UNIVERSITETET I AGDER

Forside

IS-304: 2023

Tittel: Utvikling av web-applikasjon for bestilling av dagligvarer

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Janis Gailis
Oppdragsgiver:	Spar Åsane

Studenter:

Etternavn	Fornavn
Nordbø	Marius
Shwane	Dilan
Vikan	Knut Erik

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI __
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI __
Vi bekrefter at alle i gruppen har bidratt til besvarelsen	JA X	NEI __

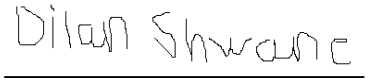
Forord

Denne rapporten er skrevet av vi i gruppe 3 og representerer den siste delen av bachelorstudiet IT og informasjonssystemer på Universitet i Agder. Vi vil gjerne benytte denne anledningen til å si takk til noen personer som har spilt viktige roller gjennom arbeidet med bachelorprosjektet.

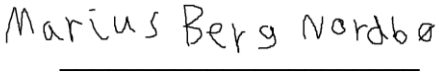
Vi vil gjerne gi en stor takk til Janis Gailis som har vært veileder for gruppen. Janis har alltid vært der for oss for å svare på spørsmål og gi oss tilbakemeldinger på arbeidet vi har gjennomført hele semesteret. Videre vil vi takke Hallgeir Nilsen som er fagansvarlig. Hallgeir har satt opp presentasjoner mellom gruppene som har bidratt til læring på tvers av gruppene. Han har også holdt nyttige forelesninger.

Til slutt vil vi gjerne takke Spar Åsane for at vi fikk samarbeide med dem i et spennende prosjekt og mer spesifikt daglig leder Arild Nordbø. Arild har vært tilgjengelig for både møter, spørsmål og tilbakemeldinger gjennom hele semesteret.

Denne rapporten er skrevet av gruppe 3 der alle i gruppen samtykker at de har bidratt til utviklingen av prosjektet.



Dilan Shwane



Marius Berg Nordbø

Knut Erik

Sammendrag

Denne bacheloroppgaven har blitt skrevet av gruppe 3 i samarbeid med Spar Åsane. Dette samarbeidet begynte i November 2022 etter at gruppen tok kontakt med dem. Gruppen begynte deretter på prosjektet allerede i januar 2023. Målet med prosjektet var å lage en web-applikasjon der Spar Åsane sine private kunder kunne lage bestillinger som skulle overta den prosessen de allerede brukte som var å bruke email.

Gruppen brukte Scrum som prosjektstyringsmetodikk gjennom utviklingen av applikasjonen. Gruppen valgte Scrum fordi vi hadde hatt god erfaring med det tidligere. En annen grunn til at det ble valgt var fordi Scrum gjør det lettere å jobbe agilt som tillater gruppen å kunne gå tilbake og fikse opp i ting. Dette ble gjort ved oppnå mål i korte perioder om gangen og kontinuerlig kommunikasjon med produkteier.

Gruppen har lært utrolig mye gjennom semesteret og fått lært hvordan det er å jobbe med en kunde i den virkelige verden. Gruppen fikk muligheten til å bruke mye av kunnskapen vi har fått gjennom studiet i dette prosjektet. Gjennom prosjektet ble det gjort en del programmering som ga gruppen verdifull erfaring som vi kan ta med videre i karrierene våre.

Prosjektet krevde både Frontend og Backend utvikling der gruppen startet med en beskrivelse av systemet og etter å ha vært gjennom flere prosesser, endte vi opp med en webapplikasjon som oppfyller de viktigste kravene til Spar Åsane.

Denne linken fører til en video av det ferdige resultatet:
https://www.youtube.com/watch?v=YryXoVmvyJk

Innholdsfortegnelse

Innhold

Figur liste

Tabell liste

1 - Introduksjon

I november 2022 kom gruppen i kontakt med Spar Åsane der de presenterte gruppen for et prosjekt. Spar Åsane er en lokal dagligvare-bedrift i Kristiansand og er en del av Spar kjeden. Prosjektet virket veldig spennende for gruppen og vi fikk et veldig godt inntrykk av Spar Åsane. På bakgrunn av dette kom gruppen til å takke ja til prosjektet og å ha det som bachelorprosjekt.

Prosjektet Spar Åsane presenterte var et system der deres private kunder kunne, gjennom en katalog, legge til produkter i en handlekurv og sende som ordre. Systemet skal sørge for at kundene ikke ender opp med noen ting dersom lageret er tomt for produktet. Gjennom systemet skal også ansatte på Spar kunne finne frem til nye ordre og sende en melding til kunden når produktene er pakket og sendt. Meldingen utlyser alle produkter som er sendt og alle produkter som manglet. Gruppen så på dette som en spennende utfordring og vi mente at vi kunne få dette til.

Gruppe 3 består av tre medlemmer der to av dem har jobbet med hverandre tidligere. Gruppen har veldig god kjemi sammen og det var lett for oss å jobbe sammen. Gruppemedlemmene har sjeldent hatt de samme valgfagene gjennom studiet og alle har forskjellige områder vi er gode på. Dette ser gruppen på som et stort pluss og det tillater oss å spille på hverandres styrker. Tidligere har alle på gruppen hatt god erfaring med Scrum og derfor ønsket vi å fortsette å bruke det som prosjektstyringsmetodikk. Gruppen hadde ikke mulighet til å jobbe hos Spar Åsane ettersom at det er en dagligvarebutikk, så derfor ble gruppen nødt til å benytte seg av grupperom på UiA.

Gjennom denne rapporten vil det bli redegjort for beslutninger som gruppen har tatt gjennom arbeidet med prosjektet. Første delen av rapporter tar for seg analyse og designprosessen av prosjektet. Etter det vil det bli avklart språk og verktøy som ble tatt i bruk under utviklingen av prosjektet og viktige beslutninger som ble tatt for utviklingsdelen. Deretter kommer det en del som handler om Scrum som prosjektstyringsmetodikk og hvordan gruppen gjennomførte det. Til slutt kommer det både en refleksjon og en konklusjon om hvordan arbeidet i prosjektet har gått.

2 - Analyse

I dette kapittelet presenteres analysedelen av prosjektet som ble utført slik at gruppen kunne lære mer om hvordan webapplikasjonen skulle fungere. Det vil bli redegjort for arbeidet som gruppen gjorde for å få tilgang til nødvendig informasjon. I analysedelen ble det utarbeidet systemkrav, brukerhistorier, rikt bilde, entity relationship diagram, og risikoanalyse.

2.1 - Systemkrav

Basert på samtaler med Spar Åsane, opprettet vi sammen krav til systemet som definerte hvilke funksjoner og konsepter som var viktigst å inkludere. Da fikk vi også definert det nøyaktige målet for prosjektet. Systemet skal bytte ut den originale manuelle metoden Spar Åsane bruker for å motta bestillinger fra spesifikke bedriftskunder, og samtidig gjøre det enklere for kundene å få nøyaktig riktig vare de ønsker. Kravene som er prioritert som Must Have må til for at systemet skal være nøyaktig som den manuelle prosessen. Målet med prosjektet er å forbedre dette, derfor utarbeidet vi flere krav under lavere prioriteringer som definerer systemet som en oppgradering til den manuelle prosessen. Tabell 1 viser noen av systemkravene vi kom frem til. Resten kan bli funnet i vedlegg 3.

Tabell 1 - Systemkrav

Systemkrav	**Prioritet**
Systemet må ha en katalog for kundene som gjør det mulig å søke etter og filtrere for varer.	Must Have
Systemet må lagre kundenes varer i en handlekurv, og ha mulighet for å sende kundenes ordre til Spar Åsane, slik at varene kan deretter pakkes.	Must Have
Systemet må ha mulighet for kommunikasjon mellom kunde og bedrift. Spar Åsane vil godkjenne ordre og informere kunden om feil.	Must Have
Systemet må ha liste over alle innkommende ordrer, og ha mulighet for at disse ordrene kan skrives ut.	Must Have
Systemet bør gi kunden en mulighet for å inkludere en alternativ vare sammen med varen de opprinnelig ville ha.	Should Have
Systemet bør la kunden sende med en melding sammen med hver vare der de kan informere Spar Åsane om spesifikke hensyn som må tas med bestillingen.	Should Have
Systemet bør ha statistikk/informasjon om kundene som bedriften kan bruke.	Should Have

Systemet bør være brukervennlig for både kunde og bedrift, slik at det er enklere å bruke systemet enn å ikke gjøre det.	Should Have

2.2 - Brukerhistorier

Etter at systemkravene var klare, satt gruppen seg ned for å analysere og tolke dem og basert på disse kravene, begynte gruppen å komme opp med brukerhistorier. Brukerhistorier forklarer funksjoner i et system ut fra et perspektiv fra en som kommer til å bruke systemet. Brukerhistorier går hånd i hånd med agilt arbeid, fordi en viktig del av agilt arbeid er å sette mennesker i fokus som er nøyaktig det brukerhistorier gjør med brukerhistorier (Rehkopf, u.å).

Tabell 2 inneholder noen av brukerhistoriene som gruppen kom frem til som ble brukt som utgangspunkt videre i prosjektet. Resten kan bli funnet i vedlegg 4. Gruppen skrev brukerhistoriene basert på systemkravene. Hver brukerhistorie starter med en bruker av systemet, som vil noe, slik at de oppnår noe (Rehkopf, u.å). Videre har vi skrevet kriterier for hver av brukerhistoriene hvor vi da skriver hva slags funksjonalitet systemet må ha for å kunne tilfredsstille brukerhistoriene.

Deretter ble MoSCoW teknikken tatt i bruk for å kunne sette prioritert på hver brukerhistorie. MoSCoW er en teknikk for å sette prioriteringer som gruppen har hatt god erfaring med tidligere i studiet. Teknikken bruker 4 kategorier for prioriteringer: “Must Have” som betyr at systemet er avhengig av noe for å i det hele tatt funke, “Should Have” som betyr at systemet bør ha det dersom det er nok tid for det, men systemet vil klare seg uten, “Could Have” som betyr at det ikke er like viktig så det går bra og ikke inkludere det, og til slutt “Want to have but won’t have this time around” som betyr at det ikke er viktig og heller vente med det til senere (Benyon, 2014, s.140). For hver brukerhistorie ble det argumentert av gruppen for hvorfor brukerhistorien fikk den prioriteringen som den fikk.

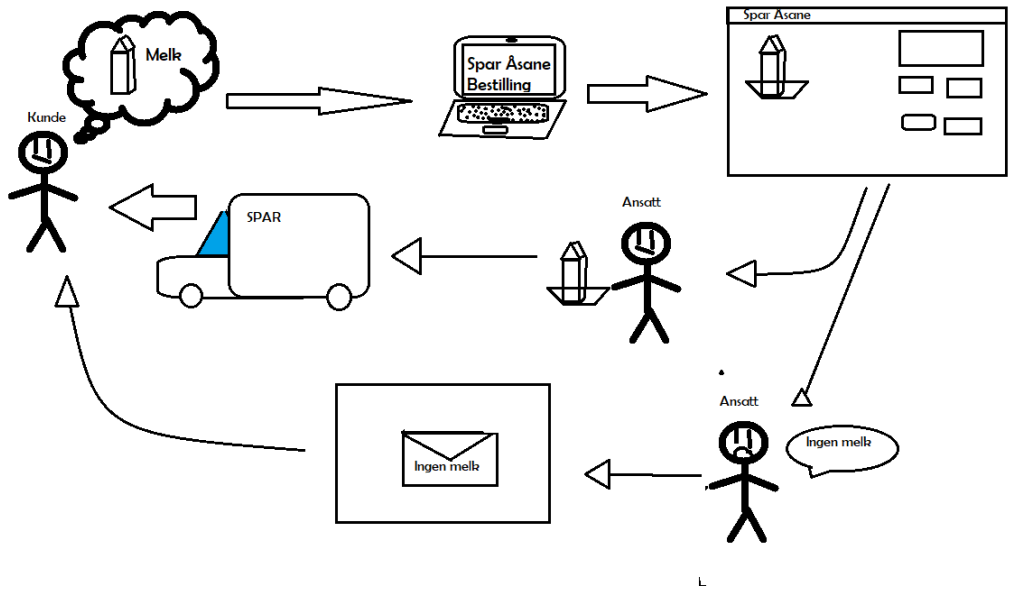
Tabell 2 - Brukerhistorier

ID	Brukerhistorie	Kriterier	Argument	Prioritet
1	Som privatkunde hos Spar Åsane, ønsker jeg å enkelt kunne finne fram til og bestille varer ved behov til pensjonister.	Applikasjonen må inneholde en katalog som gjør det mulig å kunne filtrere gjennom varer og så legge dem til en handlekurv, for så å kunne sende det inn som bestilling.	Denne funksjonen er "Must Have" som prioritet fordi denne funksjonen er hele poenget med web applikasjonen og gir grunnlaget for resten av systemet.	Must Have
2	Som bedriftskunde hos Spar Åsane, ønsker jeg å finne spesifikke varer og bestille et stort antall varer. Jeg vil også vite hvilke varer som ikke er tilgjengelige, så jeg vet nøyaktig hvilke varer som blir sendt.	Applikasjonen må inneholde en katalog som gjør det mulig å kunne filtrere gjennom varer og så legge dem til en handlekurv, for så å kunne sende det inn som bestilling. Dersom varen ikke er tilgjengelig, så får kunden sendt et varsel om det.	Denne funksjonen er "Must Have" som prioritet fordi denne funksjonen er hele poenget med web applikasjonen og gir grunnlaget for resten av systemet.	Must Have
3	Som ansatt hos Spar Åsane ønsker jeg å godkjenne bestillinger, og informere kunder om feil med bestillinger, så de blir informerte.	Applikasjonen må inneholde en funksjon som gjør det mulig for ansatte å godkjenne bestillinger og sende varsel til kunden om feil med bestillingen.	Denne funksjonen er "Must Have" fordi det er en av grunnene til at web applikasjonen i det hele tatt blir laget. Fordi det vil gjøre det lettere for kunder å få det de vil ha	Must Have
4	Som administrator hos Spar Åsane ønsker jeg å ha oversikt over alle åpne bestillinger, så jeg kan se på ordre detaljer.	Applikasjonen må inneholde en funksjon for å kunne samle og se alle de forskjellige bestillingene fra bedrifter og privatpersoner	Denne funksjonen er "Must Have" fordi vi skal effektivisere prosessen, ikke gjøre det vanskeligere å holde oversikt	Must Have

2.3 - Rikt Bilde

Etter at vi hadde definert flere brukerhistorier, lagde vi et rikt bilde basert på en av dem. Et rikt bilde er en illustrasjon av hvordan man forstår en situasjon. Det blir laget i form av en enkel og informell tegning og tar for seg viktige sider av en situasjon. I et rikt bilde ser man hvordan enheter som for eksempel mennesker, fysiske objekter, oppgaver og steder interagerer med hverandre. Når man lager rike bilder så er det veldig vanlig at mennesker blir fokusert på (Mathiassen, 2018, s.26-28).

For vårt rikt bilde, valgte vi å illustrere brukerhistorien “Som bedriftskunde hos Spar Åsane, ønsker jeg å finne spesifikke varer og bestille et stort antall varer. Jeg vil også vite hvilke varer som ikke er tilgjengelige, så jeg vet nøyaktig hvilke varer som blir sendt.” Vi lagde det først sammen på en tavle, og deretter laget en egen versjon digitalt.



Figur 1 - Rikt bilde

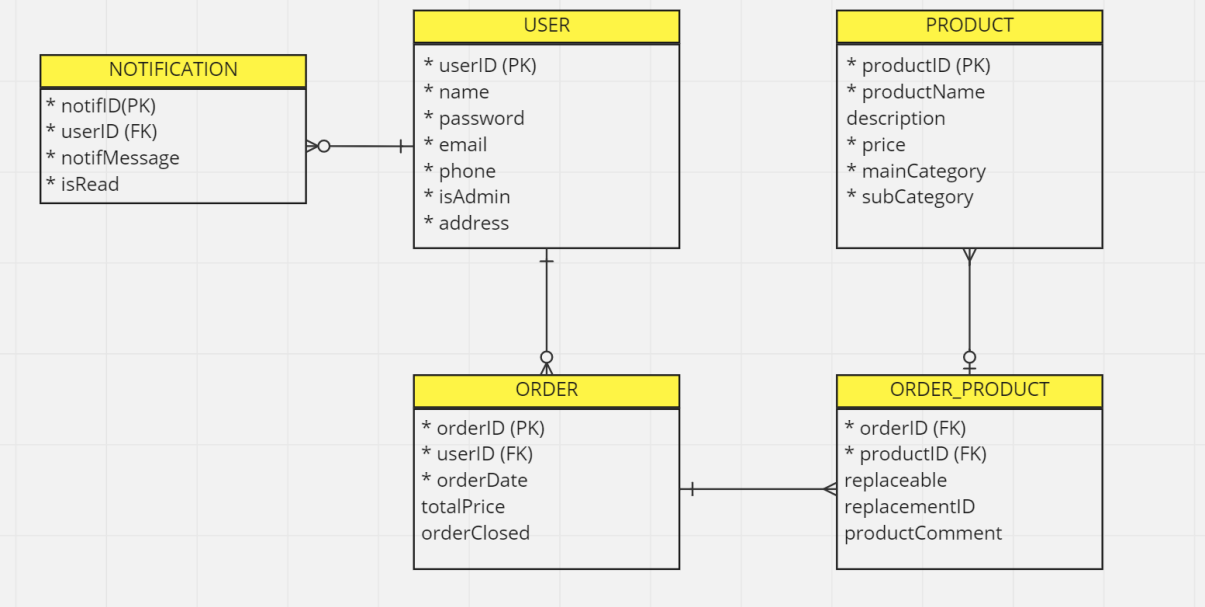
Figur 1 viser en kunde som ønsker et spesifikt produkt. Dette er en forenkling, for ordrene inneholder flere produkter om gangen. Kunden bruker systemet for å søke etter produktet de ønsker, for å så legge det til en handlevogn. Deretter illustrerte vi to muligheter. Om produktet var tilgjengelig eller ikke. Hvis det er, skal det pakkes av en ansatt på Spar Åsane. Hvis produktet ikke er tilgjengelig, skal kunden bli informert om dette sammen med ordre godkjenningen. Delen om varepakking og levering er egentlig utenfor vårt system, men valgene vi tar påvirkes av handlingene ansatte på Spar Åsane gjør som en del av deres vareleveringsprosess.

2.5 - Entity Relationship Diagram

Entity relationship diagram (ERD) er et diagram som visualiserer hva slags forhold som enhetene i en database har til hverandre. Diagrammet hjelper med å kunne forstå hvordan den logiske strukturen i databasen skal være. Diagrammet involverer 3 forskjellige elementer: enheter, attributter og forhold (Peterson, 2023).

Figur 2 viser diagrammet som gruppen lagde som et utgangspunkt for databasen til systemet. De forskjellige enhetene blir representert av boksene. Alle boksene har linjer mellom seg som blir brukt for å visualisere relasjonene de har til hverandre. Relasjonene bruker symbolene på linjene for å vise til om det er “En til en”, “En til mange”, “Mange til en” eller “Mange til mange” relasjon (Peterson, 2023).

Enhetene som kan bli funnet i figur 2 er som følger: Bruker, notifikasjon, ordre, ordre_produkt og produkt. Hver enhet har forskjellige attributter knyttet til seg. Attributtene som har tegnet (PK) er “Primary key”, mens de som har tegnet (FK) er “Foreign key” (Peterson, 2023). Attributtene med en stjerne representerer at de er “not null”, altså nødvendig for hver instans av en enhet. Produksjonen av ER-diagrammet var en viktig prosess for gruppen i starten av prosjektet, fordi på toppen av at det beskrev hvordan systemets database skulle oppstå, så fungerte det også som en pekepinne for gruppen i design fasen.



Figur 2 - Entity Relationship Diagram

Da gruppen lagde diagrammet, diskuterte vi hva hver del av databasen trengte. Først lagde gruppen “user”. UserID er primary key fordi den er unik for hver bruker, genereres av databasen, og er nødvendig for at andre deler av databasen og systemet skal kunne vite hvilken bruker de skal kobles til. Deretter la gruppen til attributter som var nødvendig for prosessen. Alle brukere må ha navn, så Spar Åsane vet hvilken kunde det er. Alle brukere må ha e-post, telefonnummer og adresse, så Spar Åsane har enkel tilgang til deres kontaktinfo.

“Product” var delen av databasen som skulle inneholde alle produktene til Spar Åsane, og dette elementet skulle bli hentet hver gang en bruker så på katalog siden av systemet. ProductID er Primary Key av samme grunn som userID er. Forskjellen er at productID er basert på et produkts EAN kode, istedenfor et tall generert av databasen. De andre attributtene er nødvendig for sortering og visning av hvert produkt. “Description” har ikke “not null” stjerne som de andre attributtene, fordi ikke alle Spar produkter har en beskrivelse.

Enheten “order_product” eksisterer for delen av ordreprosessen der varer må bli lagt til en ordre, uten at de originale unike varene blir påvirket. Denne enheten virker som en bro mellom “order” og “product”. Den inneholder to foreign keys. En for hvilket produkt den representerer, og en for hvilken ordre produktet hører til. Ikke inkludert i illustrasjonen, er at order_product har primary key for en unik ID. “Replaceable” og “replacementID” refererer til hvis et produkt i en ordre skal ha muligheten til å byttes ut med et annet produkt. Kunden bestemmer selv om de vil legge til dette, og disse attributtene kan derfor være null.

2.6 - Risikoanalyse

En risikoanalyse går ut på å finne ut av hvilke risikoer som kan forekomme, hva som får dem til å forekomme og hva man skal gjøre for å eventuelt redusere skaden som påføres. Analysen tar også for seg og setter tall på sannsynligheten for at en risiko kan forekomme og hvor store konsekvenser det kan medføre. Tallene for sannsynligheten og konsekvensene vil deretter bli satt sammen som da representerer risikoen (Aven, 2023). I de første dagene av prosjektet, utførte gruppen en risikoanalyser over tre kategorier. Disse var tekniske feil, kommunikasjonsproblemer, og utfordringer med prosjekt.

Tabell 3 - Risikoanalyse

PROSJEKT	**Konsekvens**	**Sannsynlighet**	**RISIKO**	**Tiltak for at ikke skjer**	**Tiltak for å redusere skade**
Feature creep	3	2	6	MoSCoW prioritering.	Fjern de mest unødvendige ideene.
Forsinkelser	4	3	12	God planlegging, MoSCoW prioritering.	Gjør endringer i planen.
Arbeid blir ikke gjort	5	2	10	God kommunikasjon, Scrum under programmering	Finn hovedårsaken, løs problemet. Gjøre arbeidet i plenum
Medlemmer er redde for å ta opp noe viktig	4	1	4	Skap et hyggelig miljø. Bruk effektivt SCRUM.	Åpne for samtale. Snakk om problemet privat.
Mangel på kritiske funksjoner i sluttprodukt	5	2	10	Bruk MoSCoW prioriteringer. Hold god kommunikasjon og bruk SCRUM.	Skade i sluttprodukt kan ikke reduseres.
Dårlig user experience	3	2	6	Kommuniser med Spar Åsane. Planlegg design nøye.	Gå tilbake til design planlegging, lag et nytt design.

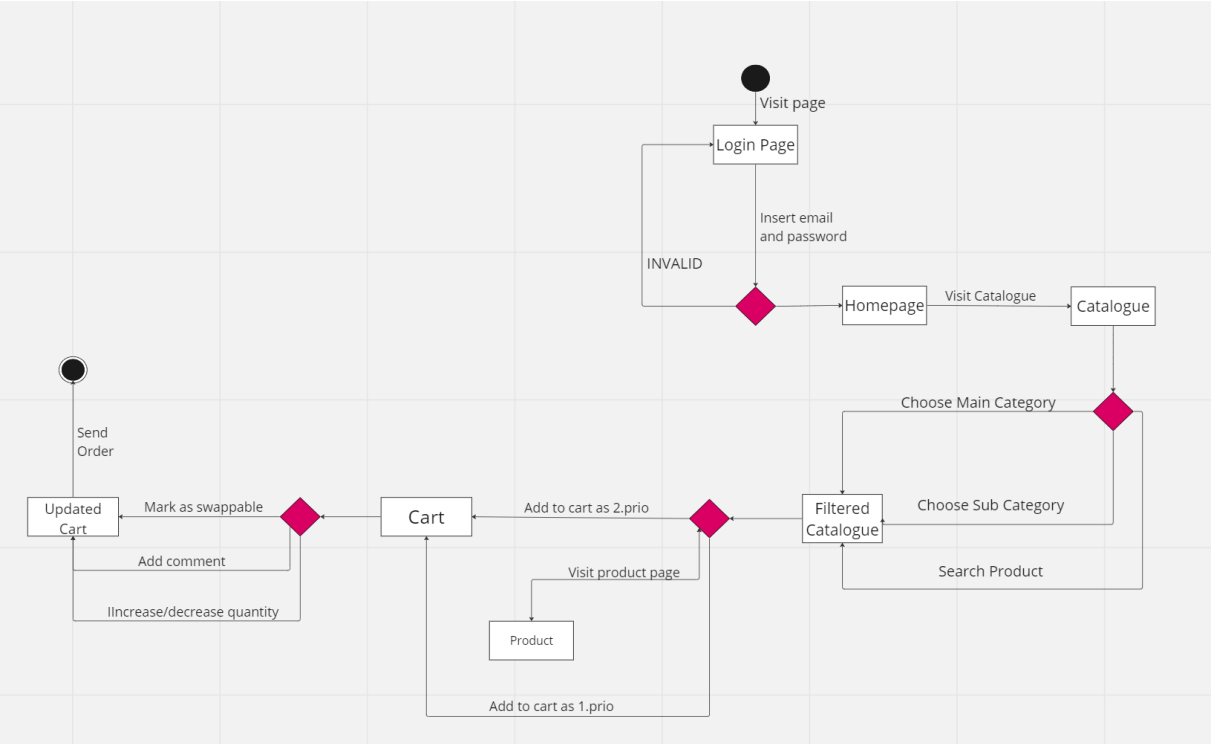
Tabell 3 viser et utsnitt av hele risikoanalysen, som ligger i vedlegg 5. Gruppen har tenkt frem til flere situasjoner som kunne oppstå gjennom prosjektet. Da dette var definert, ble gruppen enige om hvor stor konsekvens og sannsynlighet disse situasjonene kom til å ha. Risiko ble regnet ut basert på konsekvens ganget med sannsynlighet. Deretter skrev gruppen ned hvilke tiltak som måtte til for å unngå at situasjonene skulle skje, som MoSCoW prioritering og god kommunikasjon. Gruppen gjorde det samme for tiltak for å redusere skaden av konsekvensene. Hele risikoanalysen i vedlegg 5 har dette formatet med informasjon for hver situasjon gruppen kom på i forkant av prosjektet.

3 - Design

Dette kapittelet tar for seg arbeidet som gruppen gjennomførte for å visualisere hvordan webapplikasjonen skulle se ut og hvordan den skulle oppføre seg. I kapittelet ble det utarbeidet tilstandsdiagram, navigasjonskart, skisser, wireframes og mockups.

3.1 - Tilstandsdiagram

Et tilstandsdiagram blir brukt for kunne illustrere forholdene i et system. Diagrammet illustrerer oppførselen som tilstander har mot hverandre under overganger. Med et tilstandsdiagram, så kan gruppen kunne illustrere den dynamiske oppførselen i et system. Diagrammet vil kunne vise hva det er som gjør at tilstandene i systemet endrer seg (GeeksforGeeks, 2019).



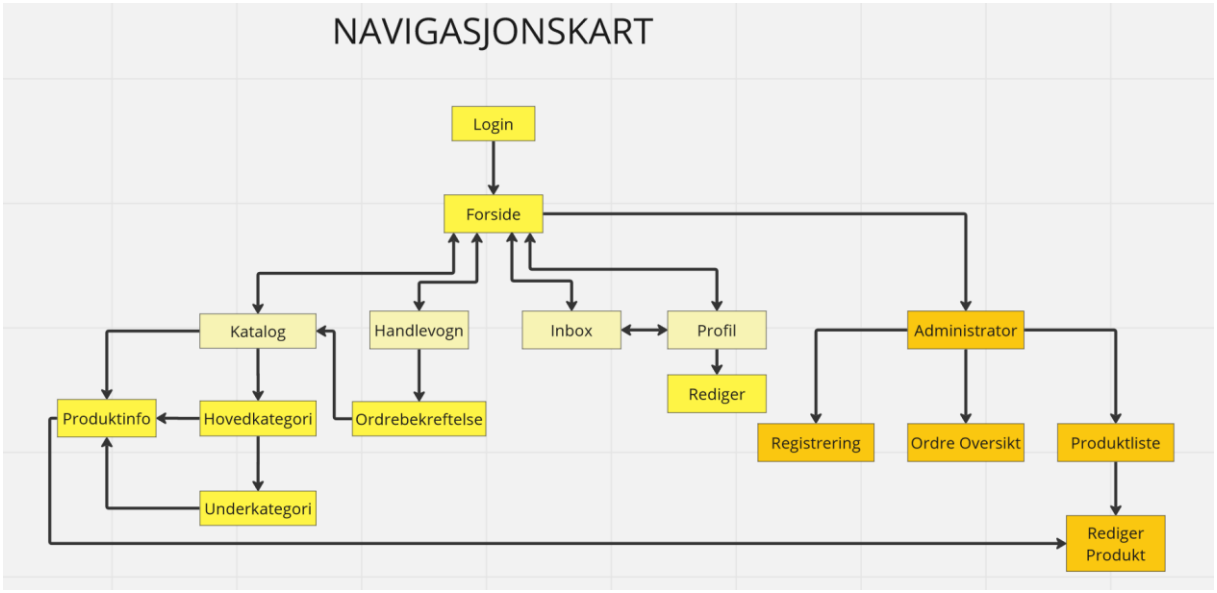
Figur 3 - Tilstandsdiagram

Figur 3 viser et tilstandsdiagram som illustrerer forholdene i systemet vi skal lage for den viktigste delen av kundereisen. Den sorte sirkelen er den første tilstanden i systemet og markerer starten på reisen, som i dette tilfellet er når du først går på nettsiden. Rektanglene representerer en spesifikk tilstand systemet er i som for eksempel en side brukeren er på, mens diamant figurene representerer flere mulige beslutninger som kan tas fra en tilstand. Pilene som går til en diamant, viser til at tilstanden har flere handlinger som kan tas mens pilene fra diamanten er handlingene. På pilene står det hvilke handlinger det er som fører til endring av tilstand (GeeksforGeeks, 2019).

Tilstandsdiagrammet i figur 3 illustrerer kundereisen for å finne produkter og sende order. Det starter med at brukeren går på nettsiden og logger inn. Deretter vil kunden bli bedt om å logge inn med mail og passord, der informasjonen de skriver enten er feil og må prøve på nytt eller riktig og derfor får logget inn. Videre kan brukeren navigere til katalog siden hvor de får muligheten til å filtrere seg gjennom produktene. Deretter kan de legge til produktet til handlekurven som 1.prio eller 2.prio. Eller så kan brukeren klikke seg inn på et produkt og deretter fortsatt ha tilgang til funksjonen for å legge produktet til handlekurv som 1.prio eller 2. prio. Dette blir illustrert ved at pilene går fram og tilbake fra Product rektangelet tilbake til diamanten. Etter det, når de går inn på handlevogn-siden, så får de muligheter til å gjøre forskjellige endringer og til slutt kan de sende ordren. Den sorte sirkelen med det hvite rundt markerer den siste tilstanden i systemet. Tilstandsdiagrammet fungerte som et grunnlag for resten av design fasen i prosjektet.

3.2 - Navigasjonskart

Et navigasjonskart brukes til å kartlegge hvordan brukere kan navigere seg gjennom systemet. Det viser hvilke deler av systemet, som sidene til vårt nettsted, som fører til hverandre og hvordan. Et navigasjonskart i hierarkiformat er en struktur med et hovedelement, som er koblet sammen med flere underelementer som kan videre kobles til flere underelementer (Benyon, 2014, s. 172-173). Navigering av nettstedet skulle være enkelt for brukeren, ifølge kriteriene vi satt i figur 2 Vår plan var at hver side som hører til en spesifikk type funksjon skulle være tilgjengelig fra hverandre.



Figur 4 - Navigasjonskart

Figur 4 er den ferdige versjonen av navigasjonskartet. Gjennom planleggingen av designet vurderte vi hvilke sider vi kom til å trenge i systemet, og hvor man kunne finne dem. Forsiden kan ta brukeren til alle relevante hovedsider: katalog, handlevogn, innboks, og profil. Alle sidene fører også tilbake til forsiden. Ikke illustrert er at hver av hovedsidene også skulle henge sammen gjennom en navigasjonsbar på nettstedet. Grunnen til dette er at da ville alle elementene hatt pil tilbake til de fire hovedelementene, de i egen farge, og dette hadde vært rotete for visualiseringen av navigasjon.

Katalog er elementet med flest underelementer. Vi planla å modellere katalogen etter andre nettbutikker vi fikk inspirasjon fra, og derfor satte vi opp at Hovedkategori og Underkategori var egne sider man kunne navigere til. I løpet av prosjektet fant vi ut at vi ville løse kategorier på en annen måte, og lagde derfor ikke egne sider for kategoriene. Når man trykker på et produkt i katalogen, blir man tatt til Produktinfo siden. Der legger man produktet til handlekurven.

Handlevogn elementet peker kun til ordrebekreftelse som underelement. Etter at kunden har lagt til alle produktene de vil ha, godkjenner de ordren i handlevognen. Dette reflekterer en av brukerhistoriene. Innboks og profil er koblet sammen i tillegg forside-koblingen som de andre siden har. Dette er fordi vi planla at kunden kunne se all relevant info om sin profil/bruker på profilsiden. Fra profil kan man gå til redigering av brukerinfo.

Siste hovedkategori av elementer er Administrator-sidene. Disse markerte vi i en annen farge for å vise at de er ikke tilgjengelige for en vanlig bruker, kun for Spar Åsane sine ansatte. En administrator kan navigere til Registrering av nye brukere, en Ordre Oversikt for å lese og pakke ordre, og en Produktliste som har andre funksjoner enn katalogen. I navigasjonskartet planla vi også at administratorer kunne Rediger Produkt, men senere i prosjektet måtte vi fjerne denne funksjonen.

3.3 - Designprinsipper

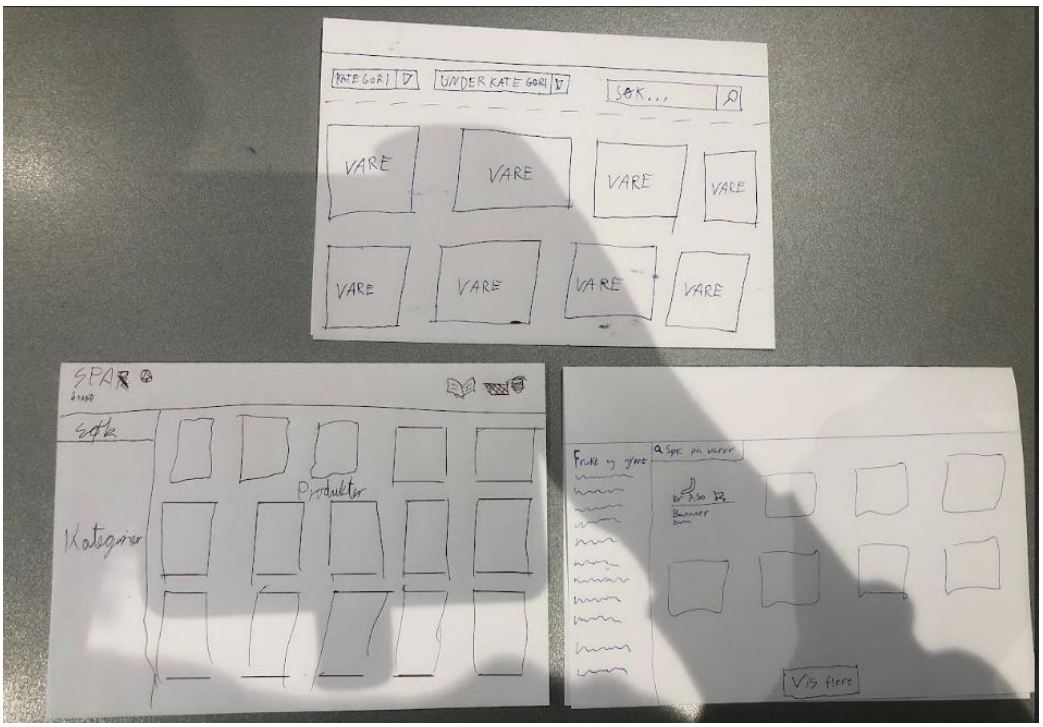
Gruppen valgte å benytte seg av Benyons design principles. Dette var prinsipper som gruppen hadde brukt gjennom studiet som de har hatt god erfaring med. Benyons designprinsipper består av 12 prinsipper og er delt inn i 3 kategorier: accomodation, effectiveness og learnability. Accomodation går ut på å legge til rette for forskjeller mellom mennesker og respektere det, effectiveness går ut på hvor lett det er å bruke og til slutt, learnability går ut på å gjøre det lett å ha tilgang til ting og at det skal være både lett å huske og lære (Benyon, 2014, s. 86).

For at gruppen skal gjøre det lettere for seg selv, har de bestemt seg for å fokusere på noen få prinsipper under designet av systemet. De prinsippene er: familiarity, consistency, navigation, affordance og feedback. Familiarity er basert på å bruke både språk og symboler som brukerne til systemet kommer til å gjenkjenne, for eksempel en handlekurv på en knapp som har som funksjon å legge produkt til handlekurv. Affordance går ut på å gjøre det tydelig hva ting gjør, for eksempel at det er tydelig at en knapp faktisk ser ut som en knapp. Consistency handler om å holde designet av systemet konsistent, for eksempel å gjøre slik at knappene i systemet ser like ut slik at brukerne lettere kan gjenkjenne det. Navigation vil si å gjøre det mulig for brukerne å kunne navigere seg gjennom systemet. Feedback går ut på å jevnlig gi brukerne tilbakemelding på det de gjør i systemet slik at de vet hva slags påvirkning de har, for eksempel at dersom de skriver feil email eller passord ved innlogging, så får de beskjed om det (Benyon, 2014, s. 86-87).

3.4 - Skisser

Skisser er enkle tegninger der hensikten er å få frem ideer og tanker. Ved hjelp av skisser blir det også lettere å kunne se potensielle feil eller ting som kan bli vanskelig (Benyon, 2014, s. 168). Gruppen brukte skisser for å hamre ned et grunnleggende design av nettsiden. Helt spesifikt skulle hver person tegne sin versjon av en gitt side på papir og så ble gruppen enige om hvilken skisse de skulle bruke videre i designet.

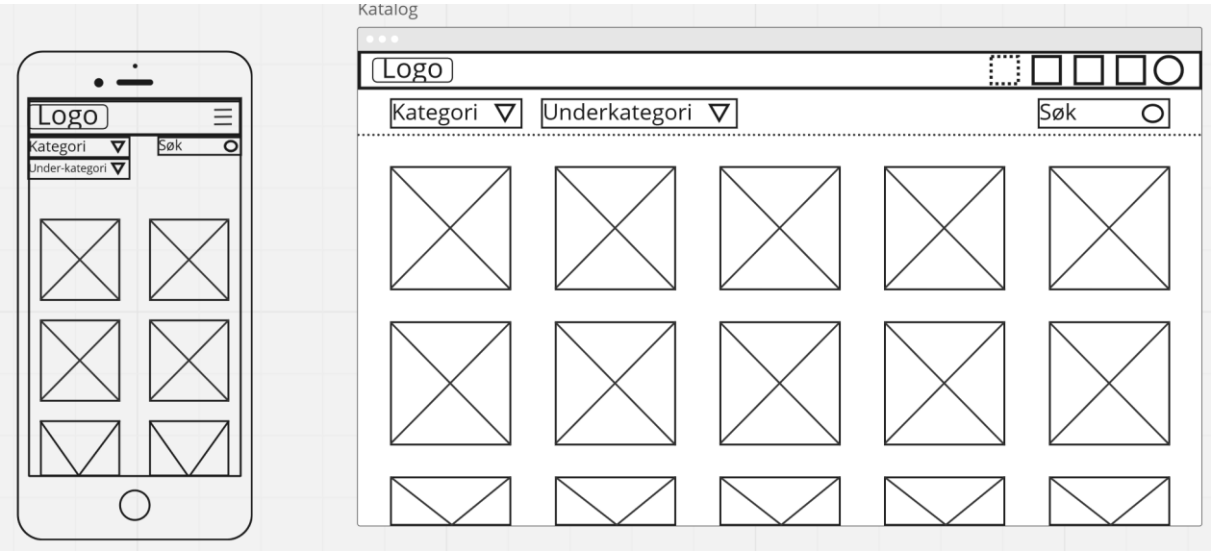
Dette ga mulighet til å velge mellom for eksempel tre forskjellige loginsider eller forskjellige katalogsider, og vi gjorde dette for alle sidene vi hadde planlagt å inkludere i produktet. Dette gjorde også at gruppen kunne, for eksempel, ta i bruk den ene skissen, men hente elementer vi likte fra de andre skissene. Figur 5 viser skisser av katalogen, mens resten ligger i vedlegg 6.



Figur 5 - Skisser av katalog

3.5 - Wireframes

Etter at gruppen hadde laget en skisse hver til alle sidene, lagde vi wireframes basert på designene. Wireframes tar for seg hovedelementene i et design og går ikke så mye i dybden. Små detaljer som for eksempel farger blir ikke vist. Det viktige er å lage et kjapt design der man bare fokuserer på oppsettet og hovedelementer (Benyon, 2014, s. 173-174). Når gruppen lagde wireframes, ble vi først enige om hvilke elementer fra hvilke skisser som vi ville gå videre med.



Figur 6 - Wireframes av katalog siden

Figur 6 viser web og mobil designet til katalog siden. Tekst og former representerer hvor elementene skal være, men ikke hvordan de skal se ut. Dette designet var basert på den øverste skissen i figur 5. Produktene er representert som firkanter med kryss. På toppen av siden er en navbar med sirkler som representerer symboler som fører til andre sider. Gruppen tenkte over designprinsippene vi valgte å fokusere på i designet til vår wireframe. Familiarity og consistency viser vi med designet som ligner andre nettbutikker, og at hvert produkt var planlagt å ha samme oppsett og funksjonalitet for hver krysset firkant.

Siden vi valgte dette designet, måtte vi endre på vårt navigasjonskart som nevnt tidligere. De andre skissene fulgte planen om å ha kategori og underkategorier som egne sider. Da gruppen lagde wireframe, måtte vi gå bort fra den originale planen. Wireframes for mobil designet er basert på web-designet, men elementer måtte kuttes for bedre plass. Gruppen lagde ikke wireframes for mobil for administrator-sidene, siden Spar Åsane sa de ikke trengte at administratorer skulle ha tilgang til dette på mobil. Resten av wireframes, og mobil designene kan man finne i Vedlegg 7. Designvalgene gruppen tok da vi gjorde skissene til wireframe kan bli funnet i tabell 4.

Tabell 4 – Designvalg tatt av gruppa fra skisser til wireframes

Login	Høyre skisse, men med en logo over. Bilde på bakgrunnen.
Profil	Øverste skisse, men med en ny side når man trykker på ordrehistorikk.
Forside	Øverste skisse.
Katalog	Øverste skisse med dropdown for kategori og underkategori.
Produktinfo	Kombinasjon av øverste og høyre skisse. Meny på venstre side med kategorier. Øverste layout men med mindre bilder og elementer.
Handlevogn	Høyre skisse, men med mindre checkbox. Legg til en kommentarboks.
Inbox	Kombinasjon av venstre og høyre skisse. Venstre skisse ute "sendt" og "arkiv", sammen med høyre skisses "unread" markør og "slett" knapp.
Admin	Øverste skisse, men med midtstilte elementer.
Ordreliste	Øverste skisse, men med høyre skisse sin ide om egen side.
Produktliste	Øverste skisse, men med egen side for produktene.

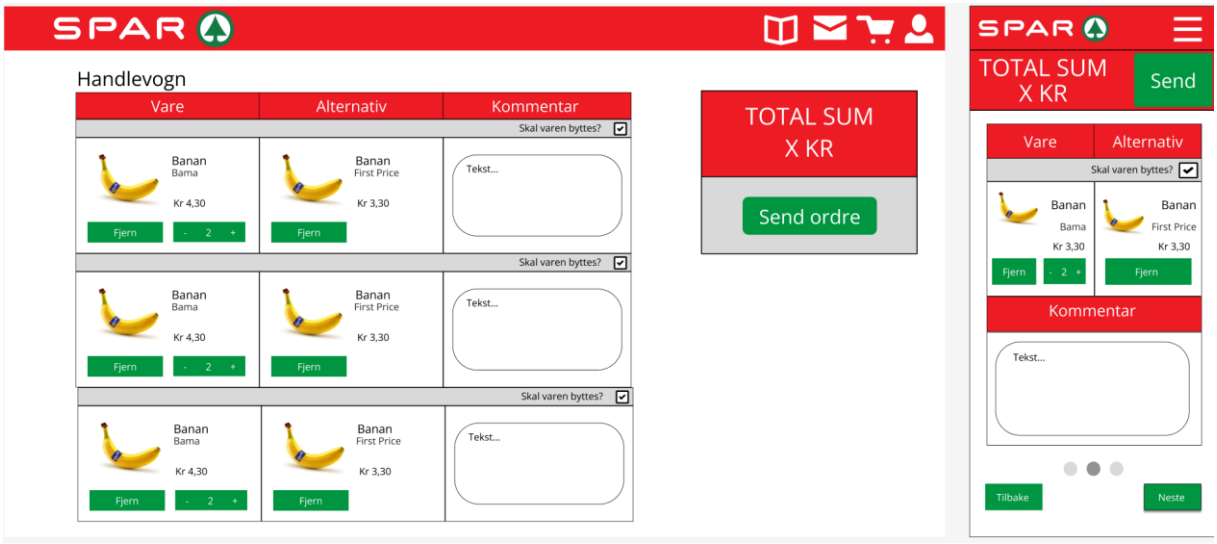
3.6 - Mockups

Etter at gruppen lagde wireframes og fikk feedback på det fra produkteier, gikk vi deretter over til å lage mock-ups. Mock-ups er det endelige designet av hvordan systemet kan se ut. Det som differerer mock-ups fra wireframes er at når man lager mock-ups så inkluderes små detaljer som typografi, farger og stilen til systemet generelt. Det gir gruppen også muligheten til å enklere kunne vise frem designet til produkteier og få tilbakemelding (Hannah, 2023).

Figur 7 - Utvalgte farger

Før gruppen begynte å lage mock-ups, så ble de, sammen med produkteier, enige om å bruke farger basert på Spar sine. Figur 7 viser de tre fargene som det gjelder. Gruppen valgte å bruke hvit som bakgrunnsfarge. Deretter ble den røde fargen brukt i navigasjonsbaren og for titler på sider. Til slutt skulle denne grønne fargen bli brukt for knapper og lenker. De grønne og røde fargene skulle bli komplimentert med hvit tekst som passer godt med tanke på kontrast, men også for å beholde Spar sine farger. Gruppen ble

også enige om å bruke Figma, ettersom at dette var et verktøy vi har hatt god erfaring med tidligere.

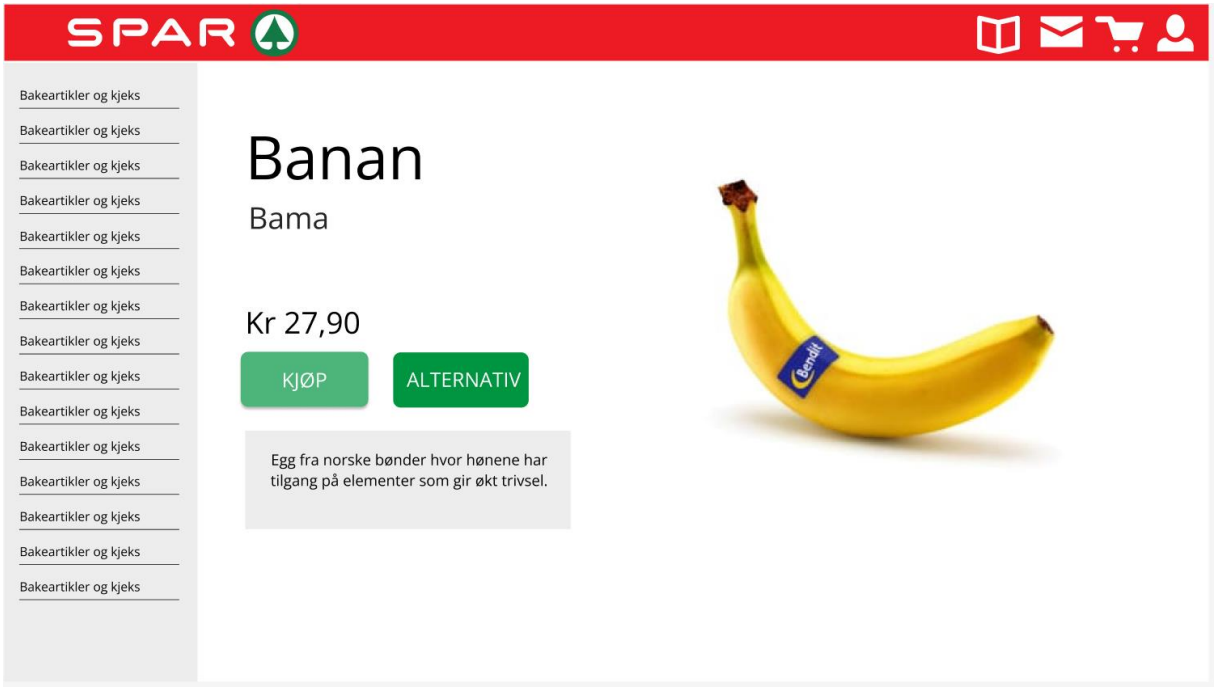


Figur 8 - Mockup av handlekurven

Da gruppen jobbet med å lage mock-ups hadde vi Benyons designprinsipper i bakhodet og fokuserte spesielt på de fem prinsippene nevnt tidligere. Figur 8 viser noen mock-ups for både mobil og desktop og hvordan vi håndterte bruken av designprinsippene. Consistency var et av prinsippene gruppen hadde fokus på og det kommer frem godt i figur 8. Her blir de samme grønne knappene brukt flere steder som vil gjøre det lett for brukerne og gjenkjenne dem. Vi har gitt knappene rektangelformer og tekst inni som tyder på hva slags handling som utføres ved å trykke på dem og dette går under designprinsippet affordance.

Figur 9 - Navigasjonsbar

Figur 9 viser navigasjons-baren og hvordan gruppen har håndtert designprinsippet familiarity. Her har vi brukt ikoner som brukerne allerede er godt kjent med, for eksempel en handlekurv hvor de da forventer å bli tatt til en handlekurv side dersom de trykker på den. Som oftest på nettsider, pleier navigasjonsknapper i navigasjonsbarer å forholde seg på høyre side, derfor har vi valgt å gjøre det samme igjen for å forholde oss til designprinsippet om familiarity.



Figur 10 – Produktinfo

Figur 9 og 10 viser hvordan gruppen har fokusert på design-prinsippet navigation. Vi har gitt brukerne flere muligheter til å kunne bevege seg rundt i systemet, først og fremst så har de navigasjonsbaren som alltid er tilgjengelig, men i figur 13 ligger det en kategori listen til venstre som de også kan bruke til å navigere til andre produkter.

Figur 11 - Knapper som lyser ved å holde musa over

Feedback var også et prinsipp gruppen hadde mye fokus på. Et eksempel på hvordan vi har fulgt dette prinsippet er når brukere holder musa over så vil knappen bli lysere som illustrert i figur 11. På denne måten får kunden tilbakemelding fra systemet at de holder musa over noe som er klikkbart.

Etter at gruppen hadde gjort klart sitt første utkast av mock-ups, ble dette vist frem til prosjekteier hvor gruppen fikk gode tilbakemeldinger om det som var bra og det som måtte fikses opp i. Resten av mock-ups some gruppen lagde kan bli funnet i vedlegg 8.

3.7 - Resultat

https://www.youtube.com/watch?v=YryXoVmvyJk

4 - Språk og Verktøy

Dette kapittelet tar for seg språkene og verktøyene gruppen har valgt å bruke for dette prosjektet. Gjennom studiet her på Universitet i Agder har alle gruppemedlemmene fått veldig god erfaring med forskjellige type språk og verktøy og siden produkteier har latt oss stå fritt når det gjelder valg av teknologier, så har vår egen erfaring spilt en stor rolle under valget av språk og verktøy.

4.1 ClickUp

Når gruppen skulle velge et prosjektstyringsverktøy under utviklingsdelen, bestemte gruppen seg for å velge ClickUp. ClickUp er et samarbeidsverktøy som alle på gruppen har hatt god erfaring med tidligere. Verktøyet gjør det lett for deg å fordele de ulike arbeidsoppgavene via et innebygd dashboard. ClickUp gir gruppen muligheten til å kunne tilpasse verktøyet slik gruppen vil ha det. Blant annet så kan man bruke funksjoner som å estimere tid, dele ut sprint points og sette prioritasjon.

4.2 Frontend

I Frontend ble gruppen enige om å bruke rammeverket/biblioteket React med Typescript som programmeringsspråk. Typescript er en utvidelse av programmeringsspråket Javascript. Det som er bra med Typescript er at den sjekker koden for feil før den kjøres. Det fører til mer strengere bruk for å unngå feil etter at koden har kjørt (Microsoft, 2023). React er et populært komponent basert bibliotek/rammeverk støttet av Meta. Gruppen havnet på disse to teknologiene i Frontend etter at det ble foreslått av en på gruppen ettersom at den personen hadde hatt god erfaring med det tidligere. React har også et veldig stort økosystem som gruppen mente kommer til å gi dem flere og bedre muligheter (Meta Open Source, u.å).

4.3 Backend

I backend bestemte gruppen seg for å bruke Node med rammeverket Express. Node.js er et “open-source and cross-platform Javascript runtime environment”. Open-source betyr at kildekoden er åpent tilgjengelig for alle. Koden er lagt ut på nett, ofte på versjonskontroll nettsider. Hvem som helst kan se over koden, opprette egne versjoner, og kan foreslå endringer til kilden. Dette gjør at open-source kildekode kan vedlikeholdes av flere utviklere. Javascript runtime environment vil si at den kan kjøre Javascript kode, som tidligere bare var mulig for nettlesere (Semah, 2022). Express er et rammeverk for Node som gjør det enklere å bygge webapplikasjoner (GeeksforGeeks, 2023). Gruppen endte opp på disse to

teknologiene på grunn av sitt store økosystem, tidlig erfaring fra noen i gruppen og fordi det vil gjøre det mulig for Frontend og Backend å forstå hverandres kode, ettersom Javascript/Typescript blir brukt på begge sider. Dette vil kunne føre til mer teamwork som for eksempel pair programming og code review. Et problem gruppen kom bort i med Node pakker, var at vi trengte en forståelse over hver spesifikk pakke. Dette førte til at gruppen av og til ikke forsto hvordan en funksjon fungerte, og i et tilfelle måtte lage egen funksjon selv. Gruppen møtte også på et problem hvor utvikler miljøet ga feilmelding selv om Node pakken fungerte.

4.4 Postman

Postman er et API-verktøy som brukes for å lage og teste API. Programmet tilbyr API klient, API dokumentasjon, API testing, og mer. (Postman, u.å). Gruppen brukte spesielt dette verktøyet for testing av funksjoner på backend. Frontend sender API kall til backend med informasjon som backend bruker i koden. Ved å sende denne informasjonen gjennom Postman, kunne gruppemedlemmene som jobbet på backend teste informasjonen frontend skulle gi før koden var ferdig. Postman var også relevant for å finne ut hvordan APIen gruppen brukte for informasjon om alle varene fungerte.

4.5 Database

Databasen gruppen har valgt å bruke er MySQL. Dette er fordi gruppen først og fremst var ute etter et relasjonelt database verktøy hvor vi kunne definere, manipulere, og kontrollere data via “Structured Query Language”, fremfor et ikke-relasjonelt databaseverktøy. Dette er fordi gruppen mener det passer best for prosjektet sitt. Derfor valgte gruppen MySQL som vi har hatt god erfaring med tidligere gjennom studiet. MySQL er et relasjonelt database styringssystem som er et av de mest populære open-source databasene i verden (Oracle, u.å). Gruppens database følger ER-diagrammet i Figur 2, med endringer som ble lagt til gjennom prosjektet.

```sql
CREATE TABLE IF NOT EXISTS users (
    userID int NOT NULL AUTO_INCREMENT,
    userName varchar(50) NOT NULL,
    userPass varchar(60) NOT NULL,
    email varchar(255) NOT NULL,
    phone varchar(20) NOT NULL,
    isAdmin smallint(1) NOT NULL DEFAULT 0,
    address varchar(50) NOT NULL,
    registerDate date NOT NULL DEFAULT CURRENT_DATE,
    passCreated smallint(1) NOT NULL DEFAULT 0,
    PRIMARY KEY (userID)
);
```

Figur 12 - Eksempel på SQL query for database setup

4.6 Utvikler Miljø

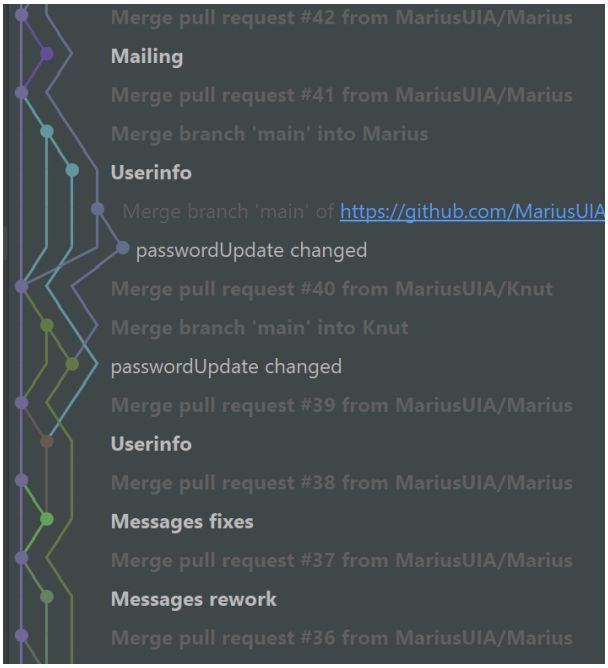
Utvikler miljøet som gruppen valgte å bruke for dette prosjektet kalles WebStorm som brukes med Javascript. Gruppen valgte dette verktøyet på bakgrunn av gruppen sin tidligere erfaring med lignende produkter. JetBrains, en programvareleverandør, tilbyr forskjellige typer utviklingsmiljøer, blant annet WebStorm. IntelliJ, som er ment for programmeringsspråket Java, er en annen en som gruppen har hatt god erfaring med gjennom studiet, og selv om JetBrains utviklingsmiljøer er forskjellige med tanke på hvilke språk de spisser seg inn på, så holder de på fortsatt mye av det samme. JetBrains sine produkter kommer med mange nyttige funksjoner og en spesielt viktig en som gruppen liker er innebygd versjonskontroll (Jetbrains, 2023).

4.7 Versjonskontroll

For versjonskontroll brukte gruppen GitHub. Organisert bruk av GitHub lot gruppen jobbe sammen på prosjektet uten konflikter. Gruppen satte opp to forskjellige repositories, en for kun frontend og en annen for kun backend. Dette gjorde vi fordi hver del kunne jobbes på uten å være koblet sammen. Frontend og backend kommuniserer bare gjennom API kall, derfor er delene uavhengige. Gruppen har hatt rutiner for når hver skulle ta inn endringer fra Git til sin lokale maskin, og har jobbet sammen hver gang systemet skulle testes og settes sammen.

Versjonskontroll har hjulpet med kvalitetssikring ved at gruppen har fått oversikt over alle endringer i koden. Alle på gruppen brukte en egen “branch” når vi programmerte. Dette gjorde at gruppen unngikk å skrive over hverandres kode, og lot oss teste ut endringer uten at det påvirket andre. Når en endring skulle legges til main-branch, måtte den først pushes til en remote branch som personen jobbet på. Deretter opprettes en pull request til main-branch. De fleste tillegg til main-branch kunne bli “merged” uten problem. Da det oppsto

konflikter, gikk gruppemedlemmet som var utnevnt til Git-sjef gjennom koden og fikset konfliktene. Figur 13 viser hvordan branches ser ut etter en del merging.



Figur 13 - Visualisering av branches etter merging

5 - Systemutvikling

I dette kapittelet skal gruppen diskutere viktige beslutninger som ble tatt under utviklingsprosessen.

5.1 Kompetanseutvikling

Før vi begynte selve utviklingen av systemet så trengte gruppen tid til å lære seg teknologiene Javascript/Typescript, React, Node og Express. Vi bestemte oss for å sette av minimum 14 dager til å lære oss disse teknologiene. Deretter skulle vi spørre oss selv på den siste dagen om vi følte at vi var klare. Hver dag skrev alle i gruppen et lite sammendrag om hva enhver hadde lært den dagen, og hvilke kilder de hadde lært fra. Dette gjorde vi for å dele kunnskap og lære dette sammen. Den siste dagen ble gruppen enige om at vi kunne begynne på selve utviklingen av systemet. To på gruppen følte i ettertid at dette var en god ide å begynne på selve utviklingen tidligere og lære samtidig når man utviklet, men på en annen side ønsket tredjemann på gruppen at han hadde hatt mer tid å lære teknologiene på. Til tross for dette begynte vi likevel å utvikle systemet. Dette gjorde gruppen fordi vi følte at vi kunne fortsette å lære mens vi programmerte, og fordi vi ville ikke sløse kostbar tid.

5.2 Backend

Backend ble programmert med Node.js, og brukte routes for å kommunisere med frontend. To på gruppen jobbet sammen på flere deler av prosjektet, og individuelt med noen funksjoner. Dette kapittelet forteller om valg gruppen tok gjennom utvikling av backend, og hvilke prioriteringer som ble gjort.

5.2.1 API

For at hele prosjektet skulle være mulig, måtte vi ha en liste over alle varer som Spar selger. Hvordan vi skulle få dette til endret seg gjennom prosjektet. Originalt planla gruppen å bruke kode fra en GitHub repo som koblet seg til en offentlig Spar API. Dessverre var dette flere år gammelt, og APIen den koblet til var ikke lenger tilgjengelig. Etter dette, snakket gruppen med Spar Åsane om det var mulig å eksportere deres lokal database til bruk i vårt prosjekt. Dette viste seg å ikke være mulig, også med tanke på datasikkerhet. Til slutt fant gruppen https://kassal.app/api/. En API for flere dagligvarer som henter priser for alle deres produkter hver dag. Denne APIen hadde alt prosjektet trengte. I møte med Spar Åsane fikk gruppen godkjent at informasjonen om varene hentet stemte med varene som Spar Åsane selger.

```
for(let j = 0; j < data.data.data.length; j++) {
    console.log("Product: " + j);
    if(data.data.data[j].store.name === "SPAR") {
        const {ean, name, current_price, description, url} = data.data.data[j];
        console.log(parseInt(ean) + "----" + name + "----" + current_price.toString()  + "----" + d
        const categories = url.split("/");
        console.log(categories[5] + "---" + categories[6]);
        await pool.query( sql: "INSERT INTO products (productID, productName, price, mainCategory, su
            values: [ean, name, current_price.toString(), categories[5], categories[6], description])
    }
}
```

Figur 14 - Screenshot av API der rundt 8000 produkter lagt til i databasen

APIen hadde noen problemer som gruppen ikke fant ut av før senere i prosjektet. Å laste inn alle varene tok lang tid, og tok mye av ressursene til datamaskinene under prosessen. En i gruppen møtte en feil som gjorde at han manglet et par tusen produkter i sin liste, og dette ble ikke oppdaget med det første. Et annet mindre problem var at flere av produktene hadde feil i navnene, som gruppen selv måtte manuelt fikse. Det største problemet APIen førte til for gruppen, var at vi måtte avbryte planen om en produktinstillings-side. Hvis vi skulle la Spar Åsane manuelt redigere produkter, måtte vi passe på at disse endringene ikke hadde blitt overskrevet av daglige oppdateringer av APIen. Hvis vi hadde gjort dette, måtte gruppen ha endret på databasen for å gi plass til manuelle endringer, og lagt en ny side i systemet der Spar Åsane kunne se over deres endringer. Dette kunne skapt problemer med automatisk oppdatering, og var generelt mer komplisert enn det Spar Åsane ønsket..

5.2.2 Authentication og Authorization

For Authentication og Authorization, valgte gruppen å ikke bruke servicer for dette som allerede eksisterer og heller gjøre det selv ved hjelp av cookies og jsonwebtoken. Ved å gjøre det selv så får vi mer kontroll og frihet istedenfor å måtte være avhengige av noen andre.

Login sammenligner hashed passord som allerede ligger i databasen for sikkerhet. Når passord som blir skrevet inn og hashed passord er likt etter dekryptering, starter autentiseringsprosessen. Brukerens ID og om de er administrator eller ikke, blir lagret som en cookie i nettleseren. Hver side i systemet, utenom login, sjekker om denne cookien eksisterer. Det sjekkes også om bruker ID stemmer overens med ID i systemet. For administrator sidene, sjekkes også om bruker er administrator eller ikke. Når autoriseringen feiler sender backend info om dette til frontend, som så fører brukeren tilbake til sider de har tilgang til. Ved å lage egen funksjoner for dette kunne vi ha kontroll og oversikt over hva som skjer når funksjonen kjører, og hvilken sikkerhet den tilbyr. I ettertid var gruppen veldig fornøyde med dette valget og møtte ikke på noen problemer ved å gjøre Authentication og Authorization selv.

5.3 Frontend

Fra gruppen var det en person som hadde ansvar for brukergrensesnittet. Dette underkapittelet vil ta for seg sentrale beslutninger som ble gjort i Frontend.

5.3.1 Localstorage

Localstorage er en oppbevaringsmetode i nettleservinduet. Den tillater nettsider til å lagre data som deretter vil oppbevares og bli brukt neste gang man besøker siden. En av de negative aspektene ved LocalStorage er at data som blir lagret ikke er trygt ettersom at dersom en annen person bruker maskinen,så vil de også kunne få tilgang til nettsiden sin LocalStorage. En annen nedtur er at for mye data lagret vil gjøre nettsiden tregere (Obaseki, 2023).

I Frontend ble det bestemt å bruke LocalStorage for lagring av handlekurv data. Denne beslutningen kom gruppen frem til fordi vi mente at det var unødvendig å bruke databasen til det og det hadde ført til unødvendige frem og tilbake mellom Frontend og Backend. Gruppen følte seg trygge med denne beslutningen av følgende grunner: handlekurv data i LocalStorage er ikke sensitiv informasjon og det involverer ikke mye data som potensielt kan gjøre web-applikasjonen treig.

5.3.2 CSS

Når det kommer til CSS, så finnes det mange rammeverk som kan øke effektiviteten under utviklingen av brukergrensesnitt. Noen populære rammeverk er blant annet Bootstrap og Tailwind CSS. Et negativt aspekt blant disse rammeverkene, er at de kommer med mye ferdigskrevet lagde komponenter og stiler, noe som vil redusere friheten til utvikleren. Det er også stor sjanse for at man ikke bruker alt sammen som følger med rammeverket, som kan føre til at systemet unødvendig blir tregere. Det er også tidkrevende å lære seg et nytt rammeverk for CSS. (Batista, 2023).

I Frontend ble det bestemt å ikke ta i bruk noen slike rammeverk og heller bare bruke vanlig CSS. Sånn ble det fordi personen fra gruppen som jobber Frontend føler seg komfortabel med vanlig CSS og liker friheten som kommer med det hvor man har mer kontroll. Da slipper også gruppen å bruke ressurser på å lære et nytt rammeverk og heller bruke tiden på noe mer produktivt.

5.3.3 Desktop-first

Når man begynner å skrive CSS for en nettside som har både et mobildesign og et design for datamaskiner så er det viktig å tenke på hvilken av dem du skal skrive CSS for først. Da må du enten velge mobile-first, altså å designe for mobil først og deretter legge til media queries for styling til større skjermer og bygge seg oppover i skjermstørrelse, eller desktop-first hvor man da bygger seg nedover. Som regel er det anbefalt mobile-first fordi det er lettere og CSS for dataskjermer er vanligvis vanskeligere å skrive fordi det innebærer flere faktorer (Powell, 2020).

I dette prosjektet ble det bestemt å ta i bruk desktop-first basert på tidligere erfaring. Personen fra gruppen som har ansvaret for Frontend har erfaring med å kode desktop-first tidligere og foretrekker den måten fordi det føles mer naturlig, til tross for at det er anbefalt å gjøre motsatt.

Desktop-first endte for dem meste opp som en god ide, men det var noen ganger mobile-first kunne vært mer nyttig. Et eksempel på dette var handlekurv siden som kunne være lettere å designe dersom man startet med mobildesign. Derfor tenker gruppa at det kunne vært smart å heller gå for en hybridversjon der man for det meste holder seg til desktop-first, men bytter over til mobile-first der man ser det kan være nødvendig.

5.4 Testing

Da gruppen jobbet med utviklingen av systemet, ble det gjennomført diverse type tester av web applikasjonens funksjonaliteter og oppførsel. Testing av et system har flere gode sider til seg, det sparer utviklere tid i fremtiden, det er lettere å finne feil i koden, det kan få deg til å tenke mer over feil og bugs og generelt forbedre kode og kvaliteten i prosjektet. Når man tester et system så er det flere måter å gjøre det på. Det er blant annet manuelle tester og automatiserte tester.

5.4.1 Backend Testing

På grunn av manglende kunnskap innenfor testing hos noen i gruppen og ikke nok tid under kompetanseutviklings tidsrommet, ble det bestemt at det bare skulle bli utført manuelle tester i backend. Dette ble gjort via Postman. Hadde gruppen hatt mer erfaring i testing av backend, hadde vi implementert tester på alle sidene for å sjekke om funksjonene fungerte. At vi måtte gjøre testing manuelt, førte til at vi brukte mer tid enn nødvendig for å løse feil og kan ha ført til dårligere kvalitetskontroll enn ønsket.

5.4.2 Frontend Testing

Personen som hadde ansvar for Frontend hadde tidligere erfaring med testing og dette tillot personen å kode i form av en populær utviklingsprosess som kalles for Test Driven Development (TDD). TDD går ut på å skrive tester for hver funksjonalitet og oppførsel som feiler i starten og deretter lage funksjonaliteten og oppførselen for å passere testene (Hamilton, 2023).

Dette ble gjort av flere grunner. Det gir gruppen mer trygge på at systemet faktisk fungerer og det blir lettere å sørge for at funksjonen/oppførselen oppnår det den skal (Hamilton, 2023). En annen ting som gruppen mener er viktig basert på erfaring, er at det fører til at når man skriver tester først, så blir det lettere å tenke over hva funksjonaliteten/oppførselen krever.

I Frontend ble det utført 3 typer av automatisert testing. Unit, Integration og E2E tester. Unit testing går ut på å teste spesifikke deler av et system i separasjon og uten avhengigheter. Integration tester går ut på å teste spesifikke deler av et system med avhengigheter, for eksempel hvordan forskjellige komponenter i et system interagerer med hverandre (Mahmoud, 2021).

For Unit og Integration tester ble det tatt i bruk en pakke fra React som kalles React-testing library og Vitest som er et test rammeverk. For å kunne lage “mocks” av API requests til å bli brukt for testformål, så ble det tatt i bruk Mock Service Worker, som er et API Mocking verktøy som avskjærer requests til serveren i nettverk-nivå. End-2-End (E2E) testing går ut på å simulere hvordan en bruker hadde interagert med systemet i en spesifikk situasjon, som for eksempel å trykke på en knapp og skrive inn tekst (Mahmoud, 2021). Playwright er et rammeverk for E2E testing som gruppen tok nytte av. Dette verktøyet ga oss muligheten til å kunne utføre E2E tester på flere nettlesere som for eksempel Google Chrome, Safari og Firefox.

Test Driven Development viste seg til å være veldig verdifullt for personen i Frontend og er sikker på at det var det riktige valget. Det ble lettere å kode nettsidene når man allerede hadde skrevet tester for det på forhånd fordi det ga mye innsikt til hvordan nettsiden måtte kodes. Det sparte også personen mye tid. Unit og Integration tester var veldig nyttige, men det samme kan ikke bli sagt for E2E tester. Personen i Frontend følte at tiden det tok for å skrive E2E tester og verdien som ble gitt tilbake ikke gjorde det verdt det. Frontend utvikleren mener at Unit og Integration tester var nok for å teste systemet.

5.5 Bruk av tredjepart bibliotek

Et tredjepart bibliotek er kode som er skrevet av noen andre hvor da hvem som helst kan bruke det i sin egen kode (Schneider, 2021). Som nevnt tidligere har Javascript/Typescript, React, Node og Express store økosystemer og dette bringer så klart med massevis av tredjepart biblioteker. Dette er så klart veldig gode nyheter for gruppen fordi det gir oss en del muligheter til å bruke eksisterende kode som kan brukes i prosjektet og det vil spare oss mye tid.

Men tredjepart biblioteker er noen man bør være veldig forsiktige med. Hver gang man legger til et tredjeparts bibliotek i sin kode, så er det også en avhengighet som man legger til. Dette resulterer i at gruppen må følge med på for eksempel om det kommer en ny versjon av biblioteket eller om den nye versjonen skaper problemer på grunn av en annen avhengighet som også er lagt til fra et annet bibliotek. En annen viktig faktor er at det blir satt mye tillit til de som lagde og opprettholder biblioteket. Kan man stole på at de fortsetter å opprettholde biblioteket og at de sørger for at det er sikkert å bruke? (Schneider, 2021).

På grunn av dette har gruppen bestemt seg for å lage noen grunnregler for bruken av et tredjeparts bibliotek.

1. Må ha god dokumentasjon for bruk av bibliotek.
2. Må ha en del nedlastinger.
3. Må ha blitt oppdatert tidligst i 2022.
4. Må være eldre enn 1 år.

Dersom biblioteket har god dokumentasjon for hvordan det brukes, så vil det bli lettere å tilpasse det i prosjektet sitt og setter også opprettholderne i et bedre lys fordi de virker mer profesjonelle. Dersom biblioteket har en del nedlastinger fra før av, så betyr det at det er populært å bruke og har et samfunn bak seg. Dersom biblioteket ble oppdatert tidligere enn 2022, så kan det indikere at biblioteket ikke blir opprettholdt lenger. Dersom biblioteket er yngre enn 1 år, så er det større sjanse for at biblioteket ikke kommer til å bli opprettholdt (Schneider, 2021).

6 - Prosjektgjennomføring

Dette kapittelet handler om valg av prosjektstyringsmetodikk og hvordan det ble brukt av gruppen gjennom utviklingsfasen av prosjektet. Kapittelet vil også ta for seg viktige begreper i den valgte prosjektstyringsmetodikken.

6.1 Scrum

Gruppen bestemte seg for å bruke Scrum som prosjektstyringsmetodikk for programmeringsdelen av prosjektet fordi gruppen syntes det passet bra for prosjektet og vi har hatt god erfaring med det tidligere gjennom bachelorstudiet. Gruppen kunne også ha valgt fossefall eller kanban som prosjektstyringsmetodikker, men gruppen hadde lite erfaring med disse. Fossefall er også ikke en agil metodologi, som gruppen ønsket for dette prosjektet (Lotz, 2018).

Scrum er en agil prosjektstyringsmetodikk som går ut på at et team deler opp arbeidet sitt i små mengder og utfører det i korte perioder hvor de etter hver periode kan få tilbakemeldinger og dermed gjøre endringer forbedre kvaliteten (Scrum.org, u.å.-a). Når man benytter seg av Scrum, så kreves det et Scrum Team som i gruppen sitt tilfelle består av utviklere, en produkteier og en Scrum Master. Kapittel 6.2 går mer i dybden om disse rollene. Videre har Scrum spesielle eventer som man går gjennom. Alle disse eventene foregår i det som kalles for en Sprint. (Schwaber & Sutherland, 2020). Sprints foregår i korte perioder, i dette prosjektet varte dette som regel i 2 uker utenom en av dem som varte i 1 uke.

Eventene kalles for Sprint Planning, Daily Scrum, Sprint Review og Sprint Retrospective. Disse eventene blir også beskrevet i mer detalj senere i kapittelet. Scrum involverer også 3 viktige “Artifacts” som kalles Product Backlog, Sprint Backlog og Increment som også er skrevet om senere i kapittelet (Schwaber & Sutherland, 2020).

Gruppen valgte Scrum fordi vi synes det passet best til prosjektet, men Scrum har også noen negative trekk. Siden Scrum er agilt, så er det forutsetning at spesifikasjoner endrer seg hele tiden gjennom prosjektet. For noen type prosjekter, er dette ikke ønskelig. Scrum krever også at man følger metodikken nøye, hvis ikke oppnår man ikke målet med Scrum og prosjektets metodikk vil mangle identitet. Gruppen har prøvd å følge Scrum konsepter gjennom prosjektet for å oppnå agil metodikk så nær som mulig. Hvis vi trodde at vi ikke kom til å klare å følge dette, kunne vi ha valgt å gjennomføre en annen, mer passende metodikk.

6.1.1 Scrum Roller

Et Scrum team består som regel av en produkteier, en Scrum master, og et lite antall utviklere. Produkteier har ansvar for at Scrum Teamet er klare over hva det er de jobber mot, altså hva målet med prosjektet er. Det er de som har hovedansvaret for Product Backlog (Schwaber & Sutherland, 2020). I Scrum Teamet var produkteier daglig leder hos Spar Åsane.

Scrum master skal lede gruppen i implementering av Scrum rammeverket og hjelpe gruppen med å forstå hvordan de skal benytte seg av Scrum. Det er også Scrum Master sitt ansvar for å sørge for at Scrum Teamet utfører alle events i en Sprint. Scrum master må også prøve å fjerne hindringer for gruppens effektivitet (Schwaber & Sutherland, 2020). I dette prosjektet var gruppen såpass liten at Scrum Master, Dilan, også var en utvikler, og i praksis ledet Daily Scrum møter og Sprint Reviews.

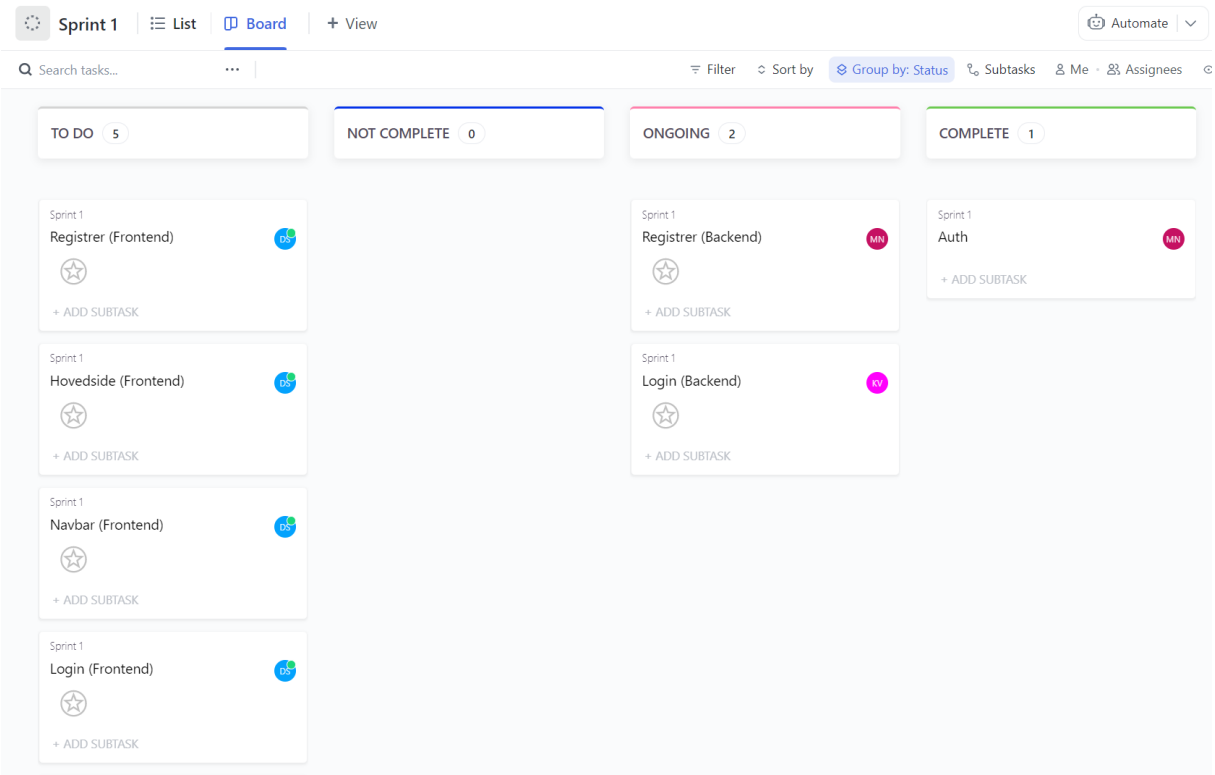
Utviklere er de som gjør arbeidet for å produsere resultater som får dem nærmere målet til Sprinten. De deltar i planleggingen og estimering av hver sprint, de skal sørge for at arbeidet gjøres med god nok kvalitet, og de må kontinuerlig gjøre endringer på planen dersom de blir nødt til det (Schwaber & Sutherland, 2020).

6.1.2 Sprint Planning

Sprint planning markeres som starten av en sprint og er et møte der Scrum teamet planlegger hva de skal få gjort i løpet av sprinten. Dette gjøres ved å plukke ut oppgaver fra Product backlog som legges i en Sprint backlog. Hver oppgave bør bli splittet inn i mindre biter som kan bli utført i løpet av en dag som dermed kan resultere i et Increment som kan bli

sett på som ferdig (Scrum.org, u.å.-b). Et Increment er utført arbeid i en Sprint sammen med det som ble laget i tidligere Sprinter. Det er viktig at et Increment faktisk kan bli brukt og at det funker med Increments fra tidligere Sprinter. Increments resulterer i at teamet har kommet nærmere målet til Sprinten (Scrum.org, u.å.-c).

Produkteier, altså bedriften Spar Åsane, har også hatt innspill på hva som skal planlegges for hver sprint. Gruppen spurte Spar Åsane etter hver Sprint Review om hva de ønsket skulle være ferdig innen neste Sprint, og hva de ville gruppen skulle prioritere. Informasjon om hva nøyaktig gruppen gjorde for hver sprint er i egne kapitler.



Figur 15 - Skjermbilde av ClickUp board fra Sprint 1

Figur 15 viser et skjermbilde av hvordan gruppen brukte Click-Up til å styre statusen på oppgaver via Click-Up sitt Board Display. I starten av hver sprint starter alle oppgaver som "To Do". Deretter, når en på gruppen velger en oppgave, så dras den inn i "On Going" som betyr at den er under arbeid. Når man er ferdig med en oppgave blir den lagt i "Complete". Dersom gruppen ikke rekker å bli ferdige med noen oppgaver etter fullført sprint, så blir de lagt i "Not Complete".

6.1.3 Daily Scrum

Daily Scrum er et møte mellom Scrum teamet, der teamet selv velger hvordan det skal fungere. Det er viktig at målet med møtet er å fokusere på å oppnå målet for hele Sprinten og at teamet kan få en plan for det neste de skal gjøre av arbeid. Dette er et daglig møte som skal ta max 15 minutter. Daily Scrums fører til bedre kommunikasjon mellom alle i teamet, hindringer kan bli satt lys på og det fører også til at teamet sparer tid fordi de får mindre behov for å ha flere møter (Scrum.org, u.å.-d). I dette prosjektet har gruppen en struktur på Daily Scrum som fungerer som følger: Hver deltaker forteller om hva de gjorde sist gang, om de har møtt på noen hindringer og hva det neste er de skal gjøre (DeClute, 2023).

6.1.4 Sprint Review

Sprint Review er et møte der Scrum teamet presenterer hva som har blitt gjort i sprinten og den generelle statusen til prosjektet til interessenter. Scrum Teamet vil deretter få tilbakemeldinger fra interessentene som er til stede. Til slutt blir det diskutert hva som skal bli gjort og prioritert i neste sprint. Dette er viktig informasjon som Scrum teamet vil ta med seg videre til neste Sprint planning (Scrum.org, u.å.-e).

6.1.5 Sprint Retrospectiv

Sprint Retrospective er et møte mellom Scrum teamet som tar sted etter fullført Sprint Review. Her diskuterer Scrum teamet hva de syntes gikk bra i sprinten, hva som gikk dårlig og hva som kan forbedres til neste sprint. På slutten av møtet, skal Scrum Teamet ha identifisert faktorer som kan forbedres til neste Sprint, slik at de kan produsere bedre kvalitet og øke effektiviteten for arbeidet (Scrum.org, u.å.-f).

6.1.6 Product Backlog

Product Backlog er en form for liste over oppgaver som skal gjøres, sortert etter hvor viktig hvert element er. Alle elementer på en Product Backlog skal være delt opp til oppgaver som skal kunne gjøres innen en enkelt sprint. Når oppgavene har blitt delt opp nok, kan de legges inn i en Sprint Backlog i Sprint Planning fasen. Når det gjelder hvor mye fra Product Backlog som tas med i en Sprint Backlog, så er det utviklerne som bestemmer (Schwaber & Sutherland, 2020). I dette prosjektet ble Product Backlog basert på systemkrav som Produkteier hadde kommet med og brukerhistoriene som ble utarbeidet tidlig i prosjektet.

6.2 Sprinter

6.4.1 Sprint 1

23. Februar - 2. Mars

Sprint Planning

I Sprint 1 begynte gruppen med utviklingen av prosjektet. Gruppen startet med å produsere en product backlog basert på brukerhistoriene med prioriteringer og gjorde estimater for hver task. Ut ifra product backloggen ble det bestemt at fokuset for Sprint 1 skulle være å implementere autentisering og autorisasjon. Den viktigste grunnen til denne beslutningen er at hele systemet er avhengig av en slik implementasjon siden det er et bruker-system. Dette innebærer at vi måtte lage innloggings- og registrerings-siden sammen med navigasjonsbaren. Det ble bestemt av gruppen at Sprint 1 bare skulle vare 1 uke fordi gruppen estimerte at vi kom til å rekke å systemet for implementere autentisering og autorisasjon innen da.

For tidsestimering av de forskjellige oppgavene, bestemte gruppen at vi skulle benytte oss av poeng istedenfor tid. Gruppen mente at dette kom til å gjøre estimeringsprosessen lettere fordi det er vanskelig å estimere programmeringsoppgaver når man ikke har så mye erfaring med det og estimering med poeng er lettere fordi da trenger man bare å estimere oppgavene relativt til de andre. Click-Up hadde også et innebygd estimering system for poeng, så det gjorde jobben lettere for gruppen.

Sprint Review

Første Sprint Review ble holdt sammen med produkteier til stede. Møtet startet med at gruppen viste fram Product backlog, prioriteringer og estimater. Produkteier var enig i Product backlog vi hadde laget og prioriteringene som var satt. Deretter viste gruppen systemet vi hadde implementert i Sprint 1 for autorisasjon og autorisering. Produkteier fikk dermed sett funksjonalitetene for registrering og innlogging. Etter det fikk vi tilbakemeldinger fra produkteier om implementasjonen hvor produkteier blant annet kommenterte på Spar logoen gruppen hadde brukt og at den bør inkludere “Åsane”. Helt til sist diskuterte gruppen og produkteier om hva som skulle bli gjort i neste Sprint.

Sprint Retrospective

Etter Sprint Review satte gruppen seg ned for å utføre Sprint Retrospective. Her diskuterte gruppen de 3 hovedspørsmålene som stilles i et slikt møte. Når det gjaldt hva som gikk bra i Sprinten, så var gruppen enige at vi hadde gjort en god jobb siden vi rakk å bli ferdige tidligere enn forventet med det som var planlagt i sprinten og kunne derfor legge til 2 oppgaver til som var navigasjonsbar og hovedside som også ble fullført. Gruppen fikk til mye parprogrammering hvor gruppen kunne hjelpe hverandre. Gruppen var flinke til å holde Daily Scrum konsekvent og kort gjennom sprinten.

Gruppen diskuterte også Github og hvordan det førte til en del problemer når det gjaldt repository og pushing av kode. Dette førte til en del debugging som spiste mye tid. Noen oppgaver ble også fullført senere enn det som var estimert. Grunnen til dette var fordi gruppen hadde gjort estimater på hver oppgave som ikke var basert på programmeringskunnskapene til personen som hadde ansvaret for oppgaven. En annen ting gruppen diskuterte var hvordan mye tid ble brukt i starten av hver dag for å finne ledig rom på UiA, fordi gruppen noen ganger glemte å holde av rom tidlig.

For å prøve å fikse disse feilene, bestemte gruppen at de i fremtidige sprinter alltid skal spørre om noen trenger hjelp med noe og har lyst til å programmere før vi legger til flere oppgaver i sprinten. Gruppen valgte også ut 1 person som skulle holde av rom på UiA 1 uke på forhånd for å unngå problemer med å ikke ha noe rom.

6.4.2 Sprint 2

6. Mars - 16. Mars

Sprint Planning

Sammen med produkteier, ble det bestemt at Sprint 2 skulle fokusere på funksjonalitet for å finne produkter og legge til handlekurv og deretter sende ordre. Dette innebar at gruppen skulle utvikle sidene for produktinfo, katalog og handlekurv. Disse sidene representerer det viktigste løpet for sluttbrukerne og hadde derfor høy prioritasjon. På toppen av det skulle gruppen også håndtere validering av data fra API requests. Dette er for å passe på at det som blir sendt til serveren er som forventet og ikke inneholder noe ondsinnet. Gruppen fortsatte å bruke poeng for estimering og var mer komfortable denne gangen etter å ha fått mer erfaring med programmering.

Sprint Review

Andre Sprint Review ble holdt sammen med produkteier og gruppen sin veileder til stede. Møtet startet med at gruppen viste frem det vi rakk å gjøre i sprinten, hvor lang tid vi brukte på oppgavene og om det var innenfor estimatene. Gruppen viste frem katalog, produktinfo og handlekurv siden. Gruppen rakk ikke å gjøre ferdig designet for handlekurv på mobil sammen med noen funksjonaliteter i backend og derfor ble det forklart til produkteier at oppgaven vil fortsette i Sprint 3. Etter det ga produkteier og veileder tilbakemeldinger. Gruppen fikk gode tilbakemeldinger, men også noen ting som måtte legges til som for eksempel at produkteier ville ha en pop-up når man trykker på knappen for å legge til produkt i handlekurv, hvor man da får alternativer for å legge til handlekurv som 1. prioritet eller som alternativ vare (2. prioritet). En annen viktig ting var at produkteier ville ha registreringssystem som gir engangspassord til sluttbrukerne fremfor at Spar Åsane selv lager passordet for dem. Helt til sist diskuterte gruppen og produkteier om hva som skulle bli gjort i neste Sprint.

Sprint Retrospective

Andre Sprint Retrospective diskuterte gruppen igjen de 3 spørsmålene. Gruppen hadde diskusjoner rundt parr programmering og hvordan vi hadde forbedret prioritasjonen av det. Daily Scrum gikk også like bra som sist.

Under Sprint Review 2 støttet gruppen på noen problemer da vi skulle vise fram det som hadde blitt gjort i webapplikasjonen som gjaldt en funksjon. Dette var fordi gruppen ikke hadde testet ut denne spesifikke funksjonaliteten før møtet. Gruppen ble derfor enige om at til neste Sprint Review så måtte vi teste absolutt all funksjonalitet i applikasjonen på forhånd. En annen ting som gruppen gjorde feil under Sprint Review, var at vi brukte tekniske begreper som produkteier ikke forstod. Derfor må gruppen til neste Sprint Review bruke mer forståelige ord som en person uten teknisk bakgrunn kan forstå.

Estimering av tid var noe gruppen syntes gikk litt dårlig. Å bruke poeng for estimering var en del forvirrende for gruppen og vi visste ikke helt hvordan vi kunne forklare estimeringen til produkteier om hvor mye tid som var estimert for en oppgave. Derfor bestemte gruppen at til neste Sprint skulle vi estimere tid istedenfor. Gruppen følte seg trygge med å gjøre dette nå som vi hadde fått en del erfaring med programmering etter 2 sprinter. Ved å estimere tid kan gruppen lettere kommunisere estimeringene som ble gjort til produkteier.

6.4.3 Sprint 3

17. Mars - 30. Mars

Sprint Planning

Sammen med produkteier, ble det bestemt at Sprint 3 skulle fokusere på funksjonalitet for at admin skal kunne se aktive ordre og lukke dem og for kundene til å se brukerinfo og ordrehistorikk sammen med en innboks der de kan se meldinger om sendte ordre. Dette innebar at gruppen måtte utvikle sidene for blant annet profil, innboks og ordreliste. Disse sidene representerer de resterende funksjonalitetene for den viktigste delen av kundereisen. På toppen av dette må også handlekurven siden, som ikke ble fullført forrige sprint, bli gjort ferdig. Som nevnt i forrige Sprint Retrospective, så begynner gruppen å estimere ut ifra tid fremfor poeng.

Sprint Review

Tredje Sprint Review ble holdt sammen med produkteier til stede. Møtet startet med at gruppen viste frem at vi rakk å bli ferdig med alt som var planlagt, altså profil, innboks og ordreliste sidene. Estimeringene av alle oppgavene ble også vist til produkteier og var veldig nær den tiden som faktisk ble brukt. Deretter fikk gruppen tilbakemeldinger, både gode og ting som bør ha blitt gjort annerledes. Blant annet ville produkteier at meldingene i innboksen skulle være i tabellformat slik at det er lettere å forstå. Helt til sist diskuterte gruppen og produkteier om hva som skulle bli gjort i neste Sprint.

Sprint Retrospective

Tredje Sprint Retrospective diskuterte gruppen igjen de 3 spørsmålene. Gruppen var alle enige om at overgangen fra estimeringen via poeng til tid var det riktige valget. Gruppen gjorde en god jobb med estimeringen, mange av oppgavene ble fullført veldig nær det som var estimert. Det ble også lettere å kommunisere det til produkteier i Sprint Review og det ble mye mer forståelig.

Planleggingen gikk også bedre denne gangen, etter som at vi rakk å fullføre alt som var planlagt. Vi møtte ikke på så mange Github problemer nå som vi hadde fått mer erfaring med det. Vi var også godt forberedt på Sprint Review og passet på at all funksjonaliteten funket på forhånd. Og som vanlig fikk vi til gode Daily Scrum og parprogrammering. Noe som gikk dårlig denne sprinten var at en funksjonalitet måtte bli skrevet om igjen flere ganger. Dette førte da til en del bortkastet tid. Gruppen skal prøve å fikse dette problemet i neste sprint ved å planlegge hvordan vi skal utvikle funksjonalitetene mer nøye.

6.4.4 Sprint 4

11. April - 20. April

Sprint Planning

Sammen med produkteier, ble det bestemt at Sprint 4, som var siste sprint, skulle fokusere på de siste funksjonalitetene i webapplikasjonen som er produktliste og produktinnstillinger for admin. På toppen av det skal også produkteier sine tilbakemeldinger fra tidligere sprinter bli implementert som for eksempel førstegangspassord funksjonalitet og meldinger i tabellformat. Dette var alt som manglet for at webapplikasjonen skulle bli ferdig.

Sprint Review

Fjerde og siste Sprint Review ble holdt sammen med produkteier og veileder til stede. Her viste gruppen frem det som ble gjort i sprinten og hvor lang tid det tok i forhold til estimatene som ble gjort. Gruppen forklarte at vi måtte gjøre en beslutning på å gå vekk fra prosjektinnstillinger funksjonaliteten fordi det ikke hadde funket med databasen som regelmessig oppdaterer seg automatisk. Produkteier sa at dette var greit og var enig at det ville bli problematisk. Gruppen viste deretter frem hele applikasjonen slik at produkteier og veileder kunne få se resultatet av prosjektet. Gruppen fikk veldig gode tilbakemeldinger og alle parter var fornøyde med resultatet. Etter det mottok gruppen tilbakemeldinger om noen små ting som kunne endres som for eksempel tekst som heldigvis er lett å endre på.

Sprint Retrospective

Fjerde og siste Sprint Retrospective diskuterte gruppen igjen de 3 spørsmålene. Her var det mye diskusjoner om hva som hadde gått bra og hvor mye som var forbedret siden Sprint 1. I Sprint 4 fikk gruppen veldig sjeldent problemer med Github, vi rakk å bli ferdige med alt i sprinten, Daily Scrum og parprogrammering funket fortsatt ganske bra, gruppen diskuterte funksjoner mer nøye sammen og de fleste estimeringene ble møtt. Det eneste gruppen ikke var fornøyde med var at vi måtte fjerne siden for produktinnstillinger midt i sprinten. Heldigvis, så hadde ikke gruppen brukt mer enn 2 timer på det, så det var ikke mye tid som ble kastet bort. Men gruppen skulle fortsatt ønske at vi hadde tenkt mer nøye over denne funksjonaliteten.

7 - Refleksjon og Konklusjon

I dette kapittelet vil gruppen konkludere rapporten og deretter reflektere over hvordan prosjektet har gått.

7.1 Konklusjon

Spar Åsane ville at gruppen skulle utvikle et bestillingssystem for dems private kunder som også sørget for at de ikke stod tomhendte dersom det var tomt i butikk hvor de også kunne fått beskjed på forhånd om hva som blir sendt til dem. Gruppen utviklet dermed en web-applikasjon der man kan legge til produkter i en handlekurv via en katalog og dermed sende ordre. Når Spar Åsane sender og lukker ordren, vil kundene få en melding som utlyser alle produktene som blir sendt. Web-applikasjonen tilfredsstiller de viktigste kravene fra Spar Åsane. Dette inkluderer alle kravene som er kategorisert som “Must-Have” og de fleste som er kategorisert som “Should-Have”.

Vi skulle gjerne hatt mer tid til å gjøre ferdig resten av det som var kategorisert som “Should Have”, men dessverre så hadde vi ikke det. Gruppen fikk lært utrolig mye gjennom prosjektet og vi er glade for at vi fikk muligheten til å utvikle et system som både de ansatte på Spar Åsane og de private kundene kan få stor nytte av.

7.2 Refleksjon

Gjennom arbeidet med dette prosjektet har gruppen hatt en veldig komfortabel reise uten mye stress. Gruppen er svært stolte av utføringen av prosjektet og føler at vi fikk benyttet oss godt av alt det vi har lært gjennom studiet. Mye av arbeider gruppen gjorde gjennom semesteret hadde mange likheter fra det vi gjorde i 3. semester med fagene systemanalyse og systemutvikling, programmeringsprosjekt, og datamodellering og databasesystemer. Dette var veldig positivt ettersom gruppen hadde alle gode minner fra fagene i det semesteret. Prosjektet ga gruppen god innsikt i hvordan det er å jobbe med en ekte kunde der man leverer et produkt til en bedrift. Dette er god erfaring som alle kan ta med videre i karrierene sine.

Som en gruppe jobbet vi veldig tett sammen, hver dag, gjennom hele semesteret. Gruppen lærte utrolig mye om hvordan det er å programmere sammen, noe som tok litt tid å bli vandt til. Å programmere alene var vi alle erfarne med, men å programmere sammen og kunne lese hverandres kode var en stor utfordring, men det var noe vi ble bedre på gjennom hele prosjektet. Selv om vi ble bedre på programmering gjennom prosjektet burde vi ha brukt mer

tid på å lære programmering i kompetanseutvikling fasen av prosjektet. Hvis gruppen hadde hatt mer kunnskap i begynnelsen av utviklingen, kunne vi unngått noen tidlige feil.

En annen utfordring ved prosjektet var at siden Spar Åsane er en dagligvarebutikk, så hadde de ikke noen som kunne hjelpe oss teknisk med alle teknologiene som var involvert der mye av det var nytt for gruppen. Dette tvang oss som en gruppe til å bli mer avhengige av oss selv og være selvstendige når vi har tatt beslutninger.

Et annet problem som oppsto på grunn av at de er en dagligvarebutikk var dårligere kommunikasjon. Ansatte på Spar Åsane forsto ikke helt tekniske begreper som gruppen brukte da vi snakket til dem i møter. Dette kunne vi ha gjort bedre i begynnelsen. Etterhvert i prosjektet endret gruppen hvordan vi snakket om systemet til dem, ved å bruke mer gjenkjennelige begreper og ved å snakke om mindre kompliserte konsepter. Som utviklere, var det vårt ansvar å formidle informasjon til produkteier på en forståelig måte.

Litteraturliste:

Aven, T. (2023, 25. januar). *risikoanalyse.* I Store norske leksikon på snl.no. Hentet 4. mai 2023 fra https://snl.no/risikoanalyse

Benyon, D. (2014). *Designing Interactive Systems: A comprehensive guide to HCI, UX and interaction design*. (3. utg.). Harlow: Pearson Education Limited

Batista, J. (2023, 3. april). *The Pros and Cons of CSS Frameworks: A Comprehensive Review*. Dev Community. https://dev.to/c0mmand3rj/the-pros-and-cons-of-css-frameworks-a-comprehensive-review-13db

DeClute, D. (2023, 24. mars). *The 3 daily Scrum questions*. The Server Side. https://www.theserverside.com/tip/The-3-daily-Scrum-questions

GeeksforGeeks. (2023, 19. april). *Express.js*. https://www.geeksforgeeks.org/express-js/

GeeksforGeeks. (2019, 7. november). *Unified Modeling Language (UML)* | *State Diagrams.* https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/

Hamilton, T. (2023, 8. april). *What is Test Driven Development (TDD)? Example.* Guru99. https://www.guru99.com/test-driven-development.html

Jaye Hannah. (17, April. 2023). *What's the Difference Between a Wireframe, a Prototype, and a Mockup?* Careerfoundry. https://careerfoundry.com/en/blog/ux-design/difference-between-wireframes-prototypes-mockups/

JetBrains. (2023, 24. mars). *Getting started with WebStorm.* https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html

Mathiassen, L., Munk-Madsen, A., Nielsen, P.A & Stage. J. (2018). *Object oriented analysis and design.* (2. utg.) Hadsund: Metodica ApS.

Microsoft. (23, April. 2023). *Typescript for the New Programmer.* Typescriptlang. Hentet 28. april 2023 fra https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

Meta Open Source. (u.å). *The library for web and native user interfaces*. React. Hentet 28. april 2023 fra https://react.dev/

Max Rehkopf. (u.å). *User stories with examples and a template*. Atlassian. Hentet 28. april 2023 fra https://www.atlassian.com/agile/project-management/user-stories

Mahmoud, M. (2021, 12. oktober). *An Overview of Unit, Integration, and E2E Testing.* Modus Create. https://moduscreate.com/blog/an-overview-of-unit-integration-and-e2e-testing/

Lotz, M. (2018, 5. juli). *Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?.* Segue Technologies. https://www.seguetech.com/waterfall-vs-agile-methodology/

Oracle. (u.å). *What is MySQL?*Hentet 28. april 2023 fra https://www.oracle.com/mysql/what-is-mysql/

Obaseki, N. (2023, 28. mars) *localStorage in Javascript: A complete guide.* Log Rocket. https://blog.logrocket.com/localstorage-javascript-complete-guide/#advantages-disadvantages-localstorage

Peterson, R. (2023, 4. mars). *Entity Relationship (ER) Diagram Model with DBMS Example.* Guru99. Hentet 24. april 2023 fra https://www.guru99.com/er-diagram-tutorial-dbms.html

Postman. (u.å). *What is Postman?* Hentet 2. mai 2023 fra https://www.postman.com/product/what-is-postman/

Powell, K. (2020, 7. april). *How to Take the Right Approach to Responsive Web Design.* freeCodeCamp. https://www.freecodecamp.org/news/taking-the-right-approach-to-responsive-web-design/

Schwaber, K. & Sutherland, J. (2020). *The 2020 Scrum Guide.* https://scrumguides.org/scrum-guide.html

Schneider, D. (2021, 23. juli). *Choosing a Third-party Library.* Medium. https://dana-scheider.medium.com/choosing-a-third-party-library-e8b0f7aa9497

Scrum.org. (u.å.-a). *What is Scrum?*Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-scrum-module

Scrum.org. (u.å.-b). *What is Sprint Planning*?Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-is-sprint-planning

Scrum.org. (u.å.-c). *What is an Increment*?Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-is-an-increment

Scrum.org. (u.å.-d). *What is a Daily Scrum*?Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-is-a-daily-scrum

Scrum.org. (u.å.-e). *What is a Sprint Review?*Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-is-a-sprint-review

Scrum.org. (u.å.-f). *What is a Sprint Retrospective?*Hentet 4. mai 2023 fra https://www.scrum.org/resources/what-is-a-sprint-retrospective

Semah, B. (2022, 5. desember). *What exactly is Node.js? Explained for Beginners.* freeCodeCamp. https://www.freecodecamp.org/news/what-is-node-js/

Vedlegg

Vedlegg 1 - Uttalelse fra oppdragsgiver

Ble kontaktet fra en gruppe fra UIA om det var mulig å gjøre et prosjekt for Spar Åsane. Det hørtes interessant ut, så vi luftet litt ideer på hva vi kunne få til og hva vi kunne trenge på våres butikk. Har en del leveranser av matvarer til bedrifter og private som bestiller mail og tlf i dag. Spar har en egen side som kunder kan bestille matvarer på, men denne passer ikke for oss. Ble enige med gruppa om å lage en egen app eller bestillingsside som bare var for vår butikk og tilpasset våre behov. Den måtte være enkel i bruk for både kunder og butikk. Gruppa kom med forslag til hvordan dette kunne løses og sammen med innspill fra bedrift ble dette en side som passer bra til våres behov.

Har hatt kontinuerlige møter der fremdrift ble presentert og om det evt var justeringer som måtte tas hensyn til. Oppfatter at gruppa har løst våre behov på en god måte og er svært tilfreds med resultatet. Har ikke testet det ut på våre kunder ennå, men blir presentert så snart prosjektet er helt ferdig. Regner med at dette blir godt mottatt av våre kunder og forenkler deres bestillinger. Ser matvareutvalget bedre, priser på produkter og oversiktlig side.

- Arild Nordbø. Spar Åsane. 14.05.23.

Vedlegg 2 - Egenvurdering

Dilan

I dette prosjektet har jeg hatt rollen som Scrum Master der jeg har ledet både Daily Scrums og Sprint Reviews og sørget for at gruppen alltid har et sted å jobbe fysisk. Jeg har hatt ansvar for Frontend delen av web-applikasjonen og lært masse om React og Typescript. Jeg trivdes veldig godt i denne rollen, både som Scrum Master og Frontend utvikler hvor jeg har fått mye relevant erfaring som jeg kan ta med videre i karrieren min.

Marius

I dette prosjektet har jeg jobbet sammen med Knut Erik på Backend. Jeg har også hatt rollen som Git Master, der jeg assisterte de andre på gruppen med versjonskontroll og var den som sammenslått pull requests til main branch. Jeg har lært mye om Javascript og fått brukt min kunnskap om SQL og databaser i dette prosjektet. Jeg har jobbet sammen med gruppen og lært hvordan man utviklet funksjonene vi trengte for vårt system, og hvordan teste sider med eget programvare

Knut Erik

I dette prosjektet har jeg jobbet med Marius på backend. Jeg har ikke hatt noen andre roller, men har som regel fått mindre oppgaver som input validation, rate limiter og liknende. Jeg har også laget noen av sidene på egen hånd, og noen ved hjelp fra Marius. Gjennom prosjektet har jeg lært mye om utvikling generelt, og har fått erfaring innen Javascript og SQL. Jeg har lært om filstruktur og arkitektur, og sikkerhetstiltak som må tas hensyn til når man skal lage en web app.

Vedlegg 3 - Systemkrav

Systemkrav	Prioritet
Systemet må inneholde ryddig kode slik at systemet blir lett å vedlikeholde	Should Have
Systemet må ha en søkefunksjon for produkter så kunder kan enkelt finne det de vil ha	Should Have
Systemet må la kunder velge dato for når ordren skal bli send til dem for kunder mer fleksibilitet	Could Have
Systemet må la kunder endre passord slik at det går bra dersom de glemmer passordet sitt	Could Have

Vedlegg 4 - Brukerhistorier

ID	Brukerhistorie	Kriterier	Argument	Prioritet
5	Som administrator hos Spar Åsane ønsker jeg å analysere statistikk om bestillingene, så jeg kan bli mer informert.	Applikasjonen kan ha generell informasjon om hvor mange aktive bestillinger som finnes, hvilke bedrifter som har bestilt, hvor ofte de bestiller, gjennomsnittlig statistikk	Denne funksjonen er Should Have fordi den er ikke kritisk til applikasjonens generelle funksjoner, men er nyttig å implementere for administratorene i Spar Åsane.	Should Have
6	Som kunde vil jeg komme med forslag til alternativ dersom varen jeg ønsker ikke er tilgjengelig, så det er større sjanse for at jeg får noe.	Dersom varen ikke er tilgjengelig så får kunden varsel om dette og mulighet til å velge alternativ.	Denne funksjonen er Should Have fordi det er en viktig del av forbedringen av den fysiske prosessen, men ikke nødvendig for	Should Have

			systemets grunnlag.	
7	Som admin så ønsker jeg en oversikt over kundene som bruker applikasjonen, slik at jeg har lett tilgang til kontakt med disse personene.	Applikasjonen skal ha en side for admins der man kan få en tabell med kunder og kundeinformasjon.	Denne funksjonen prioriteres som “Should Have” fordi det er nyttig for admins å kunne ha kontakt informasjon om brukere slik at de kan kontaktes for hva enn det måtte være. Det er ikke “Must Have”, fordi applikasjonen vil kunne fungere helt fint uten.	Should Have
8	Som eier av produktet vil jeg ha en applikasjon som er effektiv og lett å vedlikeholde, så det blir mindre jobb.	En applikasjon som inneholder ryddig kode hvor programmererne har fokusert på ytelse og lagt opp for videre vedlikehold. Koden har fulgt kodestandarder og brukt “Best practices”.	Dette er prioritert som “Should Have” og ikke “Must Have” fordi selv om applikasjonen ikke er effektiv og har kode som ikke er ryddig, så vil det fortsatt kunne fungere. Men det er fortsatt viktig for at systemet har høy responstid og er lett å vedlikeholde.	Should Have
9	Som privatkunde vil jeg ha en applikasjon som er brukervennlig, slik at jeg kan lett og effektivt utføre handlinger.	Når brukeren er inne på applikasjonen, så er det lett å forstå hvordan ting gjøres og handlingene blir også lette å utføre.	Dette er “Should Have” fordi det er viktig at nettsiden er universelt utformet slik at flest type personer som mulig kan bruke applikasjonen, men det er ikke “Must Have”, fordi applikasjonen vil fortsatt kunne fungere uten.	Should Have
10	Som privatkunde vil jeg kunne søke etter varer ut ifra navn på varene for å enkelt finne fram.	Brukeren har tilgang til et søkefelt i varekatalogen der brukeren kan skrive inn navn til varer og trykke på en knapp. Deretter vil katalogen oppdatere seg og bare hvis varer som inneholder strengen med bokstaver som er skrevet i søkeboksen.	Denne funksjonen prioriteres som “Should Have” fordi den gjør applikasjonen mer brukervennlig og mer effektivt, men applikasjonen vil fortsatt kunne fungere uten den, så derfor er det ikke “Must Have”	Should have
11	Som privatkunde ønsker jeg	Brukeren har tilgang til en	Dette er “Could	Could

	å kunne velge dato for når bestillingen av varer blir sendt, så jeg har mer fleksibilitet	funksjon som gjør det mulig å velge en dato hvor da bestillingen av varer blir sendt denne dagen.	Have", fordi det gjør det mulig for brukere å planlegge bestillinger fremover i tid, men det er ikke "Should Have" fordi det er ikke en viktig funksjon som de fleste kommer til å ha nytte av og applikasjonen er ikke avhengig av den i det hele tatt.	Have
12	Som privat kunde ønsker jeg å kunne bytte passord, så det ikke er problematisk om jeg glemmer passordet mitt.	Kunden kan ha tilgang til en funksjon på innloggingssiden som sender en forespørsel om passordendring til email som er knyttet til brukeren.	Dette er "Could Have". Endring av passord kan blir gjort manuelt gjennom at kunden sender mail til Spar Åsane og ber om endring gjennom administratorer, istedenfor en egen automatisk prosess.	Could Have

Vedlegg 5 - Risikoanalyse

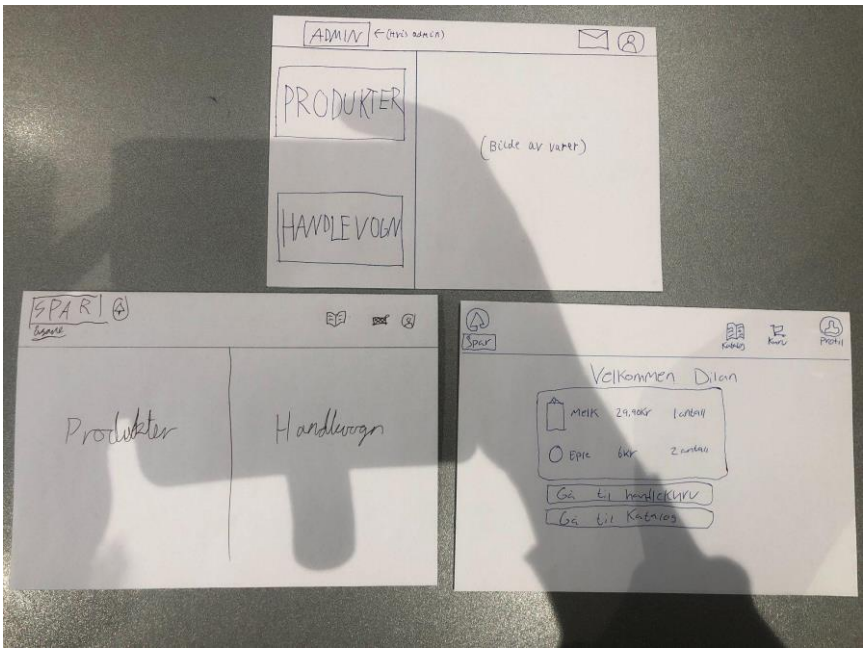
Risiko	**Konsekvens**	**Sannsynlighet**	**RISIKO**	**Tiltak for at ikke skjer**	**Tiltak for å redusere skade**
TEKNISK	—----------	—----------	—---------	—----------	—----------
Internettproblemer	2	2	4	Arbeid i et sted med godt internett	Mobildata. Dra til et sted med internett. Verktøyene kan brukes offline.
Maskinvare- problemer	4	1	4	Ha fungerende maskinvare.	Lagre backups av viktige filer.
Manglende ressurser til digitale verktøy	4	1	4	Bruk verktøy som har god støtte.	Bytte verktøy så effektivt som mulig.
Menneskelig feil i programvare	2	4	8	Kommunisere i gruppen. Se over innstillinger.	Søke opp feil og undersøk problemer.
Miste filer	4	2	8	Lagre backups av viktige filer. Bruke Github og Google Docs	Husk innholdet til filen, prøv å gjenskape den.
Miste tilgang til skylagring	4	1	4	Lagre lokale backups av filer. Bruk pålitlige skylagrings verktøy	Hent backup av filen. Bytt skylagrings verktøy.
Programvarefeil	3	1	3	Bruk pålitelig programvare. Bruk riktig versjon av programvare.	Undersøk løsninger. Potensielt bytte programvare.

Programvareversjon forskjeller	2	3	6	Bli enige om versjon som skal brukes. Oppdater programvare.	Bytt versjon av programvare.
Programmeringsfeil	2	3	6	Ha god kunnskap om programmeringsspråket. Spørre andre om hjelp.	Debug feilen. Undersøk løsninger. Jobb rundt feilen.
Mangler i database	3	2	6	Planlegg databasemodellen godt. Se over felt i databasen.	Fiks manglene i databasen.
Problemer med Merging	2	3	6	Bruk github riktig. Bruk Branches. Pull endringer regelmessig.	Jobb sammen for å løse konflikten.
KOMMUNIKASJON	—----------	—---------	—---------	—----------	—----------
Møte opp for sent	1	4	4	Planlegg godt. Sett av mer tid enn forventet.	Jobb effektivt uten personen som ikke har møtt.
Manglende kommunikasjon med Spar Åsane	4	2	8	Planlegg kommunikasjon. Opprett godt forhold.	Fortsett arbeid under tidligere planlagte mål. Jobb med andre deler av prosjektet.
Dårlig planlegging til møter	4	2	8	Planlegg møter i god tid før.	Gjør det beste ut av møter man kan.
Uforventet endring i produktplan	3	2	6	Planlegg mulige endringer og retninger prosjektet kan gå i.	Finn beste alternative løsning effektivt.
Stor uenighet i gruppen	4	2	8	Skap enighet om plan for prosjekt. Kommunisere klart og tydelig.	Snakk sammen og løs problemet.
Konflikt i gruppen	5	1	5	Skap et hyggelig miljø. Vær enige om mål.	Snakk sammen og løs problemet. Jobb videre.
Distraksjoner	1	5	5	Vær fokusert. Følg planen for hver dag. Planlegge pauser.	Jobbe effektivt før distraksjon.
Nedsatt motivasjon	3	3	9	Planlegge pauser. Balansere privat og arbeid	Kommuniser i gruppen. Bytt oppgaver med andre. (SAMM metoden?)
Sykdom	3	3	9	God hygiene	Jobbe hjemmefra. Andre i gruppen fortsetter.
Mentalt utmattet	4	3	12	God døgnrytme. Planlegge pauser. Balansere privat og arbeid	Kommunikasjon og samarbeid. Pause
PROSJEKT	—----------	—---------	—---------	—----------	—----------
Feature creep	3	2	6	MoSCoW prioritering.	Fjern de mest unødvendige ideene.

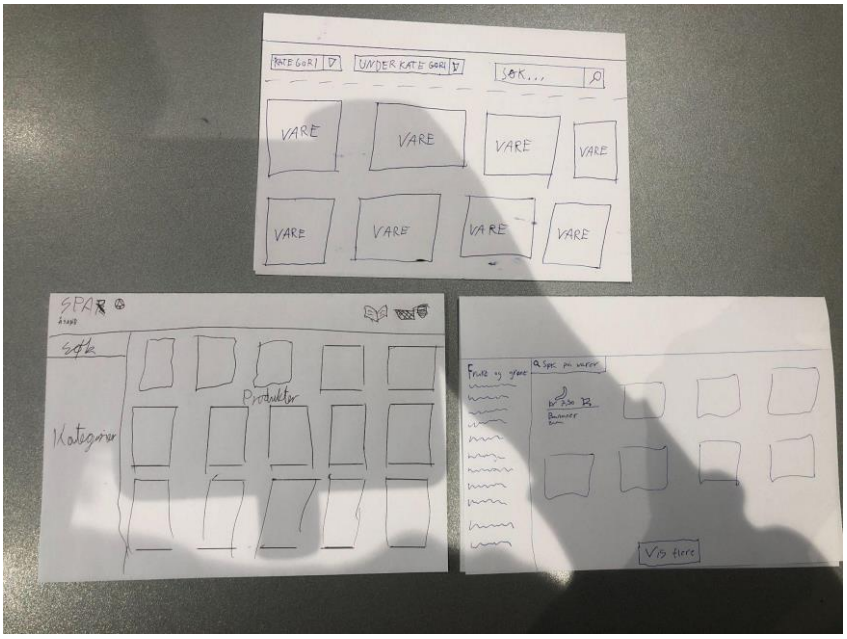
(Forsinkelse i plan)	4	3	12	God planlegging, gjør det viktigste først etter MoSCoW prioritering.	Gjør endringer i planen.
Arbeid blir ikke gjort	5	2	10	God kommunikasjon, scrum under programmering	Finn hovedårsaken, løs problemet. Gjøre arbeidet i plenum
Gruppe medlemmer er redde for å ta opp noe viktig	4	1	4	Skap et hyggelig miljø. Bruk effektivt SCRUM.	Åpne for samtale. Snakk om problemet privat.
Mangel på kritiske funksjoner i sluttprodukt	5	2	10	Bruk MoSCoW prioriteringer. Hold god kommunikasjon og bruk SCRUM.	Skade i sluttprodukt kan ikke reduseres.
Dårlig user experience	3	2	6	Kommuniser med Spar Åsane. Planlegg design nøye.	Gå tilbake til design planlegging, lag et nytt design.
Følger ikke programmering-standarder	2	2	4	Følg programmering-standarder. Planlegg standarder.	Fiks koden. Følg programmering-standarder
Dårlig fordeling av arbeidsoppgaver	3	3	9	Skap enighet og rettferdighet når man fordeler arbeidsoppgaver.	Bytt oppgaver og del arbeidet rettferdig.

Vedlegg 6 – Skisser

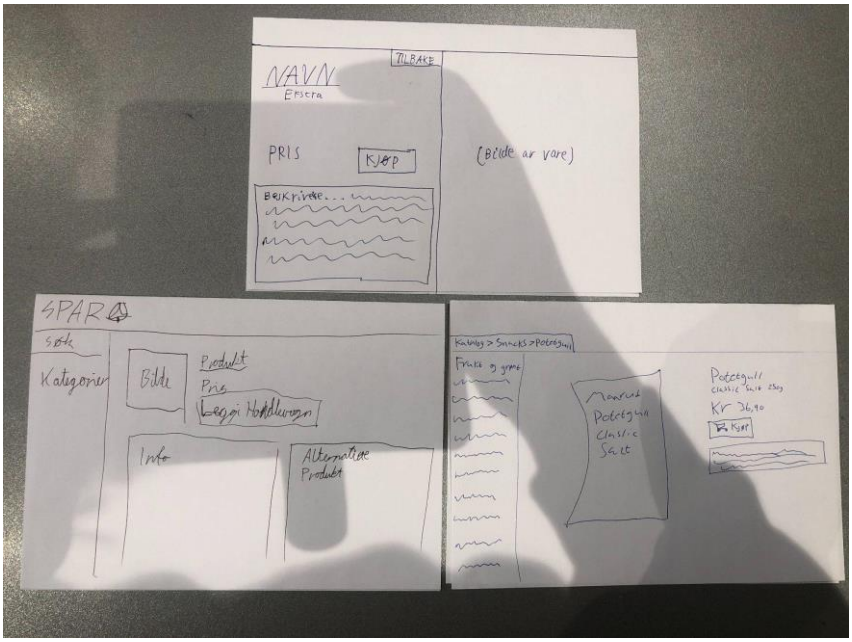
Vedlegg 6.1 - Forside



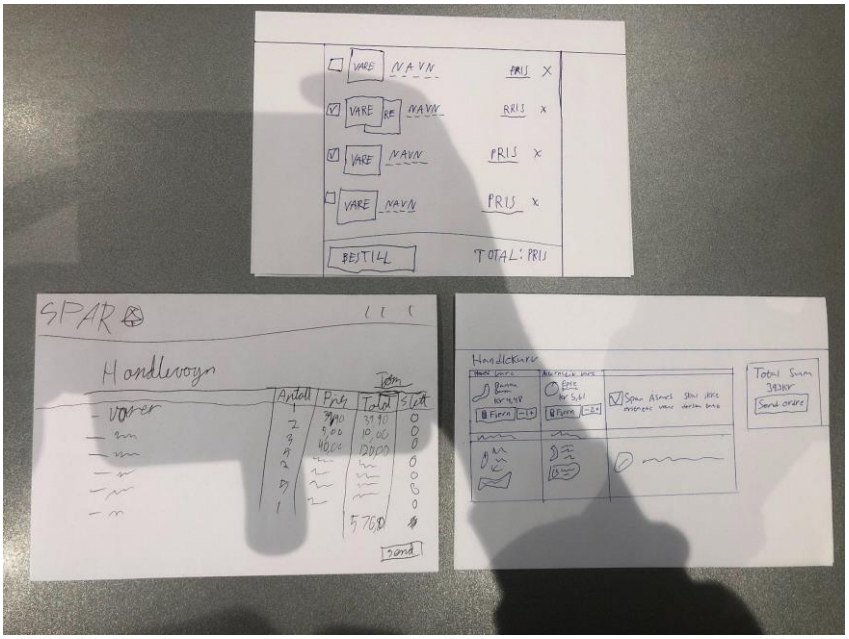
Vedlegg 6.2 - Katalog



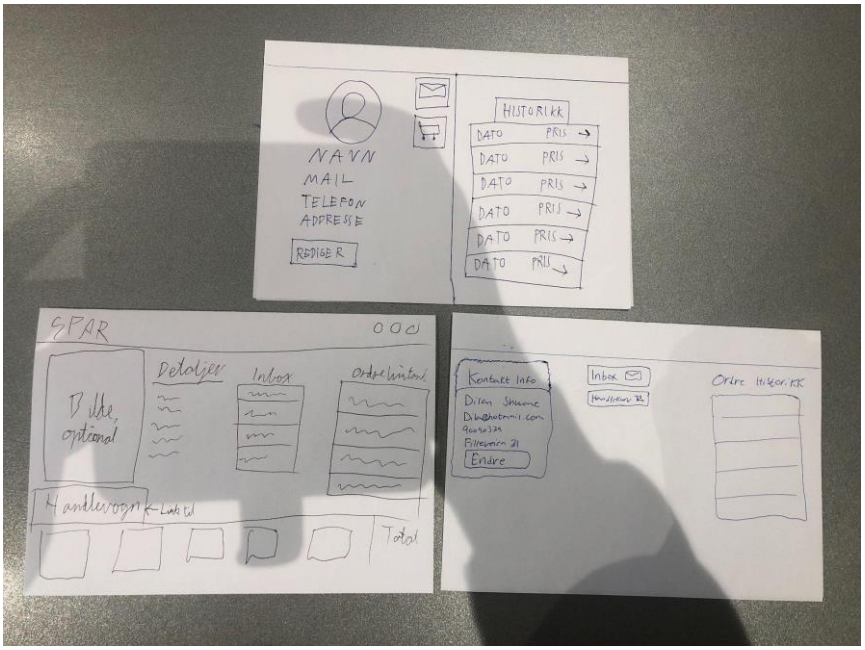
Vedlegg 6.3 - Produktinfo



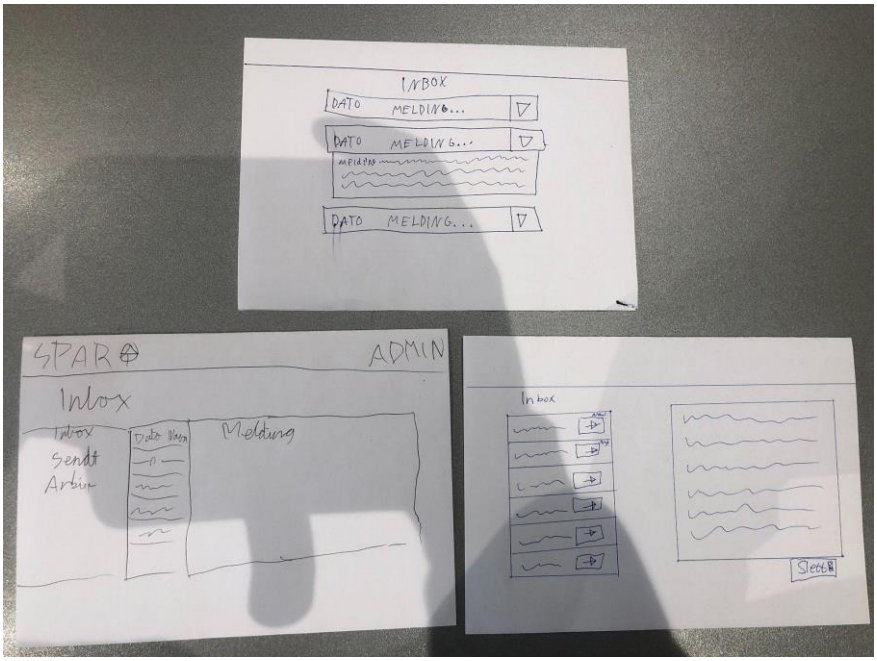
Vedlegg 6.4 - Handlevogn



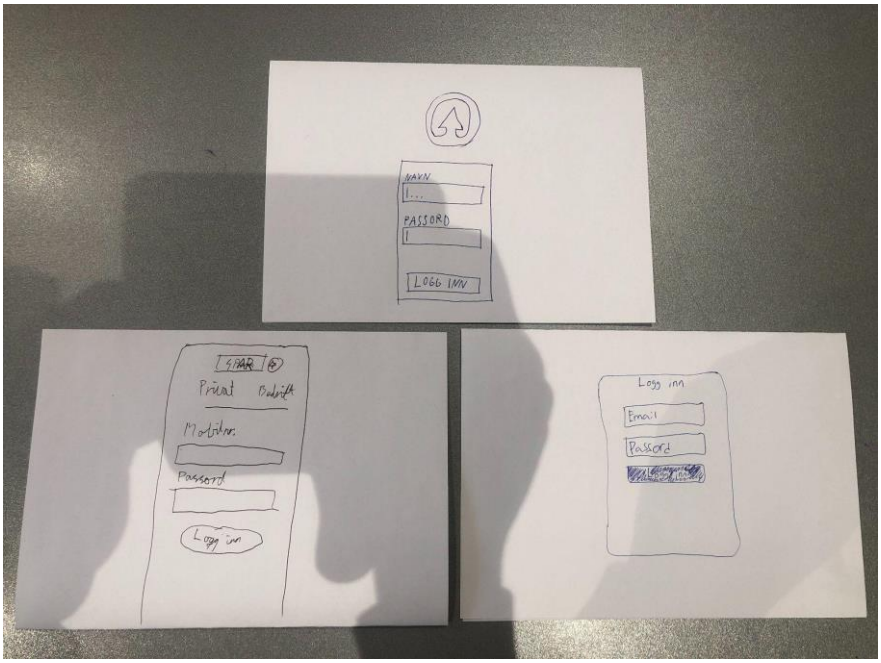
Vedlegg 6.5 - profil



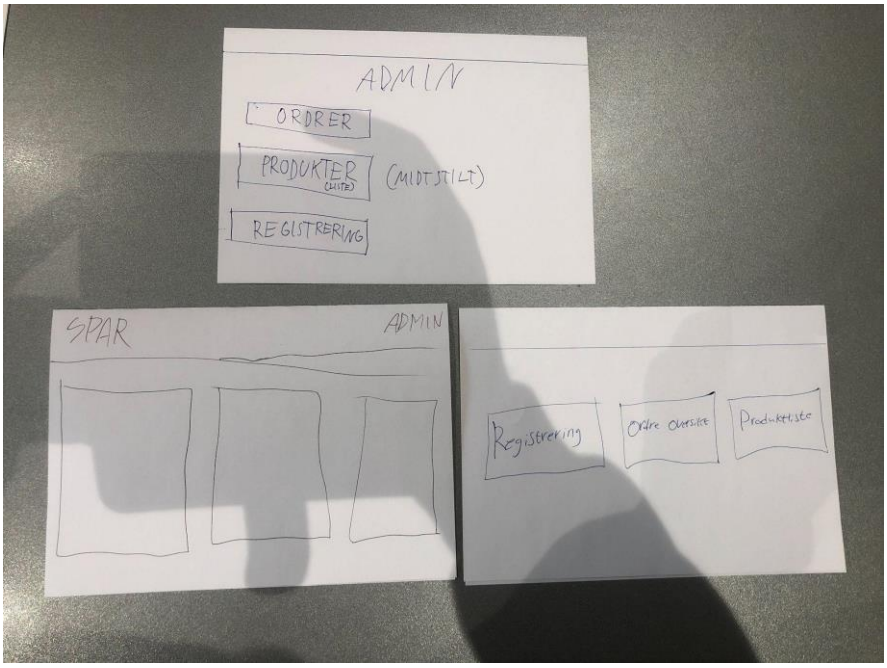
Vedlegg 6.6 - Innboks



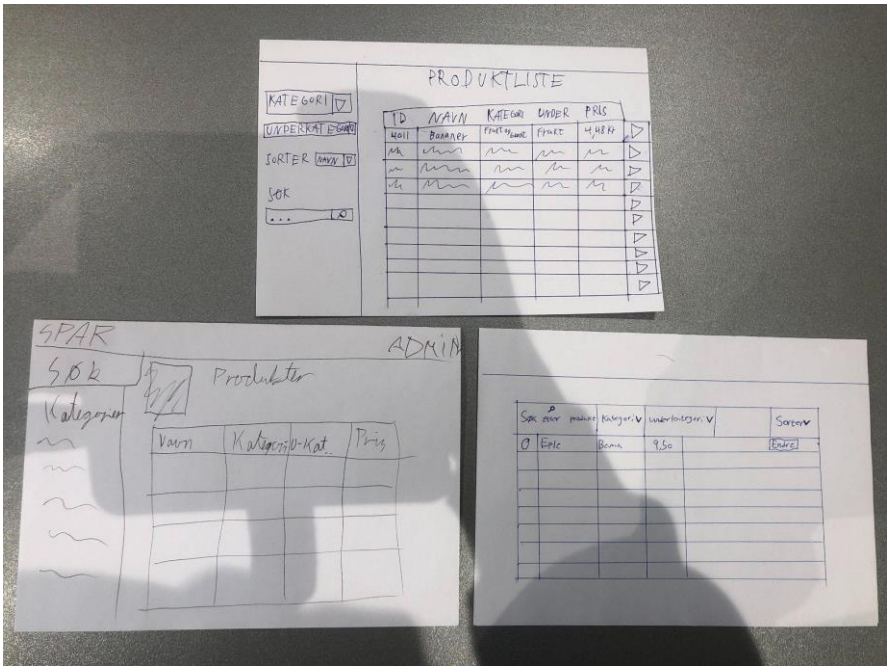
Vedlegg 6.7 - Login



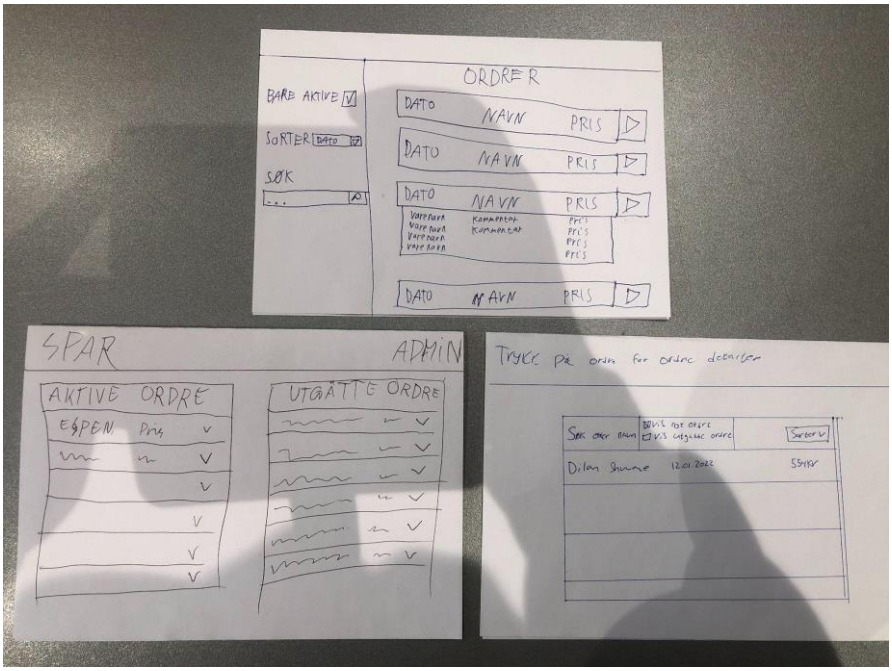
Vedlegg 6.8 - Admin



Vedlegg 6.9 - Produktliste

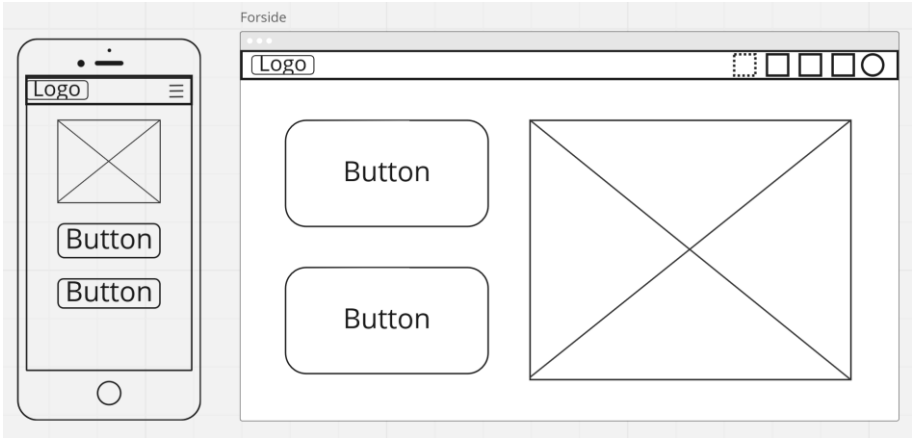


Vedlegg 6.10 - Ordreliste

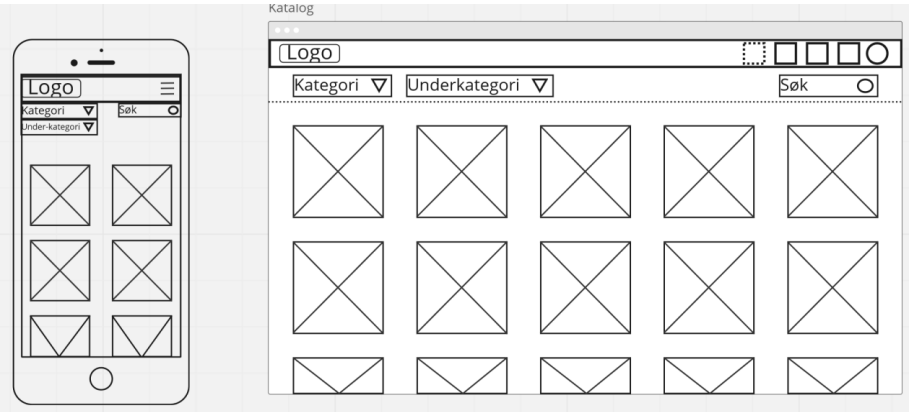


Vedlegg 7 - Wireframes

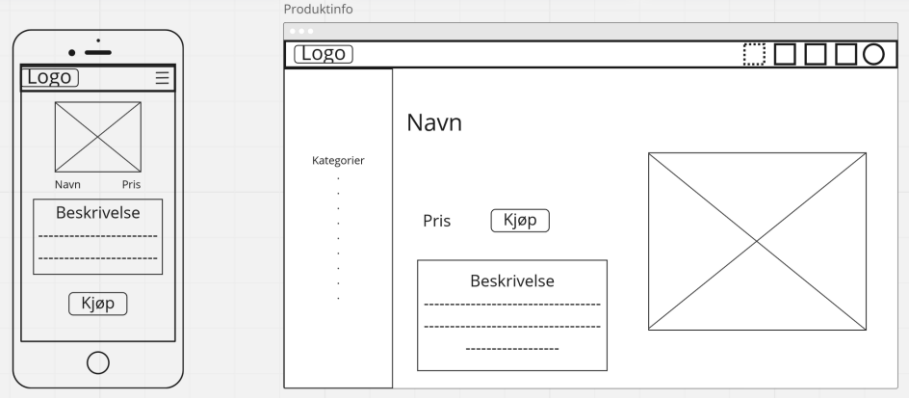
Vedlegg 7.1 - Forside



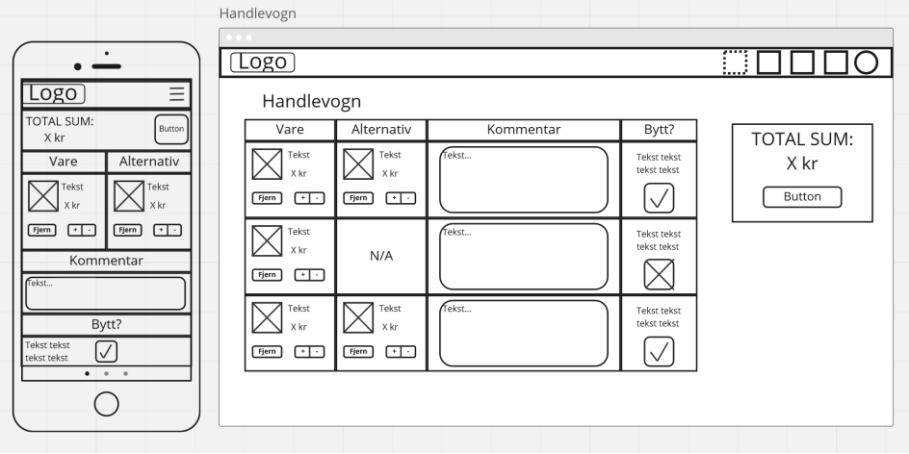
Vedlegg 7.2 - Katalog



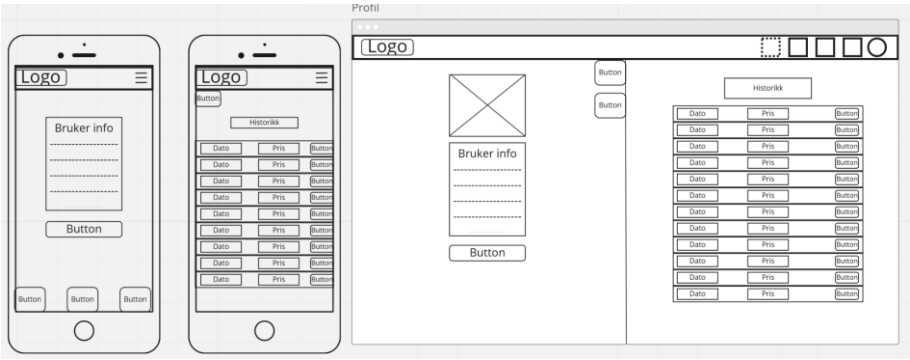
Vedlegg 7.3 - Produktinfo



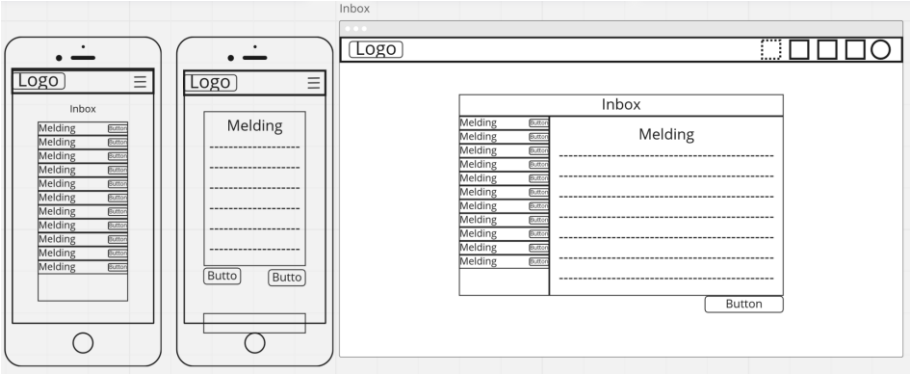
Vedlegg 7.4 - Handlevogn



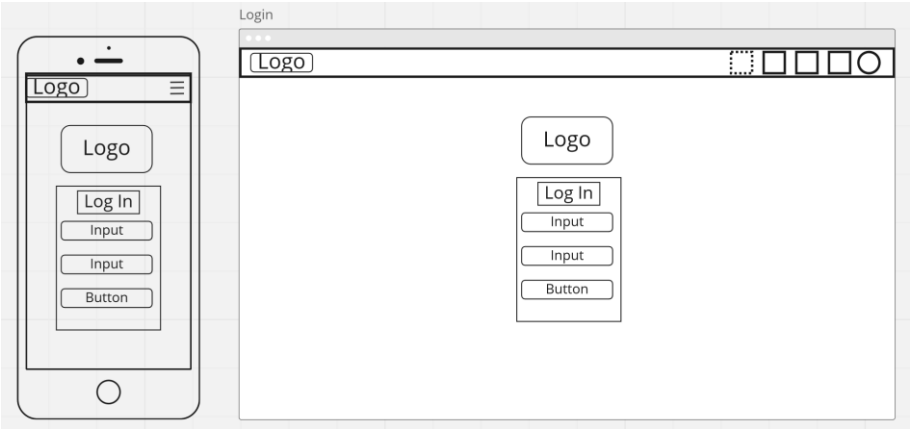
Vedlegg 7.5 - Profil



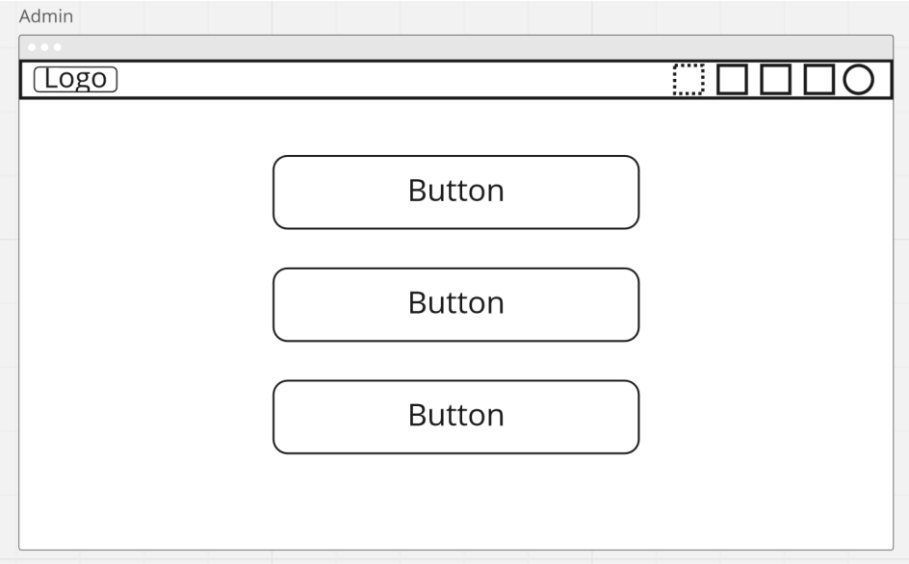
Vedlegg 7.6 - Innboks



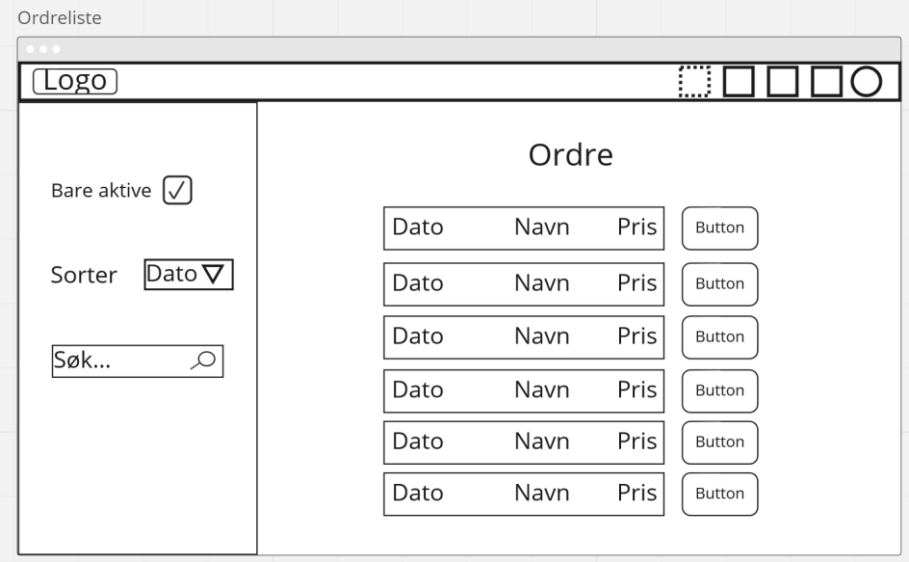
Vedlegg 7.7 - Login



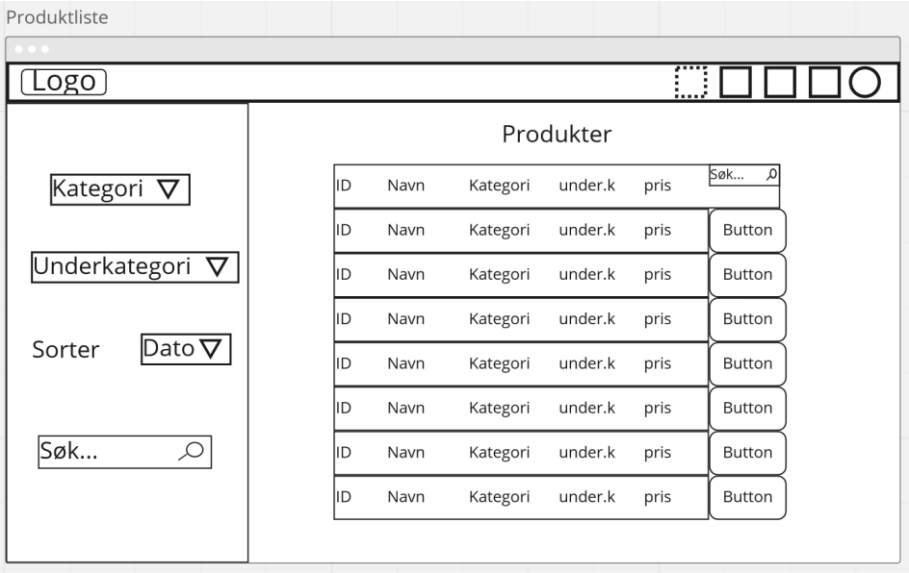
Vedlegg 7.8 - Admin



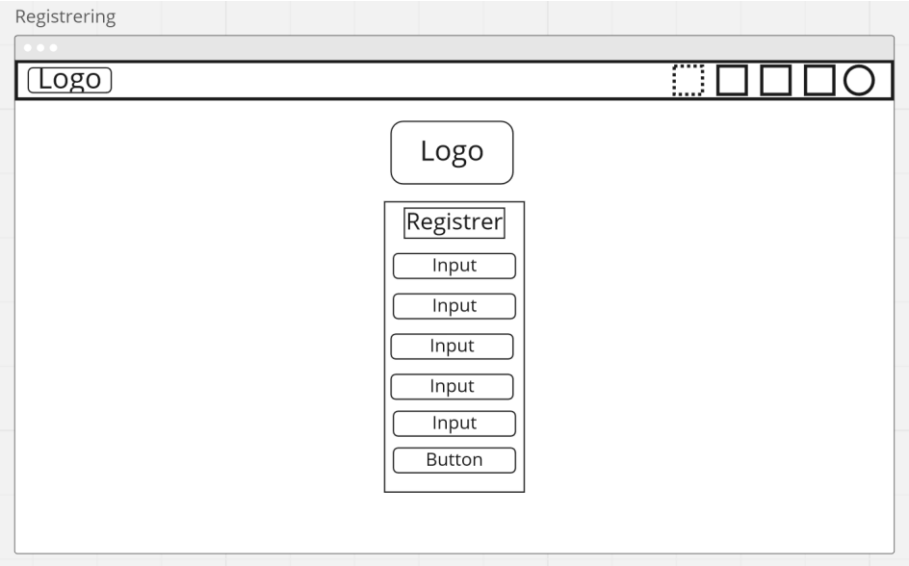
Vedlegg 7.9 - Ordreliste



Vedlegg 7.10 - Produktliste

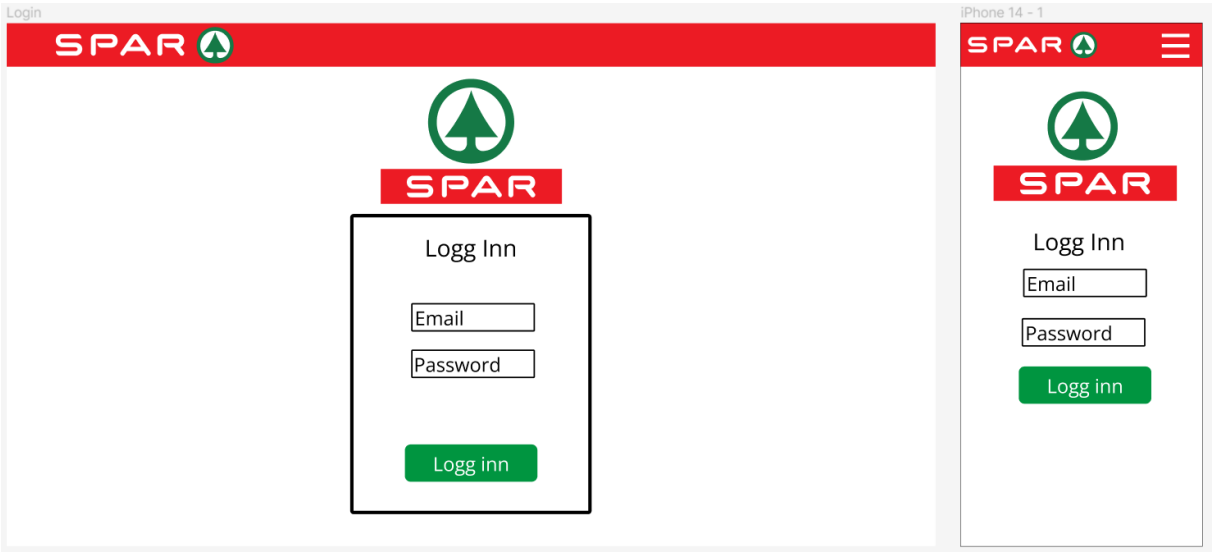


Vedlegg 7.11 - Registrering

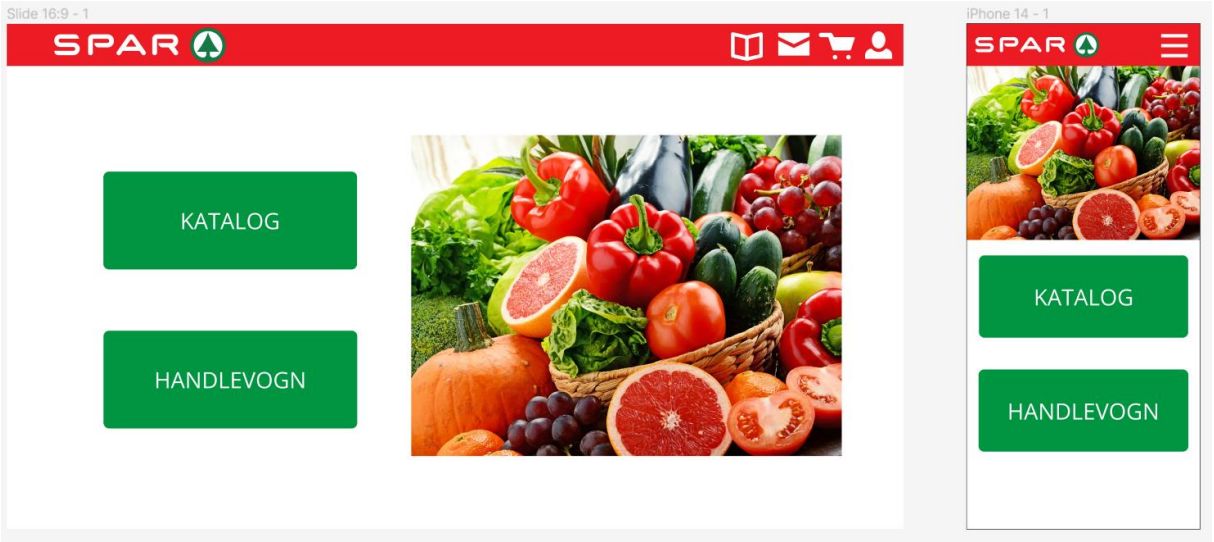


Vedlegg 8 – Mockups

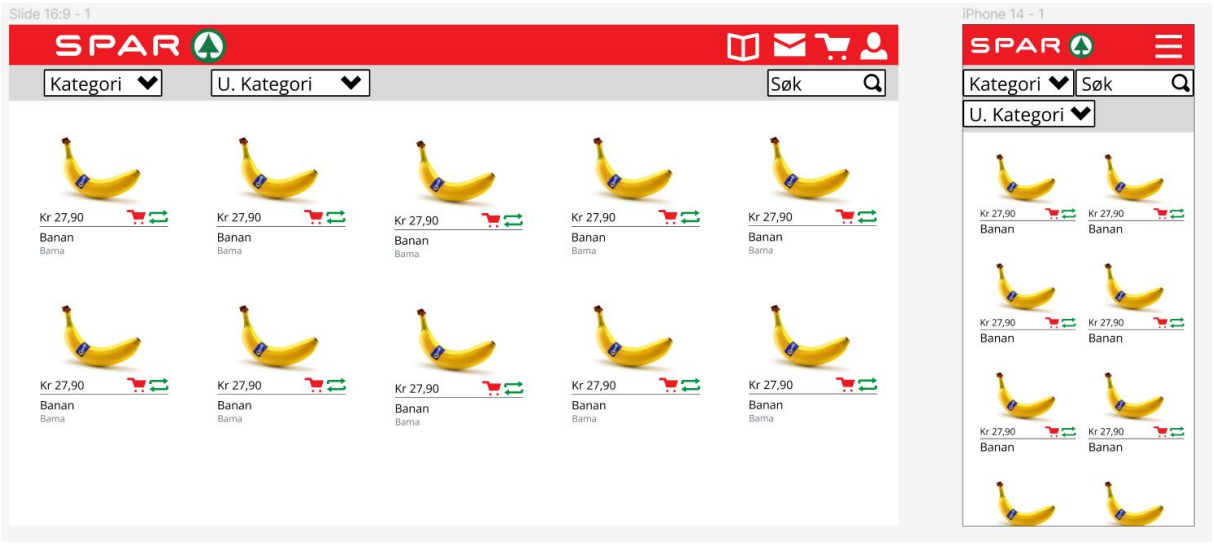
Vedlegg 8.1 - Login



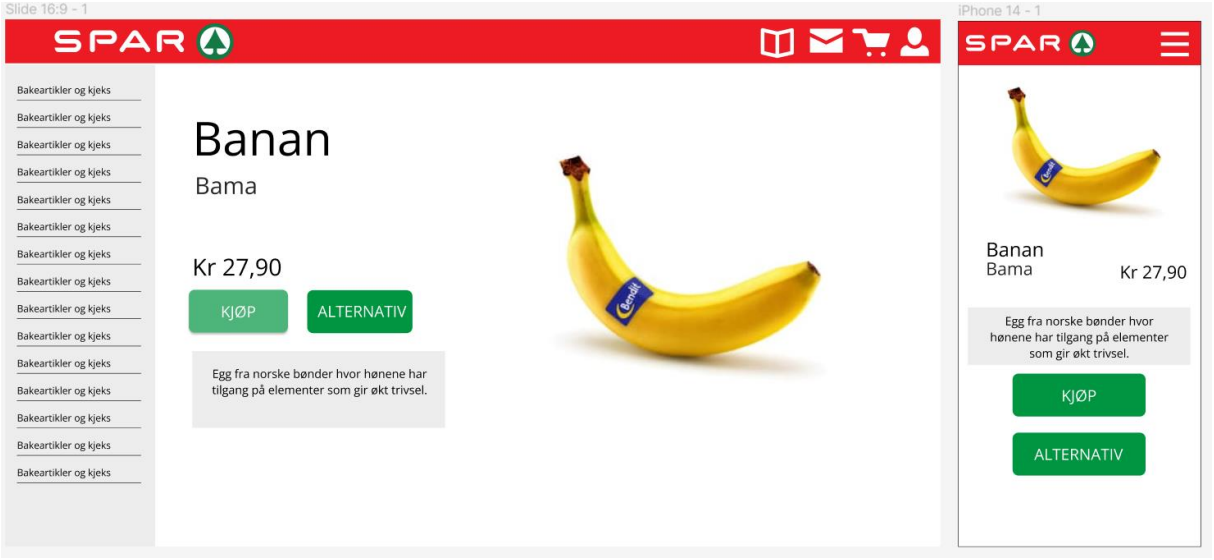
Vedlegg 8.2 – Frontpage



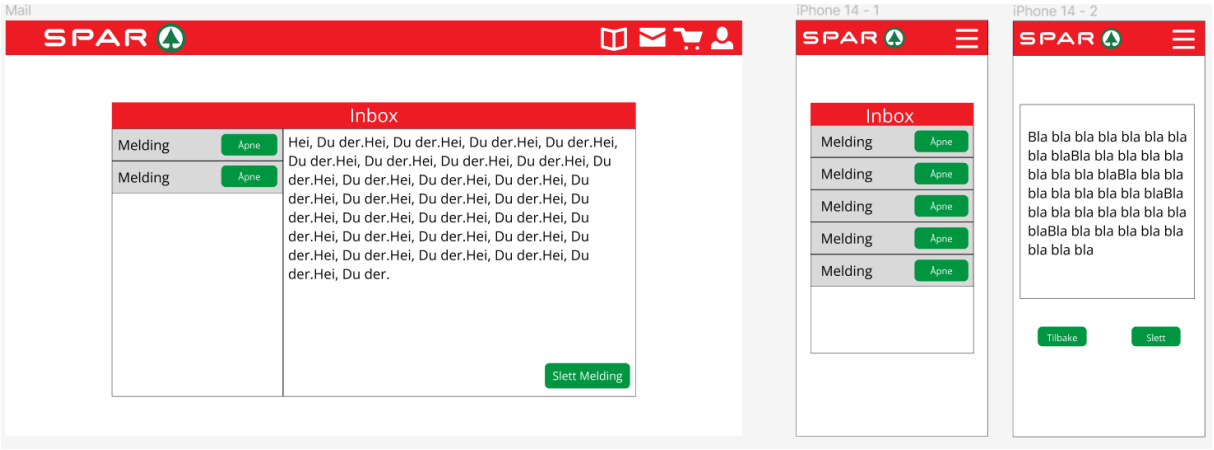
Vedlegg 8.3 – Katalog



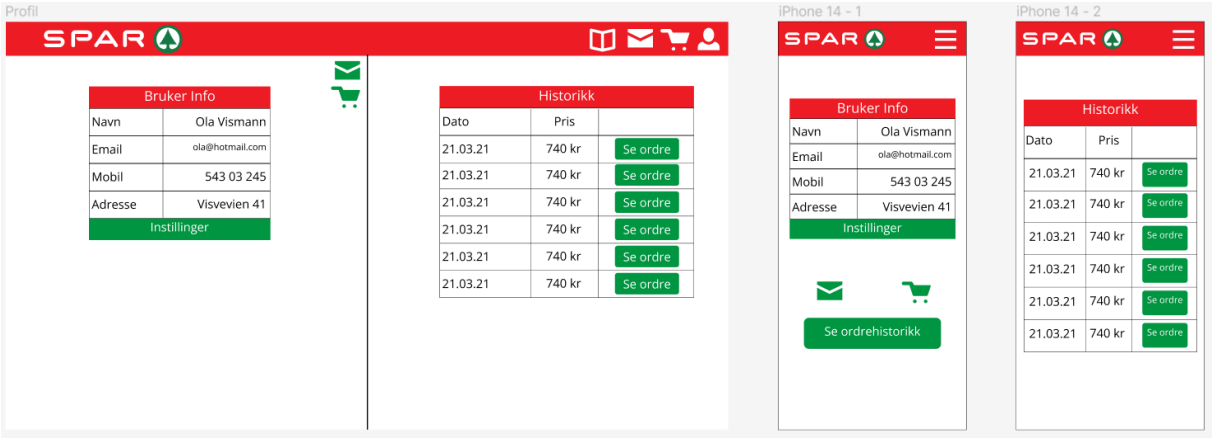
Vedlegg 8.4 - Produktinfo



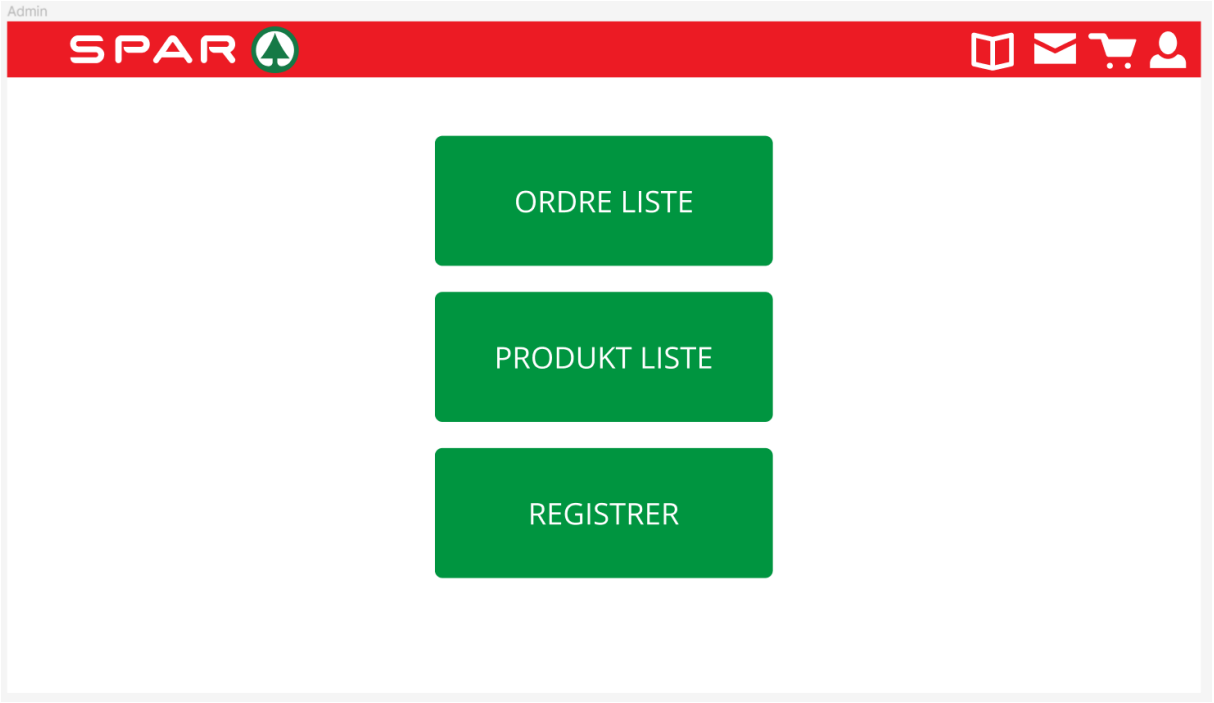
Vedlegg 8.5 – Innboks



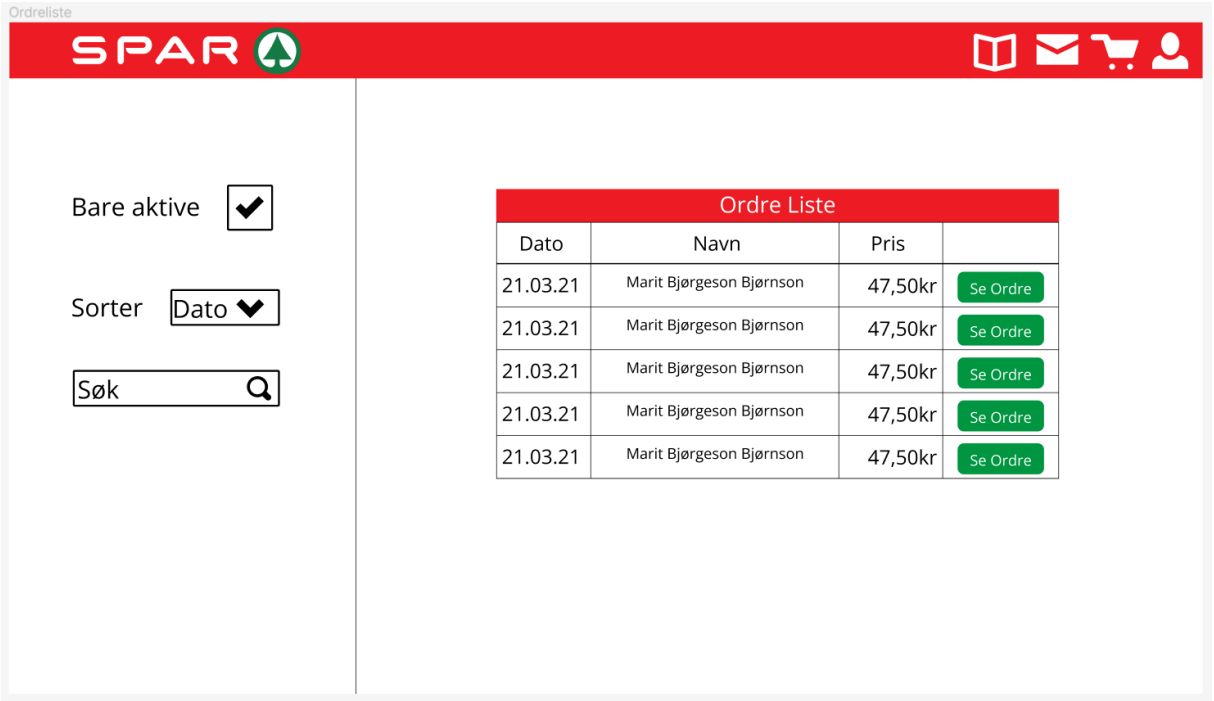
Vedlegg 8.6 – Profil



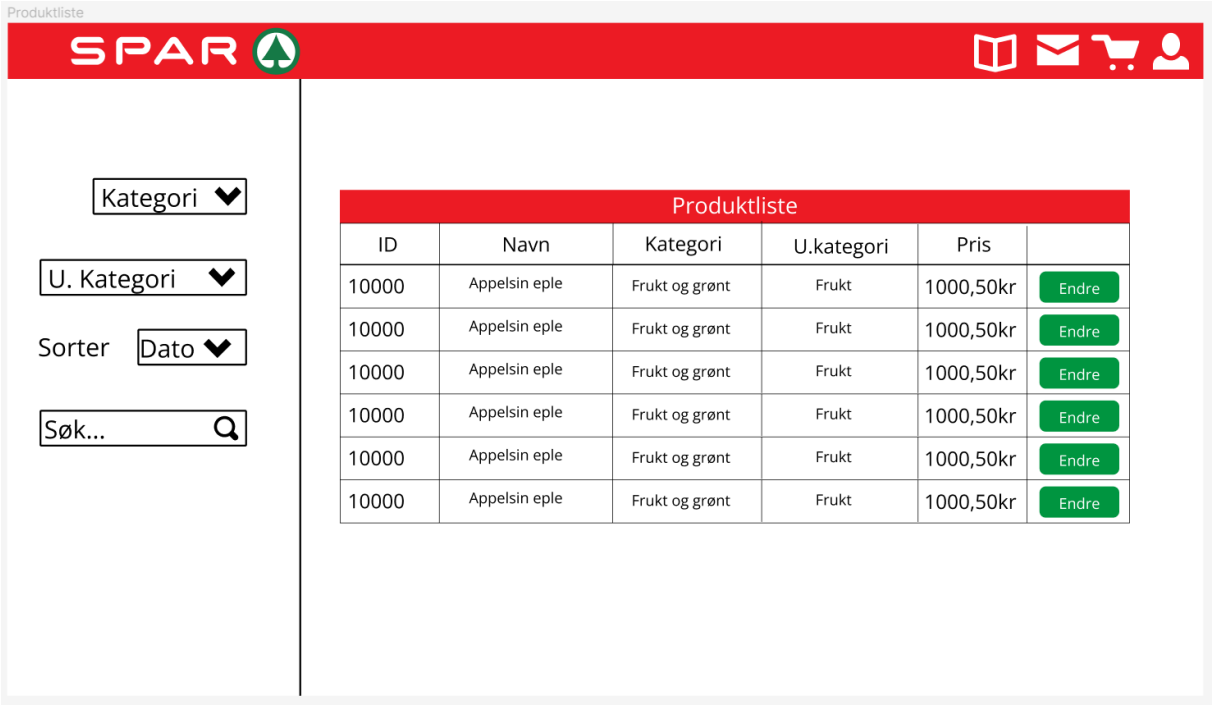
Vedlegg 8.7 – Admin



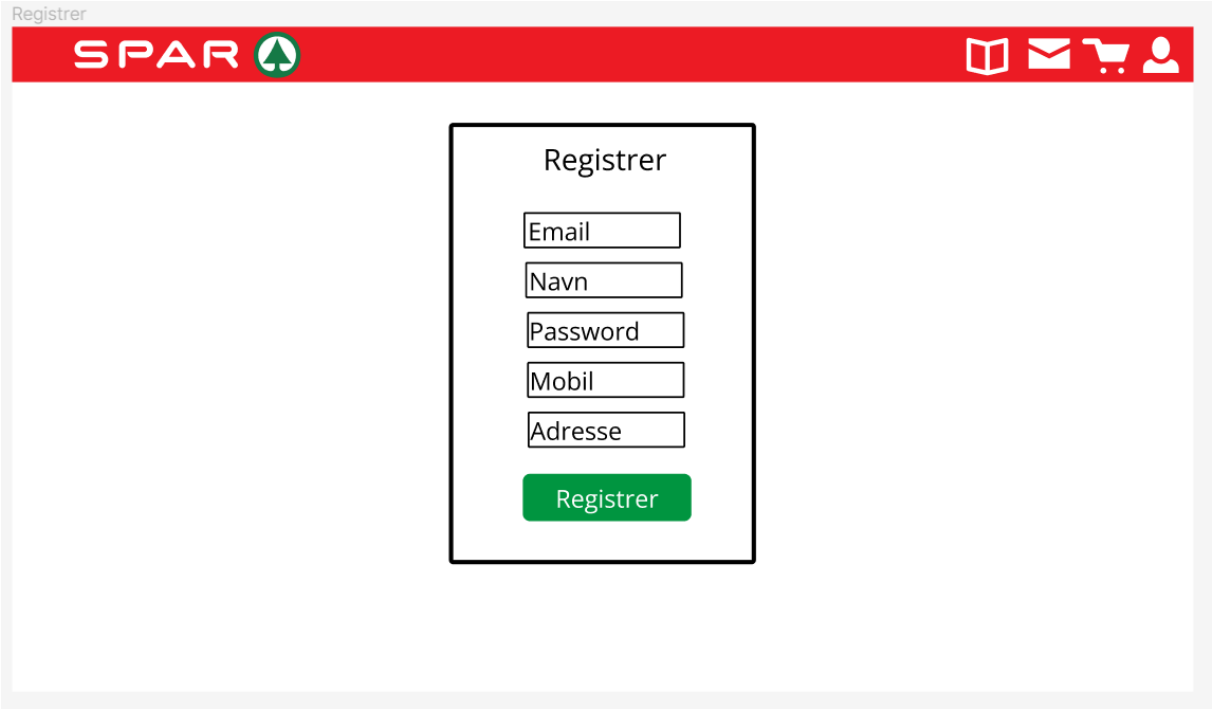
Vedlegg 8.8 – Ordreliste



Vedlegg 8.9 – Produktliste



Vedlegg 8.10 – Registrering



Vedlegg 8.11 – Produkt instillinger