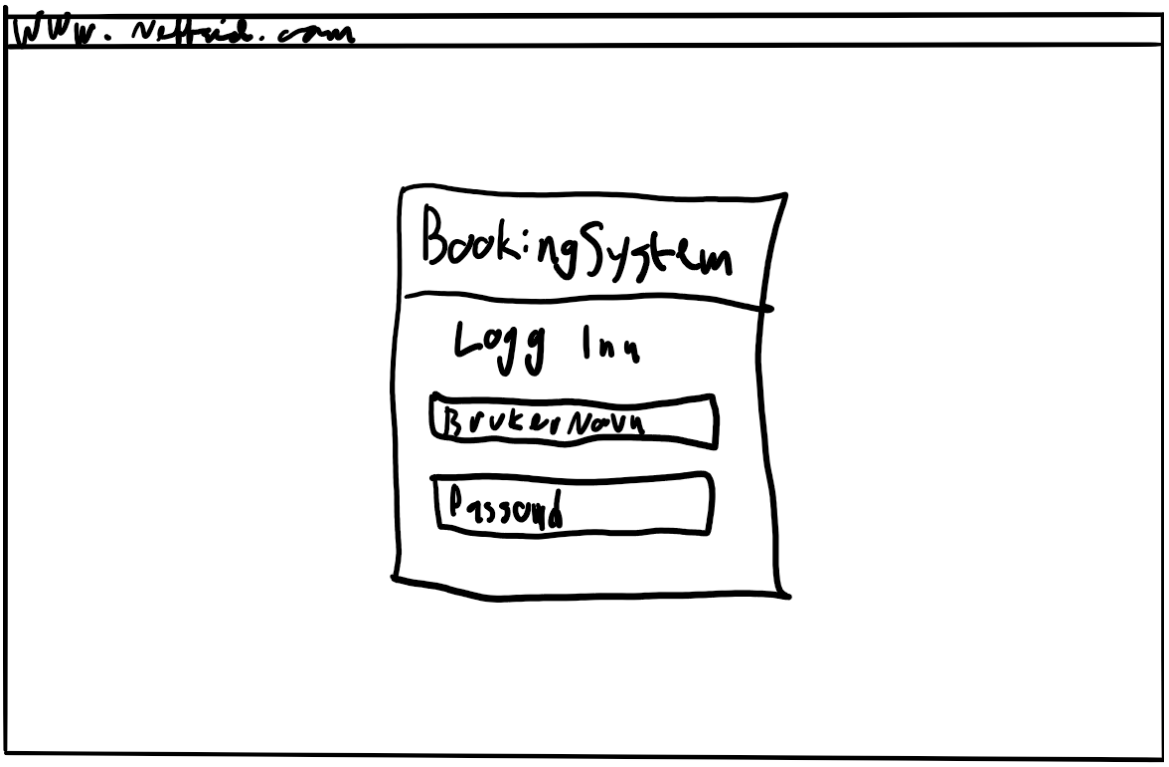
Modell 1: Tegnet kladd av database



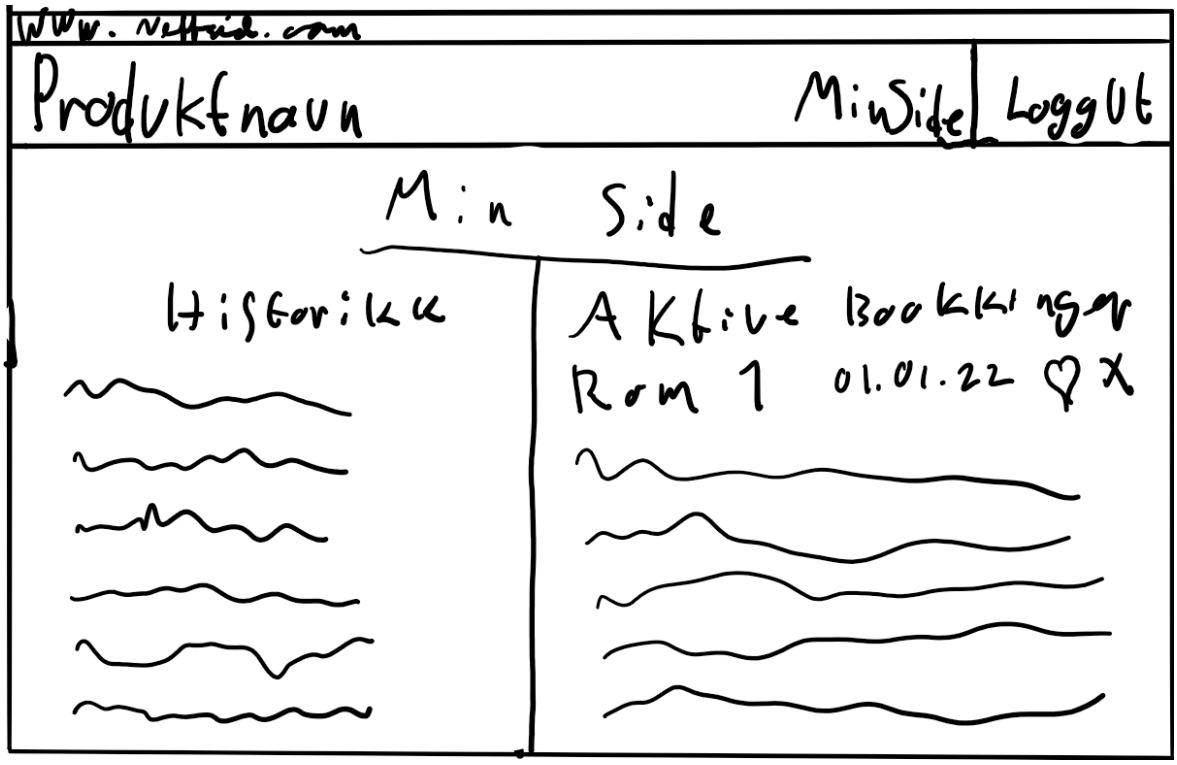
Modell 2: Lo-Fi Wireframe 1



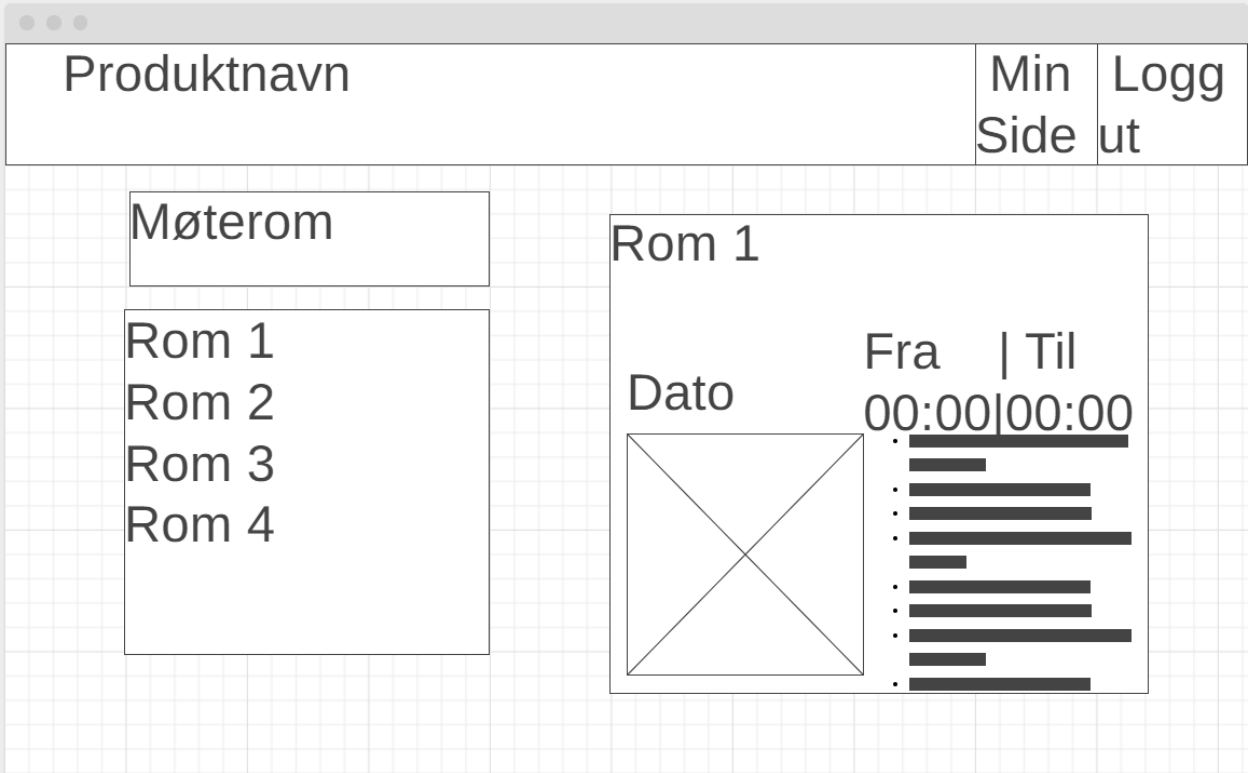
Modell 3: Lo-Fi wireframe 2



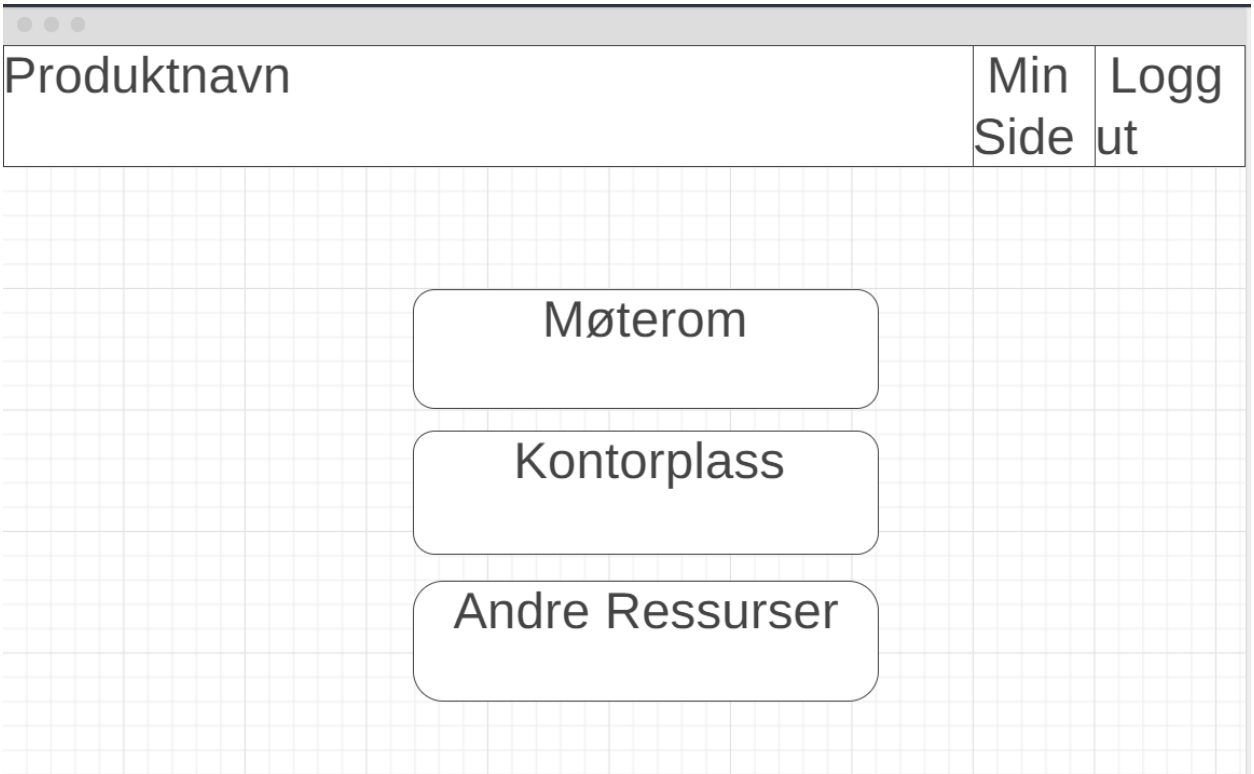
Modell 4: Lo-Fi wireframe 3



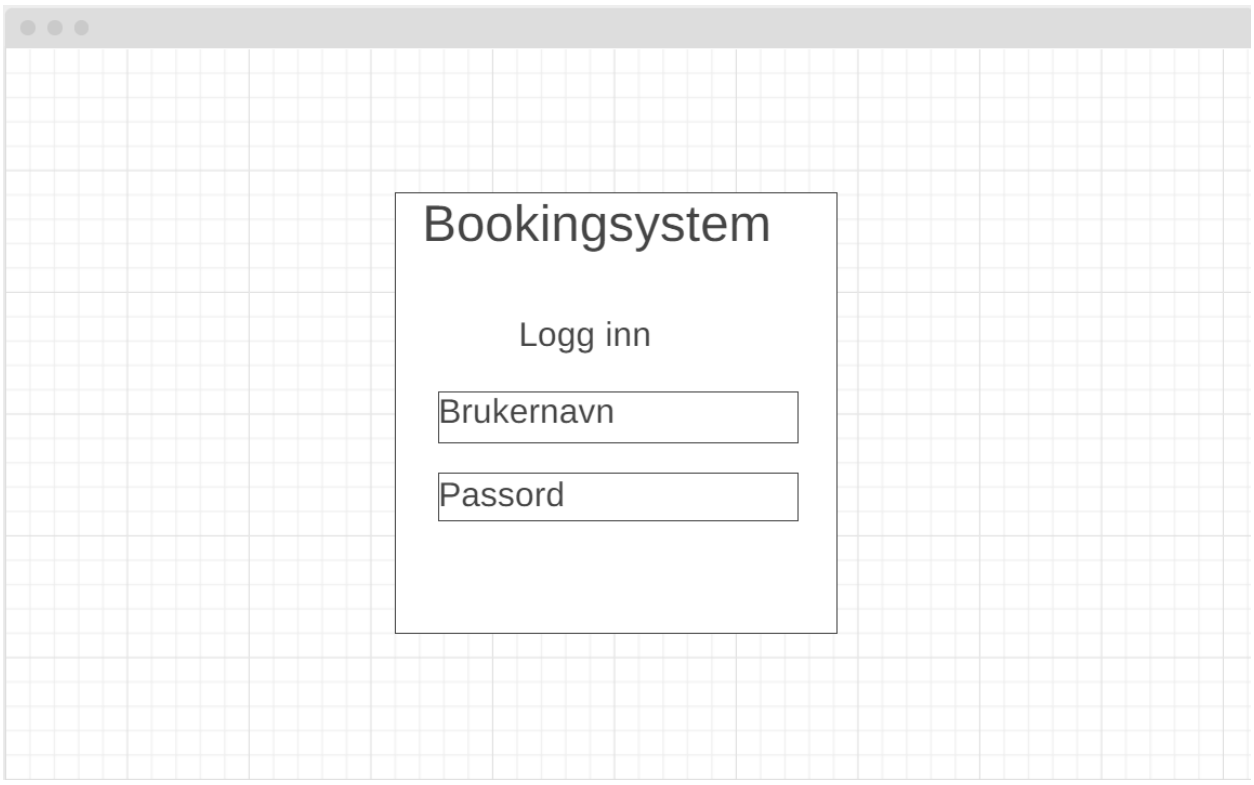
Modell 5: Lo-Fi Wireframe 4



Modell 6: Lo-Fi wireframe 5



Modell 7: Lo-Fi wireframe 6



Modell 8: Lo-Fi wireframe 7

Modell 9: Lo-Fi wireframe 8