NTNU
Kunnskap for en bedre verden

BACHELOROPPGAVE:

TITTEL

iKatalog

FORFATTERE:
HALVARD MEIJER
MAKSYMILIAN SADOWSKI

Dato: 17.05.2016

SAMMENDRAG

Tittel:	iKatalog	Dato : 17.05.16
Deltaker(e)/	Halvard Meijer	
	Maksymilian Sadowski	
Veileder(e):	Emil Bakke	
Oppdragsgiver:	SIC AS	
Kontaktperson:	Ib Christiansen	
Stikkord (4 stk)	Produkt katalog, React, Material Design, Javascript	
Antall sider: 92	Antall vedlegg: 7	Tilgjengelighet: Åpen

Kort beskrivelse av master/bacheloroppgaven:

Målet for prosjektet er å lage en produktkatalog med fokus på at produktene vises på en oversiktlig, konsis og pen måte. Katalogen skal følge Google Material Design prinsippene for å få likhet mellom katalogen og systemet til klienten. Den skal lages med Javascript rammeverket React og forlengelsen Material-UI til React. Katalogen skal gi en oversikt over produktene som finnes i kundens system. Det skal være enkelt å navigere seg mellom produktene og brukeren skal ha mulighet til å filtrere og sortere produktene slik at de enkelt kan finne de produktene som de ønsker uten å bli tvunget til å bla gjennom alle produktene

ABSTRACT

Title:	iKatalog	Date : 17.05.16
Participants/	Halvard Meijer	
	Maksymilian Sadowski	
Supervisor	Emil Bakke	
Employer:	SIC AS	
Contact:	Ib Christiansen	
Keywords (4)	Product catalog, React, Material Design, JavaScript	
Number of pages: 92	Number of appendix: 7	Availability: Open

Short description of the bachelor thesis:

The aim of this project is to create a product catalog that displays the products in a clear and concise way. The catalog should follow Google Material Design principles to get similarity between the catalog and the clients system. It should be created with the JavaScript framework React and the extension Material UI to React. The catalog will provide an overview of the products found in the customer's system. It should be easy to navigate between the products and the user should have the ability to filter and sort the products so that they can easily find the products they want without being forced to browse all the products.

Forord

En stor takk til vår veileder Emil Bakke for god støtte og veiledning.

Vi vil også takke oppdragsgiver, Sic AS, for at vi kunne gjennomføre oppdraget og for god hjelp under prosjektet.

Til slutt vil vi takke alle som har deltatt i intervjuer.

Innhold

1. Innledning 1

1.1. Bakgrunn 1
1.2. Prosjektbeskrivelse 1
1.3. Målgruppe 2
1.4. Faglig bakgrunn 2
1.5. Rammer 2
1.5.1. Scrum 2
1.5.2. Vår tilnærming til Scrum 2
1.6. Øvrige roller 3
1.7. Rapport organisering 4
1.7.1. Vedlegg 5
1.7.2. Terminologi 5

2. Kravspesifikasjon 6

2.1. Oppgavedefinisjon 6
2.2. Kodespråk og rammeverk 6
2.3. Favoritter og ordre 7
2.4. Enkel navigering 7
2.5. Fargebruk 7
2.6. Datasett 7
2.7. Sortering og filtrering 7
3. Verktøy 9

3.1. Rammeverk 9
3.2. Utviklingsmiljø 11
3.3. Prosjektstyring 13
3.4. Grafisk design 14
3.5. NPM 15
3.5.1. Package.json 16
3.6. Konklusjon 18
4. Design 19

4.1. Designanalyse 19
4.1.1. Intervju med klienten 19
4.1.2. Intervju med potensielle brukere 20
4.2. Første prototype 22
4.2.1. Konkurrenter 22
4.2.2. Skisser 23
4.2.3. Prototype 24
4.2.4. Testing/Tilbakemelding 26
4.3. Konklusjon 26
5. Implementasjon 27
5.1. Forberedelser 27
5.2. Første sprint 27
5.2.1. Første utgave i React 28
5.2.2. Testing 31
5.3. Andre sprint 31
5.3.1. Datasett 31
5.3.2. Produktene 32
5.3.3. Filtrering og Sortering 32
5.3.4. Produktsiden 34
5.3.5. Endring av farger 36
5.3.6. Andre utgave 37
5.4. Tredje sprint 39
5.4.1. Besøk klienten 39
5.4.2. Filtrering og nytt datasett 42
5.4.3. Produktene med nytt datasett 45
5.4.4. Dialogen 46
5.4.5. Siste utgave 48
5.5. Kvalitetssikring 50

5.5.1. Intervju 50
5.5.2. Testing/tilbakemelding 50
5.5.3. Google Material Design 51
5.5.4. Samarbeid med klienten 52
5.6. Konklusjon 52
6. Avslutning 53
6.1. Evaluering av oppgaven 53
6.2. Gruppearbeid 55
6.2.1. Arbeidsfordeling 55
6.3. Videre utvikling 56
6.3.1. Hva som må være på plass 56
6.3.2. Hva som kan være med 57
6.3.3. For implementering 57
6.4. Konklusjon 58
Litteraturliste 58
Vedlegg A 61
Vedlegg B 63
Vedlegg C 74
Vedlegg D 82
Vedlegg E 84
Vedlegg F 91
Vedlegg G 92

1. Innledning

1.1. Bakgrunn

Sic AS er en oppstartsbedrift i Oslo som har utviklet et ordrehåndteringssystem (Datorder[1]) for grossisthandel med klær og sko. Systemet er en web applikasjon bygget med reactkomponenter og backenden er laget i Clojure.

Kort fortalt så interfacer Datorder kundens ERP system og presenterer funksjoner og informasjon derfra i en webapplikasjon som er enkel å bruke for salgsapparatet. Datorder gir brukerne mulighet til å legge inn og følge opp ordre, se lagerstatus på varer, gi tilbakemelding om nye kolleksjoner og laste ned produktdata og bilder.

Oppdragsgiver bestemte seg å gå et steg videre og ga kundene sine en god måte å presentere produktene på. Derfor har de kommet med en oppgave hvor de ønsker seg en digital produktkatalog som kan integreres i Datorder. Katalogen gir brukerne mulighet til å se alle produktene på en visuelt oversiktlig måte og reduserer nødvendigheten av å trykke opp og distribuere fysiske kataloger.

1.2. Prosjektbeskrivelse

Målet for prosjektet er å lage en produktkatalog med fokus på at produktene vises på en oversiktlig, konsis og pen måte. Katalogen skal følge Google Material Design prinsippene for å få likhet mellom katalogen og systemet til klienten. Den skal lages med Javascript rammeverket React og forlengelsen Material-UI til React. Katalogen skal gi en oversikt over produktene som finnes i kundens system. Det skal være enkelt å navigere seg mellom produktene og brukeren skal ha mulighet til å filtrere og sortere produktene slik at de enkelt kan finne de produktene som de ønsker uten å bli tvunget til å bla gjennom alle produktene.

[1] **Datorder** - navnet til klientens systemet

1.3. Målgruppe

Målgruppen for prosjektet er alle som kommer til å benytte SIC AS sin løsning, det vil si klesmerker og klesdistributører. Løsningen blir også brukt av butikker, partnere og salgsapparater som tilhører klesmerker og klesdistributører.

1.4. Faglig bakgrunn

Gruppen består av to studenter med lik bakgrunn. Begge to går webutvikling ved NTNU i Gjøvik og har samarbeidet sammen flere ganger i ulike fag siden starten på bachelorgraden. I løpet av studiene, har gruppen fått erfaring fra programmering og webdesign. Tidligere, har vi erfaring med blant annet Javascript, php, html og css og har tidligere utviklet websystemer med fokus på brukervennlig grensesnitt. Vi har også erfaring med Google Material Design fra et tidligere prosjekt. Selvom prosjektet krever kjennskap rammeverk som er nytt, React og Material-UI, følte gruppen at oppgaven kom til å bli en spennende og lærerik utfordring.

1.5. Rammer

1.5.1. Scrum

Scrum er et rammeverk som er ofte brukt for å utvikle et programvare. I scrum utarbeides en produktbaklogg som er en oversikt over ønsker og krav til programvaren. Prosessen starter deretter med at teamet velger ut oppgaver fra produktbakloggen som de skal fullføre i løpet av de neste 30 dagene, dette blir kalt sprinter. I løpet av sprinten møtes teamet for å ha daglig møter, der de oppsummerer hva som har blitt gjort, hva som skal gjøres og hva som stopper fremgangen. Etter hver sprint skal det som har blitt gjort bli vist frem og det skal holdes et møte for å evaluere sprinten (1).

1.5.2. Vår tilnærming til Scrum

Vi har valgt å bruke Scrum for å utvikle katalogen, men fordi Scrum er et rammeverk som er optimalisert for større grupper, valgte vi å tilpasse Scrum for gruppen. Tidlig i prosessen ønsket vi å ha ni sprinter, men etter design prosessen var ferdig forsto vi at det er mer

hensiktsmessig å ha tre lange sprinter. På den måten får vi gjort klar en større versjon av katalogen etter hver sprint som vi kan vise frem og teste før vi setter i gang på neste sprint. Vi har daglige møter og jobber i felleskap, på den måten kan vi kan enkelt hjelpe hverandre, vi vet hele tiden hva som blir gjort av resten av gruppen og vi får en mye høyere forståelse for disse tingene. Hvis vi er usikre eller uenige får vi til en rask løsning ved å diskutere problemene med engang. Vi har notert ned en logg fra etter hver arbeidsøkt slik at vi enkelt kan gå tilbake å se hva vi har gjort når (vedlegg C). Produktbakloggen vår ble laget i et Google Docs dokument slik at vi hadde en liste over funksjoner og krav som måtte implementeres, men fordi vi hele tiden arbeidet sammen klarte vi oss ofte uten å bruke produktbakloggen. Etter sprintene viste vi frem utgaven av katalogen til klienten og fikk tilbakemelding.

1.6. Øvrige roller

Oppdragsgiver

SIC AS

Kontaktperson

Ib Christiansen ved SIC AS

Veileder

Førsteamanuensis Emil Bakke ved NTNU i Gjøvik

Gruppemedlemmer

Gruppen består av to medlemmer. Halvard Meijer og Maksymilian Sadowski.

1.7. Rapport organisering

Rapporten består av seks kapitler. I vedlegg finnes det tilleggsinformasjon som anbefales å sjekke når de refereres til i rapporten. For referanse henvisning har vi brukt Vancouver referansestil.

1. Innledning

Informasjon og organisering av prosjektet

2. Kravspesifikasjon

Kravene til det endelige produktet

3. Verktøy

Programvare og verktøy vi har brukt i løpet av prosjektet

4. Design

Design prosessen frem til prototype

5. Implementasjon

De tre sprintene og presentasjon av det endelige produktet

6. Avslutning

Drøfting og videre utvikling av prosjektet og evaluering av gruppens arbeid

1.7.1. Vedlegg

Vedlegg A

Prosjektavtale

Vedlegg B

Prosjektplan

Vedlegg C

Møtereferat

Vedlegg D

Terminologi

Vedlegg E

Skisser

Vedlegg F

Kode

Vedlegg G

Figurer og skjermbilder

1.7.2. Terminologi

Terminologi kommer fortløpende som bunntekst og en komplett liste ligger i vedlegg D

2. Kravspesifikasjon

I dette kapitlet skal vi gå gjennom kravspesifikasjonene til det endelige produktet. Vi går gjennom oppgavedefinisjon, kodespråk, rammeverk og de viktigste funksjonene klienten ønsker.

2.1. Oppgavedefinisjon

Oppgaven går ut på at vi skal lage en digital produktkatalog som skal implementeres i Datorder. Katalogen skal hovedsakelig inneholde sko og klær, og må følge prinsippene til Google Material Design. Produktene blir levert av oppdragsgiver i JSON [2] format. Hele løsningen skal lages ved bruk av React og Material-UI som klienten allerede bruker i sitt eksisterende system.

Produktene skal kunne sorteres og filtreres på sesong og kjønn. I selve katalogen skal vi vise produktnavn, fargenavn og både innkjøps og utsalgspris samt bilde av produktet. Hvis brukeren ønsker informasjon om sesong, kjønn, farge eller størrelse kan brukeren klikke på et produkt for å se denne informasjonen. Klienten ønsker også knapper for å legge til ordre og favoritter, og vil gjerne ha litt ekstra plass slik at de kan legge til en eventuelt tekst om hvert produkt.

2.2. Kodespråk og rammeverk

Katalogen skal bli produsert ved hjelp av React, Material-UI og JavaScript. For å bruke disse må vi ha en rekke pakker fra NPM. For å kunne bruke NPM, er det et nødvending at Node.js er installert på datamaskinen. Katalogen skal bruke dette slik at den fungerer likt som og at den har like design elementer som Datorder.

[2] JSON - står for JavaScript Object Notation. En JSON fil består av objekter med tabeller, hvor man kan lagre informasjon som kan bli hentet ved hjelp av JavaScript.

2.3. Favoritter og ordre

Det skal være mulig å interagere med produktene som finnes i katalogen. Det skal være mulig for brukeren å legge produktene til favoritter eller til bestilling. Klienten vil at vi skal legge opp knapper og design slik at klienten selv kan implementere dette til systemet deres.

2.4. Enkel navigering

Brukeren skal kunne navigere seg raskt gjennom systemet og få til alle funksjoner enkelt og uten feil. Katalogen skal være laget på en slik måte at brukeren allerede vet hvordan den fungerer uten opplæring. Google Material Design komponenter skal hjelpe til å sørge for at knapper, farger, lister osv. er like i hele systemet og gjenkjennbart fra andre applikasjoner.

2.5. Fargebruk

Klienten har en fargepalett som ikke er ferdig i dag. Katalogen skal tilslutt bruke samme fargene som resten av systemet, men siden den ikke er ferdigprodusert så er klienten interessert hvis vi har innspill. De vil også at vi skal ha i bakhodet at hele systemet vil bruke mørkefarger

2.6. Datasett

Databasen blir levert i JSON format hvor strukturen er basert på en reell database til en av kundene til SIC. JSON ligner på XML men er optimalisert for JavaScript og samtidig er den mer oversiktlig. Katalogen vil ikke trenge noe mer enn denne fila for å hente ut informasjon om produktene og vise disse fram i katalogen. Bildene ligger ikke i JSON-fila, men med hjelp av dataen den inneholder skal bildene kunne hentes frem ved hjelp av identifikasjon koder for produktene og fargene som finnes i dataoppsettet..

2.7. Sortering og filtrering

SIC ønsker at katalogen skal kunne sorteres og filtreres slik at brukeren enkelt kan finne produktene de ønsker. Slik kan brukeren se gjennom kun relevante produkter for eksempel

hvis brukeren ønsker å se klær som hører til sesongen FW16 (Fall/Winter 2016). I første omgang ønsker SIC å ha filtrering og sortering på kjønn, sesong, fargenavn og produktnavn.

3. Verktøy

Med verktøy mener vi alle rammeverk, utviklingsmiljø og programvare som har vært viktig for oss. Her skal vi forklare hvorfor har vi brukt akkurat disse verktøyene og samtidig forklare kort hvordan disse fungerer. Vi skal også se på NPM[3] som er et viktig verktøy for at prosjektet vårt skal fungere.

3.1. Rammeverk

I dag finnes det masse forskjellige rammeverk og biblioteker som utvider funksjonalitet til JavaScript. Disse er rettet mot forskjellige problemstillinger som kan møtes gjennom programmering. Når man velger rammeverk man skal jobbe med, er det viktig at man ser på om den tilbyr det man har behov for. Vi har brukt React og Material-UI som var et krav fra klienten.

React[4] er et JavaScript bibliotek som blir brukt for å lage brukergrensesnitt. Ved bruk av React kan man lage gjenbrukbare UI komponenter som blir brukt til å fremstille informasjon som kan bli endret over tid. Siden resten av Datorder systemet var utviklet ved bruk av dette biblioteket, var det et krav å bruke React. Fordi vi fikk produktene som en JSON fil slapp vi å bruke et kodespråk for å hente data fra en database. Med React kunne vi enkelt hente informasjonen fra JSON fila og vise den fram slik vi ønsket det. Siden en del av funksjonene som vi skulle implementere kunne ført til endring av data som blir vist i katalogen, kunne vi brukt React til å vise resultater av endringer som ble gjort for eksempel med sortering eller filtrering. Til dette bruker React noe som heter Virtuelt DOM[5].

---

[3] **NPM** - Node Package Manager
[4] https://facebook.github.io/react/
[5] **DOM** - Document Object Model er en beskrivelse hvordan HTML skal se ut som kan bli endret ved bruk av JavaScript

Vi kan se på dette som et objekt som blir formulert rundt et element som vi har laget, og ved interaksjon med elementet som kan føre til noe endring, vil objektet som er rundt elementet tilpasse seg til endringer og React vil vise disse fram. Dette er noe som skaper en god brukeropplevelse i løsningen vår. Brukeren slipper å laste opp nettsiden på nytt for å se resultater av sortering eller filtrering siden React ved å observere endringer i virtuell DOM vil kun oppdatere det elementet som har endret seg uten å laste opp resten av nettsiden på nytt. Brukeren merker ikke engang "refresh" som blir kjørt siden React tar for seg hele jobben i bakgrunn uten å forstyrre brukeren.

Siden React ble utviklet og blir brukt av Facebook og Instagram, var det fort forståelig at biblioteket kan på en rask måte behandle store mengder av data. Dette var også noe viktig å ha fokus på siden katalogen vår fort kan bli brukt til å fremstille store kolleksjoner med hundrevis eller tusenvis produkter. Samtidig ble biblioteket tatt i bruk på Facebook og Instagram før det ble offentliggjort, noe som sikrer at bruken av React i prosjektet vårt blir problemfritt (2).

Material-UI[6] er et rammeverk som består av mange ulike komponenter som følger Google Material Design prinsippene. Material-UI gjør at komponenter ser like ut gjennom hele systemet og at de er gjenkjennbare fra andre applikasjoner. Rammeverket er tilpasset React som har gjort at vi kunne på en enkel måte implementere Material Design i katalogen vår. På nettsiden til Material-UI fant vi mange ferdig bygde komponenter som vi kunne ta i bruk i katalogen og samtidig tilpasse disse ettersom det var behov for det. Ved bruk av disse har vi også lært hvordan vi kan hente ut data fra en JSON fil og benytte informasjonen i komponenter som blir levert med rammeverket. Siden hele systemet til SIC er også designet ved bruk av Material-UI, blir det mye lettere for dem å tilpasse katalogen til resten og samtidig koble opp katalogen til samme fargepaletten som Datorder bruker.

[6] http://www.material-ui.com/#/

Selv om vi har brukt lang tid for å bli kjent med begge løsninger, mener vi at det var verdt å bruke den tiden vi trengte for å lære seg disse. Når vi endelig hadde kjennskap til hvordan vi kan benytte React og Material-UI, var det lettere for oss å gjennomføre oppgaven og samtidig oppfylle kravene som vi har fått fra klienten. For at vi kunne starte arbeidet med prosjektet, måtte vi også sette opp utviklingsmiljø slik at vi best mulig kan bruke disse løsningene.

3.2. Utviklingsmiljø

Utviklingsmiljø er en type programvare som er som regel benyttet for programmering. Disse har ofte mange innebygde funksjoner som gjør utvikling og implementasjon lettere. Utviklingsmiljøet vi har brukt gjennom hele utviklingsprosessen er WebStorm, som vi har støttet med Git og SourceTree.

JetBrains WebStorm[7] er programmet vi har benyttet for å programmere. WebStrom er et tekst behandlingsverktøy med stor fokus på JavaScript. Vi likte programmet på grunn av alle de funksjonene programmet har å tilby. For eksempel en terminal som vi kan bruke med engang vi åpner prosjektet vårt. Vi slipper å navigere oss gjennom terminalen for å finne prosjektet og i stedet for, WebStorm finner den for oss ut ifra hvor prosjektet ligger. På grunn av dette kunne vi alltid i full fart kjøre skripter for å begynne og arbeide med prosjektet. Programmet går også gjennom hele koden som ble skrevet for å finne om det er noe feil som vil føre til at applikasjonen vil ikke fungere på riktig måte. Linjen hvor feilen oppstår vil bli fort markert med rød farge som hjelper med å finne en feil på en effektiv måte og hindre for at det vil oppstå noen problemer i løsningen vår. For at vi kunne dele all koden som ble skrevet i WebStorm mellom hverandre, måtte vi finne noe som kunne hjelpe oss med dette.

[7] https://www.jetbrains.com/webstorm/

Bitbucket[8], Git[9] og SourceTree[10] har vi brukt til å samarbeide under programmering. Bitbucket er et sted hvor vi kan laste opp all koden som vi har skrevet. Git er versjonskontrollsystemet vi bruker med Bitbucket og det fletter kode sammen fra ulike kilder. For å håndtere Bitbucket - oppbevaringsstedet bruker vi SourceTree. Programmet er gratis og med enkelt grensesnitt, det forenklet hele prosessen ved å kjøre ulike Git funksjoner i et visuelt brukergrensesnitt. Vi valgte å bruke Bitbucket og Git fordi vi har erfaring med disse fra tidligere og ved hjelp av SourceTree kan vi enkelt se hva vi har gjort tidligere, gå tilbake hvis noe er gjort feil og når vi skal arbeide med rapporten så gjør disse programmene det enkelt å se hva vi har gjort og når det ble gjort. Disse tre programmene har hjulpet oss også med å samarbeide og dele kode gjennom prosjektet uten at vi har mistet kode. Samtidig som de gjør det mulig å jobbe i samme fil uten at det blir problematisk å slå sammen endringer vi har gjort.

Under prosjektet har vi hatt problemer med SourceTree som førte til at vi kunne ikke bruke programmet til å dele arbeidet mellom hverandre. Selv om en av oss har sendt endringer, så kunne ikke den andre hente fram endringene. Feilmelding som vi fikk ga ingen mening, men vi fant ut fort at bedriften som står bak SourceTree har varslet alle brukere gjennom Twitter kontoen sinat [11]det finnes en del feil i siste versjonen som fører til at brukeren ikke får brukt standard funksjoner i programmet. Det kom også en anbefaling om å installere en tidligere versjon av programmet men det løste ikke problemet hos oss. Denne situasjonen har skapt en del problemer og treigheter under implementasjon av prosjektet. For å omgå problemet har vi gjennomført alle Git aksjoner i terminalen. Vi måtte bruke tid på å finne ut hvordan ting som vi kunne gjennomført på enkelt måte i SourceTree skulle gjennomføres i terminalen. Heldigvis ble problemene med SourceTree fikset dagen[12] etter og da kunne vi fortsette å samarbeide på effektiv måte igjen.

---

[8] https://bitbucket.org/
[9] https://git-scm.com/
[10] https://www.sourcetreeapp.com/
[11] https://twitter.com/sourcetree/status/699844545094295552
[12] https://twitter.com/sourcetree/status/701850745474326531

Disse programmene har hjulpet oss med å utvikle løsningen vår. Vi hadde alle verktøyene vi trengte i WebStorm. Med hjelp av WebStorm hadde vi god oversikt over alle filene i prosjektet, feilene i koden og vi kunne benytte terminalen i programmet da vi trengte dette. Bitbucket og SourceTree har forenklet prosessen med å dele kode mellom hverandre uten at det oppstår problemer med filene. Selv om vi hadde en episode der programmet fungerte dårlig, så fungerte programmet problemfritt etter de fikk fikset feilen. Videre skal vi se på verktøyene vi har brukt for å ha god kontroll på filene tilhørende rapporten og programvaren vi har brukt til å holde god kontakt med klienten vår.

3.3. Prosjektstyring

For å ha god prosjektstyring har vi brukt en del nyttig programmer som finnes i dag. Disse hjalp oss med å ha god kontroll over alle dokumenter som vi har skrevet gjennom hele prosjektet og samtidig ha disse tilgjengelig for alle gruppemedlemmene. Vi har underveis brukt Google Drive som vi har god erfaring med og to løsninger som ble anbefalt av klienten.

Google Drive[13] er en gratis skyløsning, som gir oss mulighet til å lagre filer som kan deles med andre og skrive i samme dokumenter. På denne måten kan alle i gruppen skrive samtidig i dokumenter, uten at det blir problemer med lagring. Samtidig får alle oversikt over alle endringer som ble gjort i forskjellige dokumenter, slik at vi vet hvem som har gjort hva. Løsningen har hjulpet oss med få notert informasjon fra forskjellige møter, intervjuer og felles rapportskriving. På grunn av dette, hadde vi god oversikt over alle filer som var lagret på et sted. Siden Google Docs, som er levert sammen med Google Drive, ikke er optimalt for å ferdigstille rapporten, har vi valgt å bruke Microsoft Word til dette.

[13] https://www.google.com/intl/no/drive/

Slack[14] er en online melding tjeneste som vi har brukt til å holde kontakt med oppdragsgiver vår og andre som jobber hos SIC. Slack har gitt oss muligheten til å kommunisere med klienten, slik at vi blant annet har fått kjappe svar på spørsmål eller fått hjelp hvis vi det har vært problemer under utviklingen av katalogen. Med programmet kunne vi på enkelt måte dele kodesnutter og små filer mellom hverandre. For å dele større filer har vi benyttet Dropbox.

Dropbox[15] er en skytjeneste som ble brukt for å dele store og samtidig flere filer mellom oss og klienten. Der lastet de opp filer, bilder og datasett slik at vi alltid hadde tilgang til de filene vi trengte. Siden datasettet ble endret gjennom prosjektet flere ganger, så kunne vi hente den nye og oppdaterte versjonen på Dropbox, etter avtalt endringer med oppdragsgiver.

Programmene har hjulpet oss med styre prosjektet på en god måte. Vi hadde et sted hvor vi hadde alle dokumentene tilgjengelig og samtidig slapp og sende disse manuelt mellom hverandre. Med satt opp chat-løsning mellom oss og klienten hvor vi kunne alltid diskutere ting som vi lurte eller avtale møter hvis det var noe vi måtte ta opp muntlig. Samtidig hadde vi en løsning hvor vi hadde tilgang til filer som oppdragsgiver har delt med oss. Med alt dette kunne vi jobbe effektivt som gruppe og samtidig ha god kontakt med klienten. Videre skal vi se nærmere på verktøy som vi har brukt for bilderedigering og prototyping.

3.4. Grafisk design

I løpet av prosjektet var det en del grafisk arbeid vi måtte gjennomføre, blant annet redigering av bilder som skulle bli brukt i katalogen og utvikling av en prototype. For dette har vi brukt programmer som ble anbefalt av oppdragsgiver.

[14] https://slack.com/
[15] https://www.dropbox.com/home

Gimp[16] er et program for digital bildebehandling som vi har brukt for å lage bilder av produkter. Vi brukte programmet til å redigere bilder som klienten har laget for oss. Siden vi var kjent med den lignende løsningen Adobe Photoshop, var det ikke vanskelig for oss å ta i bruk Gimp. Samtidig hjalp den oss med å samarbeide med klienten, fordi alle bildene av produktene som de leverte ble lagret i Gimp sitt eget filformat. Vi kunne lett endre farge på klær eller bakgrunnsbilde slik at vi hadde mulighet for å prøve bilder med for eksempel lys eller mørk bakgrunn. Ved å ha forskjellige farger på produktene kunne vi skape variasjon i katalogen. Vi vurderte og lage prototypen manuelt ved bruk av Gimp men vi fant en god løsning som vi kunne bruke for å lage en prototype på en effektiv måte.

Proto.io[17] er en webbasert løsning for å utvikle prototyper. Med programmet kunne vi enkelt lage en interaktiv prototype som vi kunne bruke til tidlig testing. Proto.io har også innbygd Material Design komponenter som vi kunne bruke for å utvikle prototypen med design prinsippene som klienten ønsket at vi skulle følge. Samtidig var det mulig å legge til enkle funksjoner som for eksempel åpning av produktsiden, navigasjonen eller endring av farger.

Med disse løsningene kunne vi jobbet med grafisk design på en effektiv måte. Vi kunne enkelt bearbeide bilder for katalogen og samtidig utvikle en god prototype som kunne bli brukt tidlig for å gjennomføre testing og få tilbakemelding fra klienten. Tilslutt skal vi se på NPM som var fundamentet for å kunne begynne å programmere.

3.5. NPM

For å ta i bruk React og Material-UI måtte vi bruke NPM[18] som er en standard “package manager” til Node.js og blir brukt av JavaScript utviklere slik at de enkelt kan dele kode mellom hverandre. Det deles gjenbrukbar kode som kalles “pakker” og du kan tenke på dem

[16] https://www.gimp.org/
[17] https://proto.io/
[18] https://www.npmjs.com/

som byggesteiner med kode som du kan bruke til å bygge opp og fikse problemer i din egen applikasjon. En applikasjon kan fort bestå av titalls eller hundrevis av pakker og det gjør det mulig lage store applikasjoner med disse byggesteinene. NPM brukes gjennom terminal og det er enkelt å laste ned eller oppdatere pakkene du har lastet ned. Det finnes hundretusenvis av pakker og det er over 40 millioner nedlastinger om dagen. På nettsiden til NPM er det et bibliotek av alle pakkene der du kan lese en forklaring på hva pakken gjør og hva som må gjøres for at den skal fungere. Dette gjør det enkelt å finne de pakkene du trenger og enkelt å bruke de riktig. Det er enkelt å laste ned pakkene selv om du må bruke terminal, kommandoen "NPM install" etterfulgt av navnet på pakken er alt du trenger å skrive for å laste den ned og NPM laster deretter pakken ned og installerer den slik at du kan bruke den med engang.

Mange av pakkene som finnes kan ofte installeres manuelt ved å laste ned en .zip fil og pakke den ut i prosjekt mappa. Man kan også koble prosjektet til en link som vil hente pakkene fra internett. Det er noe som virket lettere og bruke enn selve NPM løsningen siden vi hadde ingen erfaring med den, men problemet vårt var at de pakkene som vi skulle bruke i prosjektet som Material-UI, var kun mulig å installere via NPM. Derfor måtte vi lese oss opp for å finne hvordan vi kan benytte NPM for at all det grunnleggende som trengtes for å jobbe med prosjektet var på plass. På den offisielle nettsiden til NPM finnes det masse dokumentasjon og video tutorials hvordan man skal ta NPM i bruk (3). Vi fant ut at for å bruke NPM måtte vi sette opp noe som heter Package.json, en fil som samler viktig informasjon om prosjektet slik at alt fungerer og at andre kan installere prosjektet ved å benytte akkurat denne fila.

3.5.1. Package.json

For å begynne med NPM var det nødvending å generere en package.json fil noe som kunne bli gjennomført ved bruk av kommandoen "npm config" i terminalen. Gjennom prosessen måtte vi fylle ut en del informasjon om prosjektet blant annet, navn, beskrivelse eller versjon. (4). Siden prosjekt vårt ikke blir publisert offentlig på NPM, hadde vi ikke behov for å fylle ut alle feltene. Noen av feltene var meget viktig under prosjektet for å jobbe effektivt.

Vi har for eksempel brukt feltet **scripts** som hjulpet oss med å definere skriptene som blir gjenbrukt gjennom hele prosjektet. Ved å angi et navn for et script kan man på enkelt måte kalle skriptet ved å skrive "npm run" etterfulgt av script navnet som ble definert tidligere.

En av de viktigste informasjon som package.json inneholder er liste over alle pakker som er nødvendig å laste ned for å få prosjektet til å fungere. Listen over disse ligger under **devDependencis** i package.json. På grunn av dette slipper vi å laste opp alle node moduler som ble installert for å få prosjektet til å fungere og kan i stedet bare skrive "npm install". Da vil NPM hente og installere alle pakkene. For å slippe unødvendig opplasting og nedlasting av disse i oppbevaringsstedene har vi ignorert disse filene i «gitignore». Denne filen setter opp liste over ting som kan bli ignorert når man skal commite[19] og pushe[20] noe. Med dette kunne vi minske risikoen for at det skulle oppstå noen problemer med modulene under utviklingsprosessen.

Hvilke NPM har vi brukt?

- **React** - pakken som gir oss tilgang til React slik at det kan bli brukt i prosjektet vårt
- **React-DOM** - pakken som er nødvendig å ha slik at React og alt rundt DOM fungerer.
- **Browserify[21]** - denne hjelper oss med å bruke pakkene som vi har installert med NPM i prosjektet vårt. Løsningen importerer disse i hovedfila vår - main.js[22]. Browserify går gjennom hele koden og alle pakkene som ble inkludert i main.js og oversetter alt til en ny fil kalt bundle.js[23]. Bundle.js kobler opp alle nødvendig pakker og viser resultatet av koden i nettleseren.
- **Watchify[24]** - for at main.js fila blir oversatt til bundle.js ved bruk av Browserify, må vi bruke en kommando for at hele prosessen blir gjennomført etter hver endring som

[19] **Commit** - velge filer som vi skal pushe og gi en kommentar
[20] **Push** - laste opp endringene vi har gjort i repository
[21] http://browserify.org/
[22] **Main.js** - hovedfil hvor ligger hele koden til katalogen som vi har skrevet ved hjelp av React, Material-UI og JavaScript
[23] **Bundle.js** - JavaScript fila katalogen bruker. Denne blir lagd av Browserify og generert ut ifra main.js
[24] https://github.com/substack/watchify

blir gjort i prosjektet. For å slippe dette, har vi brukt Watchify som en utvidelse av Browserify. I stedet for å skrive en kommando hver gang vi endrer noe, kunne vi bare starte watchify som vil følge med på endringer i bakgrunnen. Dette automatiserer hele prosessen med å bygge bundle.js hver gang det blir gjort noe i prosjektet som fører til at hele utviklingsprosessen blir mye mer effektiv.

- **Babel - CLI**[25] - pakken som gjør at det blir mulig å bruke JSX og ES2015 som er utvidelser for JavaScript som forenkler og utvider kodespråket. Den hjelper oss med å gjøre begge to utvidelser forståelig for nettleseren. Det er ikke et krav å bruke pakken men React utviklere anbefaler å benytte denne løsningen siden koden blir mye mer forståelig og samtidig lettere å lære for nybegynnere (5). På grunn av dette kunne vi bruke mindre tid på å bruke lange JavaScript koder og i stedet for erstatte disse med kortere skripter ved bruk av JSX og ES2015 som gjør akkurat det samme. Vi trengte også å installere ekstra utvidelser til denne pakken slik at den fungerer sammen med Browserify (Babelify) og React (Babel-preset-es2015 og -react).

3.6. Konklusjon

Alle verktøyene vi har brukt har hjulpet oss med å gjennomføre ulike oppgaver i løpet av prosjektet. De fleste av disse løsningene har vi erfaring med fra tidligere, men de mest essensielle løsningene som NPM, React og Material-UI var nytt for oss. På grunn av dette, har vi brukt en del tid til å lese oss opp og lære om disse mens vi har jobbet med designprosessen.

[25] https://babeljs.io/

4. Design

I dette kapitlet skal vi forklare hvordan vi har jobbet oss gjennom designprosessen. Her skal vi gjøre rede for valgene vi har gjort for å lage en oversiktlig og brukervennlig produktkatalog. Valgene vil være basert på resultater fra forskning og intervjuer samt tilbakemelding som vi fikk underveis fra klienten.

4.1. Designanalyse

Vi har begynt hele designprosessen med å samle informasjon om hvordan løsningen skal se ut og hva som er viktig å fokusere på når vi skal lage designet for produktkatalogen. Metoden vi har brukt for dette er kvalitative intervjuer som Kristen Ringdal kaller for samtaleintervjuer. Hensikten med denne type løsning er å hente informasjon som kan være vanskelig å få tilgang til og samtidig få bekreftelser eller avkreftelser på data som ble hentet fra andre kilder (6).

Siden vi hadde lite erfaring med bruk av kataloger, bestemte vi oss for å begynne med et ustrukturert intervju med klienten vår. I et ustrukturert intervju, spørsmål er ofte lite definert på forhånd. På grunn av at vi var ikke så kjent med tema ennå, viste ikke vi helt hva vi skal snakke om, derfor var det lettere for oss å gjennomføre denne type intervjuet i form av uformell samtale (7). På denne måten kunne vi få bedre forståelse innen feltet som vi skulle jobbe med seinere og samtidig få grunnleggende informasjon, som vi kan bruke til å stille mer konkrete spørsmål når vi skal gjennomføre flere intervjuer.

4.1.1. Intervju med klienten

Intervjuet med klienten ble gjennomført i form av en samtale. Gjennom samtalen fikk vi bedre innsikt i hvordan hele Datorder systemet fungerer og hvordan det skal være koblet til katalogen som vi skal utvikle. Da vi pratet om designet av katalogen fikk vi beskjed om at vi har ganske frie tøyler så lenge vi følger Google Material Design prinsippene. Vi ble anbefalt å se på andre applikasjoner som bruker Material Design for å få bedre inntrykk av hvordan det er brukt. Siden klienten vår hadde erfaring med engroshandel av klær, fikk vi innføring i

hvordan hele prosessen blir gjennomført. Klienten fortalte oss at mange bedrifter fortsatt benytter seg av gammeldagse løsninger som papirkataloger hvor de fyller ut papirskjemaer eller Excel dokumenter med produkter som de vil bestille, som ofte fører til mye rot og kan skape en del problemer når disse skjemaene går gjennom flere personer og blir endret uten at noen har god oversikt over endringer som blir gjort. Derfor vil de forenkle hele salgsprosessen ved å automatisere den i Datorder.

Med intervjuet fikk vi bedre forståelse på hva hele systemet går ut på og ble bedre kjent med måten engroshandel blir gjennomført og hva klienten vår vil oppnå med en digital løsning. Informasjon som vi har samlet gjennom intervjuet, brukte vi til å forberede oss til intervju med andre personer, slik at vi kunne stille mer konkrete spørsmål og samtidig avkrefte og bekrefte det vi snakket med klienten om.

4.1.2. Intervju med potensielle brukere

Etter intervjuet med klienten bestemte vi oss for å grave litt dypere ved å gjennomføre to intervjuer der vi pratet med personer som hovedsakelig driver med engroshandel av klær, begge bruker papirkataloger og kan være potensielle brukere av systemet og produktkatalogen. I intervjuene prøvde vi å få frem hva slags informasjon som er viktig og hvordan de ser gjennom og bestemmer seg for produktene. Vi var også var interesserte i å se hvordan papirkatalogene presenterer produkter.

Intervju #1

Den første personen vi intervjuet driver en liten bedrift og handler produkter to-tre ganger i måneden. Hun driver også med trykk og reklameartikler.

Da vi pratet med den første personen ville vi først vite hva hun så etter når hun valgte hvilke klær hun skulle kjøpe. Hun sa det at hun setter seg ned med kunder for å se gjennom kataloger, hvis kunden ser noe de liker så kjøper hun flere av det produktet for å se om det selger. Hun driver også med en del trykk og derfor kjøper hun ofte ensfargede t-skjorter o.l. og trykker det kunden selv ønsker å ha på.

Vi pratet litt om bilder i katalogene og hun sa det var en selvfølge at det er gode bilder. Hun sa hun aldri har kjøpt et produkt uten å ha sett hvordan det så ut først, selv om det har skjedd at hun har kjøpt feil produkt. Det betyr ikke så mye for henne om produktene er på en person eller om det bare er produktet, men at ofte når kunder ser på produktene så vil de helst se bilde av en person med produktet på seg. I følge henne så gir det bedre inntrykk av produktet.

Hun virket også som hun var veldig bevisst på hvilke merker som har god kvalitet og hun holder seg helst til dem. Det kom også frem at hun syntes det var vanskelig å handle nye merker fordi størrelsene ofte er så forskjellige.

Med dette intervjuet fikk litt bedre forståelse på hva en potensiell bruker er opptatt av når den går gjennom en katalog. Vi fikk bekreftet hvilken informasjon som er viktig å ha i en katalog. Takket være intervjuobjektet fikk vi også mulighet til å se hvordan en papirutgave ser ut.

Intervju #2

Den andre personen vi snakket med bruker også papirkataloger, men har prøvd litt forskjellige løsninger. Derfor hadde vi lyst til å spørre ham om hvordan prosessen er. Han mener at å bestille på den måten er tidkrevende, men at det ble lettere med forskjellige digitale løsninger for å fylle inn og bestille produktene. Som regel må man bla gjennom mange sider med produkter og samtidig huske å notere produktkoder i bestillingskjema online. Vi fant ut at å gjøre det på den måten kan ofte føre til en del feil. Han går ofte gjennom katalogen flere ganger for å være sikker at han har valgt riktig produkt.

Under intervjuet snakket en del om løsningen vi skal lage. Vi ville høre med han om hva som kunne gjøres for å gjøre en webbasert løsning brukervennlig. Det han sa med engang var at hele opplevelsen hadde vært bedre hvis han kunne fokusere seg på å lete etter produktene

han vil ha i stedet for å bruke mye tid på bestillingsprosessen. Han har også prøvd kataloger i PDF format, men mener at hele dokumentet var presset sammen og rotete.

Intervjuene bekrefter at det finnes behov for en løsning som klienten vår utvikler. Hele prosessen med å gjennomføre handel blir enklere hvis det finnes et system som kan presentere produktene oversiktlig med mulighet for å filtrere og som eliminerer sjansen for å bestille feil produkt. Med informasjon vi samlet fra intervjuene kunne vi begynne å lage skisser og utvikle en prototype for katalogen.

4.2. Første prototype

For å produsere en prototype benyttet vi oss av fire steg. Først så analyserte vi konkurrenter og lignende løsninger for å få inspirasjon. Deretter tegnet vi opp papirskisser og diskuterte oss frem til et ønskelig utgangspunkt. For tredje steg produserte vi selve prototypen i en webapplikasjon. Til slutt avsluttet vi med å få tilbakemelding på prototypen.

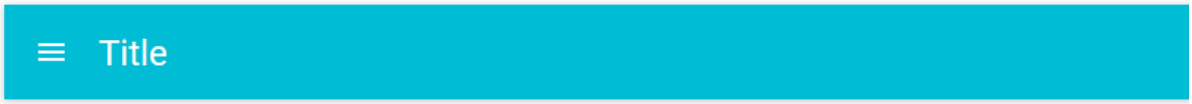
4.2.1. Konkurrenter

Ved å se på lignende løsninger kan vi se hvordan dagens marked ser ut for å få inspirasjon til vår katalog. Etter intervjuene snakket vi med klienten og fikk tilsendt linker og litt inspirasjonsbilder fra konkurrenter og lignende løsninger. Dette gjorde at vi fikk bedre oversikt over hvordan andre digitale kataloger ser ut og ikke bare papirkatalogene fra intervjuene. Vi hadde ikke direkte tilgang til noen slike kataloger, men vi så på videoer og skjermbilder som viste frem katalogene. Vi så med en gang at det var stort fokus på at designet skal se smakfullt ut. Vi fikk en generell idé om hvordan de viser frem produktene, alle de vi så på brukte en form for “grid” med bilde og litt relevant tekst, som ga oss en følelse på hvilken informasjon som vil være relevant å vise i oversikten av produkter og hvilken informasjon som kan spares til en “produkt-side”. Det var en katalog vi så på som vi syntes så rotete ut, mest fordi det virket som de prøvde å få alt for mye informasjon inn på skjermen samtidig.

Vi diskutere det vi hadde observert og kom frem til at vi skulle vise produktene i en grid, sortering og filtrering skulle kunne gjemmes bort og at vi skulle ha en for "produkt-side" for å vise mer detaljert om hvert produkt.

4.2.2. Skisser

Vi startet med papirskisser der vi startet med å tegne opp hvordan vi ønsket at de tre store komponentene skulle vises, sortering og filtrering, produktene og ett enkelt produkt (Vedlegg E). Siden vi visste vi måtte jobbe med Google Material Design kom vi frem til å bruke menylinjen definert i Material-UI og ha filtrering og sortering i en vertikal meny som åpner seg på venstre side, hvis brukeren trykker på hamburger-tegnet i venstre hjørnet som vist i figur 1.



Figur 1: Viser hvordan en Material-UI meny ser ut

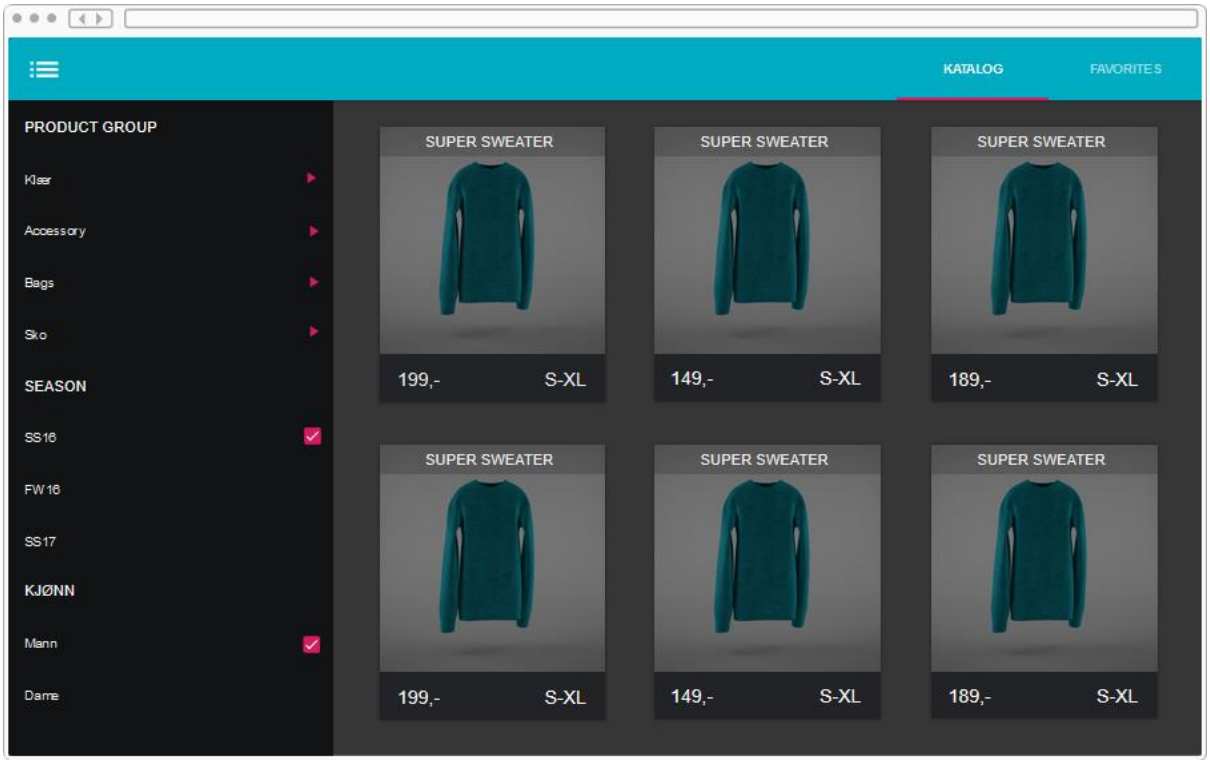
Vi var allerede sikre på at produktene skulle vises i en grid med x antall produkter i forhold til skjermstørrelse, vi tenkte å vise produktene på denne måten for å gjøre at det blir enklere generere dem ut senere. Videre falt diskusjonen naturlig på hvordan hvert enkelt produkt skal vises i visning av produkter. Vi tegnet opp litt forskjellige utgaver av disse produktene. Selv om alle skissene lignet på hverandre er plassering av tekst og bilde og diverse funksjoner forskjellig. Naturlig nok bestemte vi oss for å gå for et enkelt alternativ for å produsere den første prototypen. Ved å klikke på ett av disse produktene skulle det åpnes en skuff på høyre side som viste all relevant informasjon om dette produktet. Det ble også spesifisert at når enten sortering eller et produkt er åpent, så skal resten av siden få en mørk skygge over seg for å ta fokus bort fra resten av siden. Da vi hadde tegnet de første skissene og diskuterte dem, satt vi fokus på å lage en prototype.

4.2.3. Prototype

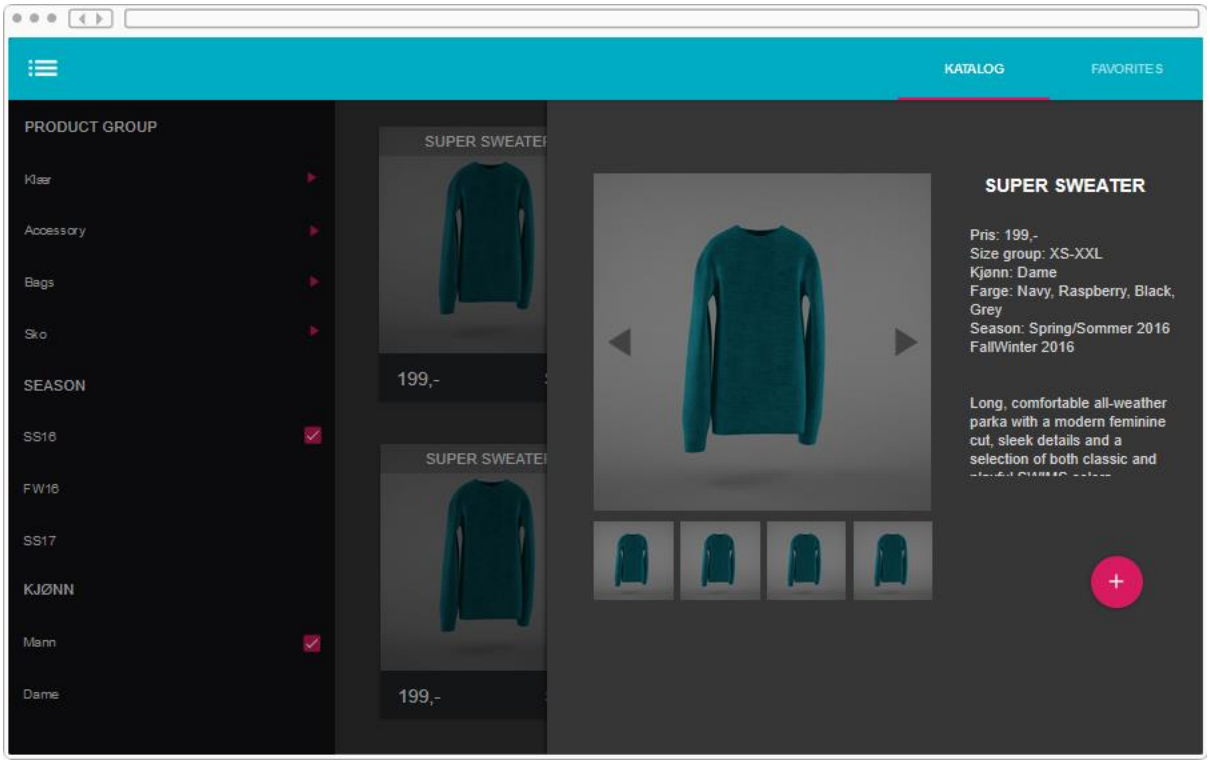
Å produsere prototypen var neste steg. Vi valgte gå for en enkel prototype[26] istedenfor wireframes for å enkelt vise hvordan produkter blir vist og fordi klienten hadde to hovedfarger på dette tidspunktet, så vi gjorde slik at vi med ett klikk byttet disse for å gjøre det enklere å velge en farge å ta utgangspunkt i. Det var heller ikke mye ekstra arbeid å lage en prototype fremfor wireframes pga. antall sider. Vi valgte å gå for et prototype-verktøy som heter proto.io da dette ble anbefalt av klienten. Proto.io lignet på andre prototype-applikasjoner som vi har brukt før, derfor trengte vi bare se på applikasjonen for å skjønne hvordan de viktigste funksjonene fungerte.

I den første prototypen viser vi hvordan vi tenkte at filtrering og sortering skal ligge nedover, med mulighet for å åpne og lukke forskjellige kategorier, figuren under viser hvordan "Klær", "Accessory" osv. har en liten pil som kan åpnes for å gjøre mer detaljert filtrering, slik at vi reduserer mengden informasjon som blir vist samtidig. Vi tenkte også å bruke sjekkbokser slik at brukeren enkelt kan se hva det er blitt filtrert ut ifra. Figur 2 viser hvordan vi ønsket å vise produktene med bilde, tittelen på produktet i toppen og litt mer relevant informasjon som pris, størrelser, farger osv. under bildet. Produkt-siden kommer ut som en skuff og viser "hoved-bildet" stort, med flere bilder av samme produkt under som vist i figur 3. På høyre siden av bilde kommer den all informasjon som brukeren ønsker å se. Vi tenkte også å bruke en knapp for å legge produktet til som favoritt. Resten av siden blir også mørkere når produktet vises slik at fokus blir tatt bort fra resten av siden. Da prototypen var ferdigprodusert kunne vi få tilbakemelding på prototypen.

[26] https://shabbyabby.proto.io/share/?id=b04791d2-60aa-4152-9303-515275c5c189&v=1



Figur 2: Screenshot av produkter og filtrering tatt fra første Prototype



Figur 3: Screenshot av et produkt tatt fra første prototype

4.2.4. Testing/Tilbakemelding

Vi fikk hentet en link til prototypen slik at vi kunne teste og få tilbakemelding. Først tok vi kontakt med klienten og ba dem se på prototypen. Responsen vi fikk var (hovedsakelig) positiv, det var noe slikt de var ute etter og de var fornøyde med resultatet. De ønsket at vi skulle ta utgangspunkt i dette for når vi skal starte å kode katalogen. Vi skulle komme tilbake og prate mer om hvilken informasjon som skal vises om produktene når dette blir mer relevant senere i prosjektet. Foreløpig skulle vi bruke den informasjon vi selv tror er relevant. Vi tok også å kjørte en liten test der testpersoner klikket seg rundt, men vi fikk veldig lite ute av dette fordi prototypen hadde få funksjoner. Dette var ikke problematisk ettersom prototypen hovedsakelig skulle fungere som en layout.

4.3. Konklusjon

Ved å tegne opp skisser og diskutere oss frem til hvordan vi ønsker at de forskjellige komponenten skulle se ut, ble vi enige og fikk samlet tankene på papir. Slik at vi satt med de samme tankene om hvordan komponenten skulle se ut. Ved å produsere prototypen fikk vi sett hvordan komponentene så ut når de var plassert sammen. Samtidig som det førte diskusjonen videre slik at begge ble fornøyd med resultatet. Tilbakemeldingen fra klienten ga oss bekreftelse på at vi var på riktig vei slik at vi var sikre på at vi hadde laget et godt grunnlag. Med prototypen ferdig og tilbakemeldingene notert kunne vi fortsette med å sette opp det tekniske slik at vi kan starte å kode.

5. Implementasjon

I dette kapittelet skal vi gå gjennom hvordan vi har utviklet katalogen, hva som ble gjort gjennom sprintene og samtidig diskutere resultatene. Vi tar også for oss forskjellige problemstillinger vi har møtt gjennom prosessen og hvordan klarte vi å løse disse.

5.1. Forberedelser

Vi hadde kjennskap til JavaScript og Google Material Design før vi startet prosjektet, men vi hadde aldri brukt React, Material-UI eller NPM før. Vi satt av tid i starten av prosjektet mens vi arbeidet med prosjektplanen til å begynne å lese om React. Det vi ikke visste var at for at React skal fungere ordentlig trengte vi å bruke endel pakker fra NPM. Derfor ble vi nødt til å lese oss opp og se utallige videotutorials om React og eksperimentere for å få alt til å fungere skikkelig. En av grunnene som har også gjort at vi måtte benytte NPM var at Material-UI rammeverket var tilgjengelig kun på NPM. Vi så på videoer, leste artikler og prøvde og feilet for å få React med Material-UI til å fungere. Dette har hjulpet oss med å forstå hvordan React fungerer og hvordan vi kan hente Material Design komponenter fra Material-UI. Da vi hadde fått alt til å fungere slik det skulle så satt vi med endel kunnskap om NPM og hvordan vi skal benytte det for å sette opp React og Material-UI. Tilslutt bestemte vi oss for å lage et nytt oppbevaringssted for å starte på katalogen, slik at vi kunne begynne første sprinten med et oppbevaringssted med bare det vi trengte for å få React og Material-UI til å fungere.

5.2. Første sprint

Da vi startet å programmere, var målet å legge grunnlaget for katalogen ved å få på plass de tre viktigste komponentene, fremvisning av alle produktene, fremvisning av ett produkt og skuffen til filtrering og sortering. Disse tre komponentene ble implementert ved å følge fire steg. Først implementerte vi en meny med mulighet til å åpne og lukke skuffen til filtrering og sortering. Deretter i steg to hvor vi skulle gjøre klar visningen av produktene ved hjelp av et datasett med dummy data. I steg tre gjorde vi slik at produktene ble klikkbare og at vi

hadde en dialog som åpnet seg med lik data som det produktet som ble valgt. Tilslutt gjorde vi litt testing for å se om alle funksjoner fungerte som vi ønsket.

5.2.1. Første utgave i React

For å produsere første utgaven av katalogen bestemte vi oss for å bruke dummy data. Vi hadde ikke fått noen JSON-fil av SIC så vi brukte dummy JSON data fra Material-UI sine nettsider[27]. Vi valgte bruke dette foreløpig for å forstå hvordan vi skulle jobbe med datasettet senere. De tre komponenter skulle også gjøres klare til senere bruk og gjøres slik at riktig data blir hentet ut når vi klikker på et produkt.

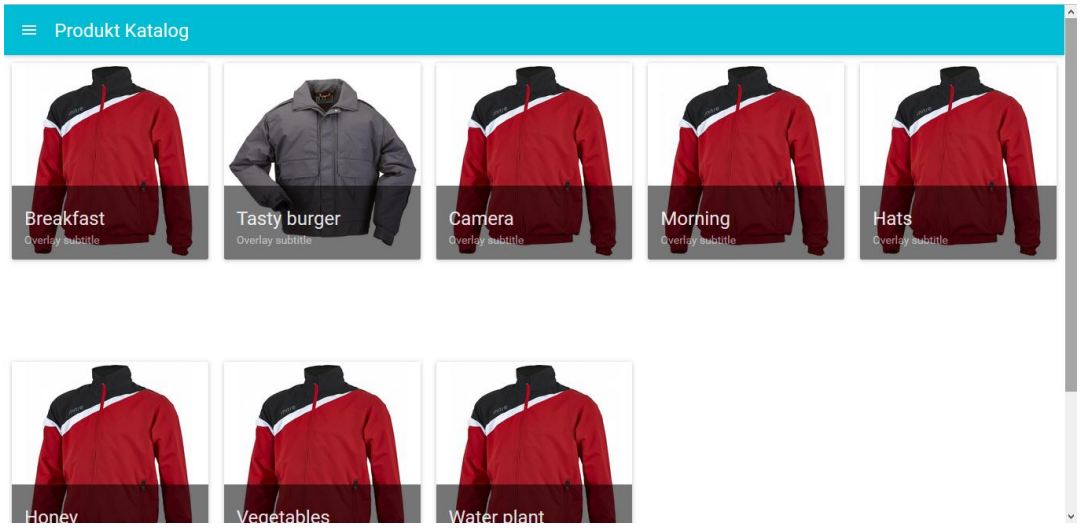
For å få implementert menylinjen måtte vi hente komponenten fra Material-UI sine nettsider. Vi kunne gjøre det samme med skuffen til filtrering og sortering, men vi hadde problemer med å få åpnet skuffen da vi hentet disse komponentene separat fordi de ikke hadde noen sammenkobling. Det ble også rotete fordi det ble mye ekstra koding å bruke disse som to komponenter. Siden skuffen skulle være en del av meny-linjen, fant vi ut at det var lettere å slå disse to komponentene sammen til en komponent. Da kunne vi sette opp en hamburgermeny-knapp i venstre hjørne som åpnet skuffen og lukket den når man klikker utenfor skuffen.

For å presentere alle produktene diskuterte vi hva slags Material-UI komponent vi skulle bruke. Valget falt mellom "Card"[28] eller "Grid List"[29]. Vi foretrakk "Card" fordi det er den som er mest gjenkjennbar hvis brukeren bruker Android eller Google og fordi den gir mulighet for å bli tilpasset senere. Vi valgte allikevel å gå for "Grid List" fordi det var det raskeste å gjøre på dette tidspunktet og det gjør slik at det blir flere arbeidsoppgaver. På den måten kunne vi dele arbeidet slik at vi både jobbet med å få en dialog til å komme frem å vise riktig data og å jobbe videre med produktene. Arbeidet videre på produktene gikk bedre enn forventet og vi

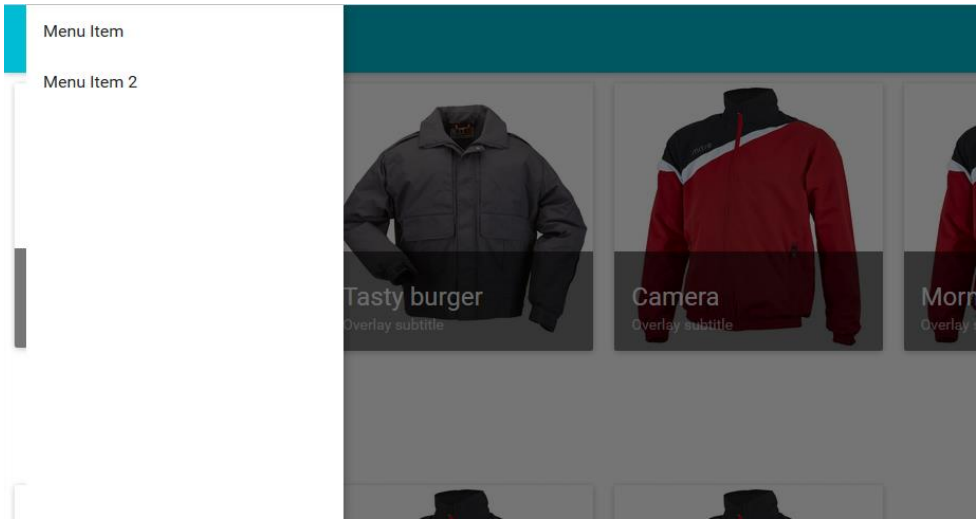
[27] http://www.material-ui.com/#/components/grid-list
[28] http://www.material-ui.com/#/components/card
[29] http://www.material-ui.com/#/components/grid-list

fikk enkelt gjort slik at den viste riktig mengde produkter i bredden og at de hadde god størrelse, samt at vi fikk satt opp dummy bilder til hvert produkt slik at vi kan se hvordan tekst og bilde la seg i forhold til hverandre. Fordi dette gikk raskere enn forventet rakk vi å integrere "Cards" inn i koden til "Grid List" og tilpasset det slik vi ønsket. Å få en dialog til å åpnes var ikke i utgangspunktet noe problem, men vi hadde problemer fordi vi måtte gjøre slik at det var forskjellige dialoger for hvert produkt. På et tidspunkt hentet den ut alle dialogene samtidig og på et annet fungerte det, men da kunne vi ikke åpne mer enn en dialog før vi måtte oppdatere siden. Vi fikk tilslutt gjort slik at dialogen åpner og lukker seg skikkelig og at den viser navnet på det produktet som ble valgt.

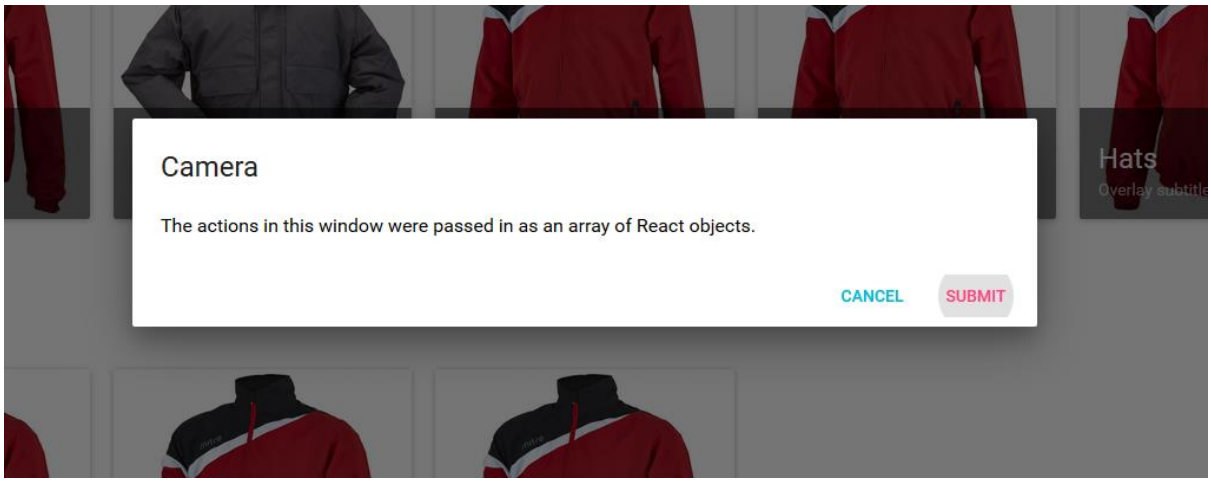
I figur 4 viser vi hvordan produktene blir generert ut og viser produktene fra JSON-fila. Hvert bilde blir generert ut ved hjelp av navn i JSON-fila. Menyen er standard Google Material Design, og har en hamburgermeny som åpner en skuff tiltenkt sortering og filtrering som vist i figur 5. Når et produkt blir klikket på åpnes en dialog som viser samme navnet som produktet som har blitt klikket på. I figur 6 har vi klikket på det tredje produktet "Camera" og dialogen viser tittelen "Camera" slik at vi også enkelt kan hente mer data fra produktet. Fargevalget foreløpig har vi valgt å bruke det som er i standard Material-UI til vi har blitt enige med Sic om en fargepalett.



Figur 4: Første utkast av katalogen laget med React og Material-UI, hvor vi satt opp meny-linjen og generer produkter fra en dummy data



Figux 5: Første utkast av katalogen, viser hvordan sortering og filtrering vil se ut.



Figur 6: Første utkast av katalogen, viser produkt-siden. Her klikket inn på produkt nr 3. Camera fra figur 4.

Det første vi produserte av katalogen var oversikt over produkter med bildet og litt tekst. Vi hadde laget grunnlaget for resten av applikasjonen ved å sette opp å gjøre klar de tre viktigste komponentene, skuff til sortering og filtrering, vise frem produktene og en dialog som får data ut ifra hvilket produkt som blir klikket på. Selv om både tekst og bilder bare er dummy data valgte vi å kjøre en liten test på katalogen.

5.2.2. Testing

Da vi hadde lagd den første utgaven, valgte vi å gjøre testing ved å la noen personer klikke seg rundt litt å se om de finner de forskjellige tingene og klarer lukke dem igjen. Test personene skjønte at de kunne klikke på produktene med engang og de klarte lukke dialogen. Test personene var delt når det gjaldt å finne filtrering. Halvparten klikket på hamburger-menyen og resten trodde de hadde sett hele katalogen. Vi fant ut at vi måtte på et tidspunkt gjøre noe med filtrering og sortering slik at den blir mer synlig for bruker. Etter vi var ferdig med testing, kunne vi fortsette med å lage andre utgaven av katalogen.

5.3. Andre sprint

For den andre utgaven av katalogen bestemte vi oss for å gjøre katalogen klar til den siste sprinten. Først skal vi erstatte dummy dataen vi har brukt med reelle produkter i datasettet. Produktene skal formateres og vise frem riktig data, vi skal også hente ut et bilde per produkt. Vi skal gjøre klar noe filtrering og sortering slik at implementering eller fjerning av funksjoner kan bli gjort raskt senere. Dialogen som viser frem et produkt må bli forandret slik at den viser tekst og bilde fint sammen. I tillegg må vi passe på så den henter ut riktig data til hver dialog. Tilslutt endrer vi fargepalletten fra et lyst tema til et mørkt etter ønske fra klienten.

5.3.1. Datasett

Når vi satt i gang arbeidet med å klargjøre produktkatalogen benyttet vi oss av en JSON-fil fra klienten. Derfor var det naturlig å erstatte dummy dataen vi hadde tatt fra Material-UI sine nettsider og fortsette med dette nye datasettet. For at datasettet skal vise produktene må vi kode den til å gå gjennom hvert produkt og hente ut all relevant data. Vi var fortsatt ikke helt komfortable med å jobbe i React og strukturen på datasettet var helt nytt for oss. Enkle ting som for eksempel en for loop [30]var vanskelig å bruke på riktige steder, og vi brukte

[30] **For loop** - en funksjon som går gjennom hver objekt i et array

lang tid på å få data til å vises riktig. Etter datasettet var oppdatert med relevant data startet vi prosessen med å vise frem alle produktene.

5.3.2. Produktene

Oversikten over produktene er det første som blir vist og vi arbeidet videre med å gjøre slik at hvert produkt viser frem riktig data, men i oversikten over produktene ble det ikke gjort så mange forandringer. Vi måtte dog endre litt på strukturen for at teksten skulle hentes ut fra det nye datasettet. Den eneste forandringen vi måtte gjøre var å endre variabelen som skulle inneholde produktnavnet som vist i figur 7. I den gamle JSON-fila var navnet strukturert som for eksempel "title: 'produkt navn'" og i den nye er det strukturert slik "description: 'produkt navn'". For å endre hva som blir sendt videre til dialogen måtte vi gjøre akkurat det samme, erstatte "title" med "description".

```
overlay={<CardTitle title={tile.title} subtitle="Overlay subtitle" />}
overlay={<CardTitle title={tile.description} subtitle="Overlay subtitle" />}
```

Figur 7: Rød linje ble erstattet med grønn. Vi trengte bare endre .title til .description for at det nye datasettet skulle vise nye produkter

Vi fikk også tilgang på dummy bilder fra klienten som vi brukte og ga et bilde til hvert produkt og ga den likt navn som produktkoden. På denne måten får hvert produkt et eget bilde og katalogen blir med en gang mer realistisk. Dette gjorde også slik at vi hadde bilder å bruke i dialogen. Etter produktene var oppdatert til å vise data fra det nye datasettet gikk vi videre med å utvikle kode for filtrering og sortering.

5.3.3. Filtrering og Sortering

Filtrering og sortering er grunnleggende funksjon som fasiliterer søking og navigering av store produktkataloger. I dette prosjektet ble utviklingen av filtrering en lang prosess. Selv om selve filtrerings funksjonen kunne skrives i ren JavaScript, fordi den ligger i en separat fil. Det startet med litt prøving og feiling men tilslutt fant vi ut at det kan løses ved en enkel funksjon. I figur 8 viser vi hvordan den henter ut alle produkter som har navnet "McGregor". Den går gjennom hvert produkt i datasettet, lager et nytt datasett med det/de produktene

som matcher filtreringen, deretter bruker den det datasettet til å vise frem produktene. Da første funksjonen var laget, var det enkelt å sette opp filtrering for kjønn og vi hadde grunnlaget for å sette opp resterende filtrerings funksjoner.

```
+ var dataSet1 = dataSet.filter(filter);
```

```
+ function filter(a) {
+     return a.description == 'McGregor';
+ };
```

```
+ function filterWomen(a) {
+   return a.gender == 'WOMEN' || a.gender == 'UNISEX';
```

Figur 8: Første linje viser hvordan den henter inn filtrerings funksjonen vist under. Andre linje henter ut alle som har navnet McGregor og siste henter alle produkter for kvinner og "unisex"

Videre måtte vi ta en avgjørelse på hvordan filtreringen skulle fungere med farger. Skulle vi automatisk generere ut alle fargene eller lage kategorier av farger. Vi bestemte oss for sist nevnte. Hvis vi skulle generert ut en lang liste med farger, vil den bruke vanvittig mye plass og endel farger har navn som ikke nødvendigvis er kjent for brukeren. Som for eksempel at "Pale Petrol" er en blå farge. Derfor valgte vi å gå for å lage kategorier. Da tok vi ut fargene fra datasettet og plasserte dem i kategorier som "blå", "grønn" og "rød". Dette førte til at vi først måtte finne alle de forskjellige fargene i datasettet også kategorisere dem. Deretter satt vi opp en funksjon for hver kategori, figur 9 viser hvordan filtreringen ser gjennom alle fargene til et produkt av gangen og deretter ser om noen av dem matcher en av blåfargene. Hvis noen av dem matcher så sender den det produktet tilbake til et nytt datasett, hvis et produkt ikke matcher så sender den ingenting til det nye datasettet og den blir ikke med når produktene blir generert ut igjen. Et problem på dette tidspunktet var at vi kunne sjekke om funksjonen fungerte, fordi vi ikke hadde hentet ut fargene noe sted enda, derfor måtte vi manuelt gjøre stikkprøver ved å sjekke gjennom JSON-fila for å se om filtreringen hentet ut riktige produkter. Da vi hadde foretatt noen stikkprøver og var sikre på at de riktige produkter ble vist og vi kunne fortsette med å klargjøre presentasjonen av et enkelt produkt.

```
function filterColorBlue(a) {
    var n = a.colors.length;
    for (var i=0; i < n; i++){
        if (a.colors[i].description == 'Regatta Blue'||
            a.colors[i].description == 'Navy' ||
            a.colors[i].description == 'Cobalt Blue'||
            a.colors[i].description == 'Teal Blue'||
            a.colors[i].description == 'Pale Petrol'){
            return a;
        }

    }
};
```

Figur 9: Funksjon for å filtrere blåfarger, hvis produktet har en av blåfargene vist her så returner den dette produktet

5.3.4. Produktsiden

For at brukeren skal kunne få mer informasjon om et produkt må relevant informasjon være presentert på en god måte. Men før vi kunne starte å arbeide med å vise produktsiden, var det naturlig å begynne med å diskutere om vi skal fortsette å bruke dialogen. For øyeblikket var ikke dialogen i nærheten av å se slik ut som vi ønsket, så det å bytte Material-UI komponent ville ikke vært mye ekstra arbeid. Vi bestemte oss allikevel for å bruke dialogen og å formatere den slik at den ser slik ut som vi ønsket, fordi funksjonaliteten på dialogen var nøyaktig det vi ville ha. Den lukker seg når vi klikker på utsiden også er den lagd slik at det er plass til knapper i nede i høyre hjørne. Det betydde derfor at vi ble nødt til å endre formateringen på dialogen. Formateringen ble gjort som CSS via React og syntaksen var veldig lik. Dialogen ble gjort større og styling til bildet og tekst ble satt opp.

Etter vi fant ut hvordan vi kan endre på utseende av dialogen, var det på tide å hente ut data fra datasettet, slik at vi kunne vise fram forskjellig data om produktet på oversiktlig måte. Det første vi begynte å jobbe med var å hente ut alle størrelsene og alle fargene til produktet. Til dette bestemte vi oss for å bruke en dropdown meny. Det vi brukte mest tid på var uthenting av data. Her måtte vi bruke en for loop for å hente ut alle fargene og størrelsene som tilhører et produkt. Funksjonen i seg selv vil hente alle farger som finnes i datasettet, men poenget var å hente ut farger for produktet som var åpnet i dialogen. For å

få det til måtte vi bruke en if statement[31] som går ut på at hvis produktnavnet som ble åpnet i dialogen er lik med et navn som finnes i datasettet vårt, så skal skriptet kjøre en for loop for å hente alle fargene og størrelsene for det produktet som vi fikk treff på i if statmenten. Hele prosessen kan vi se under på Figur 10.

```
var rows = [];
var numrows = dataSet.length;
for (var i=0; i < numrows; i++) {
    if (this.state.tileItem == dataSet[i].item) {
        var colrows = dataSet[i].colors.length;
        for (var j=0; j < colrows; j++) {
        rows.push(<MenuItem value={dataSet[i].colors[j].description} primaryText={dataSet[i].colors[j].description}/>)
        }

    }
}
```

Figur 10: For loop som går gjennom datasettet og deretter finner produktet som er åpnet i dialogen. Tilslutt går den gjennom alle fargene i dette produktet og skriver disse til arrayet "rows"

For å vise resultatene av koden, bruker vi en funksjon som heter "push". Denne definerer hva som blir vist som resultatet av for loopen. På Figur 10 kan vi se at vi skriver ut variabelen rows som ble definert som et tomt array før for loopen. Med push legger vi resultatene av vår for loop til i arrayet rows. I denne situasjonen er det alle fargene til produktet som ble åpnet i dialogen. Etter testing av resultater i konsollen, fant vi ut at koden har gjorde det vi ønsket. På figur 11 viser vi hvordan vi henter dropdown menyen ved å skrive {rows}. Da produktsiden var ferdigstilt for denne sprinten gikk vi videre til å bytte fargetema.

```
<SelectField
    value={this.state.value}
    onChange={this.handleChange.bind(this)}
    floatingLabelText="Color">
    {rows}

</SelectField>
```

Figur 11: Material-UI komponent for dropdown-meny som blir brukt for å vise alle tilgjengelig farger for et produkt.

[31] **If statement** - et utsagn som er true eller false. Hvis true så funksjonen vil gjøre noe

5.3.5. Endring av farger

Klienten bestemte seg for å endre fra lyse farger til mørke farger i systemet sitt og vi bestemte oss derfor for å gjøre det samme da dette var ønskelig. Vi vurderte å fikse dette problemet på to måter. Den første som vi er kjent med fra før, er å endre farger ved hjelp av CSS. Enten ved å benytte en standard .css fil som blir inkludert i HTML filen, eller ved å formatere elementer med CSS ved hjelp av React. Disse metodene hadde blitt tidskrevende fordi vi måtte ha tilpasset alle fargene for alle elementene som finnes i katalogen. Den siste måten og måten vi har valgt for å løse dette, var å endre standard fargepalett til mørk ved hjelp av Material-UI. Material-UI har to innebygde fargepaletter, en lys og en mørk. Den første er satt opp som standard når man henter Material-UI komponenter. Som vi ser på figur 12, så måtte vi først hente ut en fil der den mørke fargepaletten er definert. Filen er levert som standard i Material-UI, derfor trengte vi bare å importere filen til main.js. Når vi hadde denne på plass, kunne vi definere "darkMuiTheme" som skal hente den mørke fargepaletten slik at den kan bli brukt på komponentene vi har lagd. På figur 13 ser vi hvordan vi har satt opp mørke farger på dialogen. Samme løsningen ble brukt for å implementere mørke farger på andre komponenter.

```
import darkBaseTheme from 'material-ui/lib/styles/baseThemes/darkBaseTheme';
import MuiThemeProvider from 'material-ui/lib/MuiThemeProvider';
import getMuiTheme from 'material-ui/lib/styles/getMuiTheme';

const darkMuiTheme = getMuiTheme(darkBaseTheme);
```

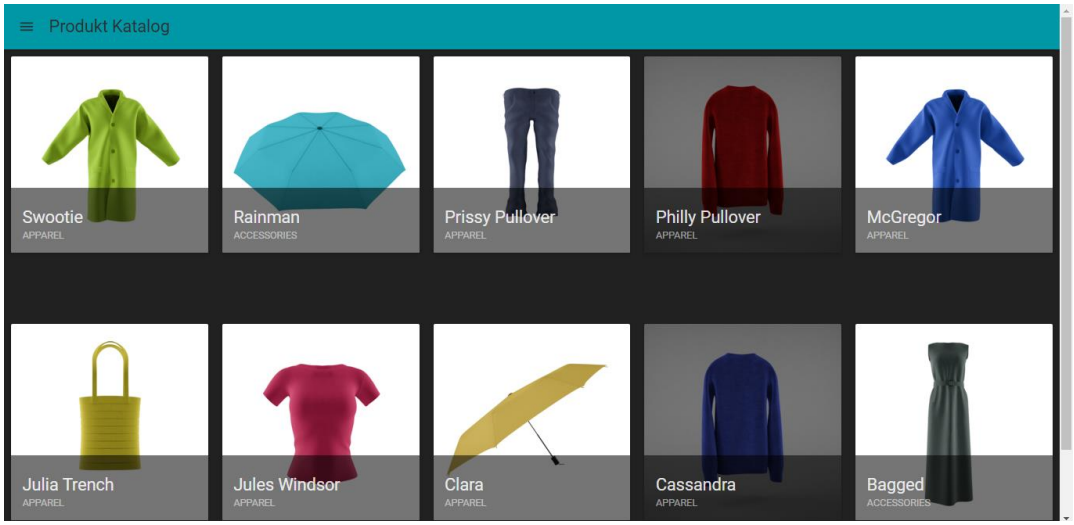
Figur 12: Importerer mørk fargepaletten med nødvending funksjoner for å ta den i bruk.

```
<MuiThemeProvider muiTheme={darkMuiTheme}>
<DialogExampleSimple/>
</MuiThemeProvider>,
```

Figur 13: Viser hvordan mørke farger ble satt opp på dialogen.

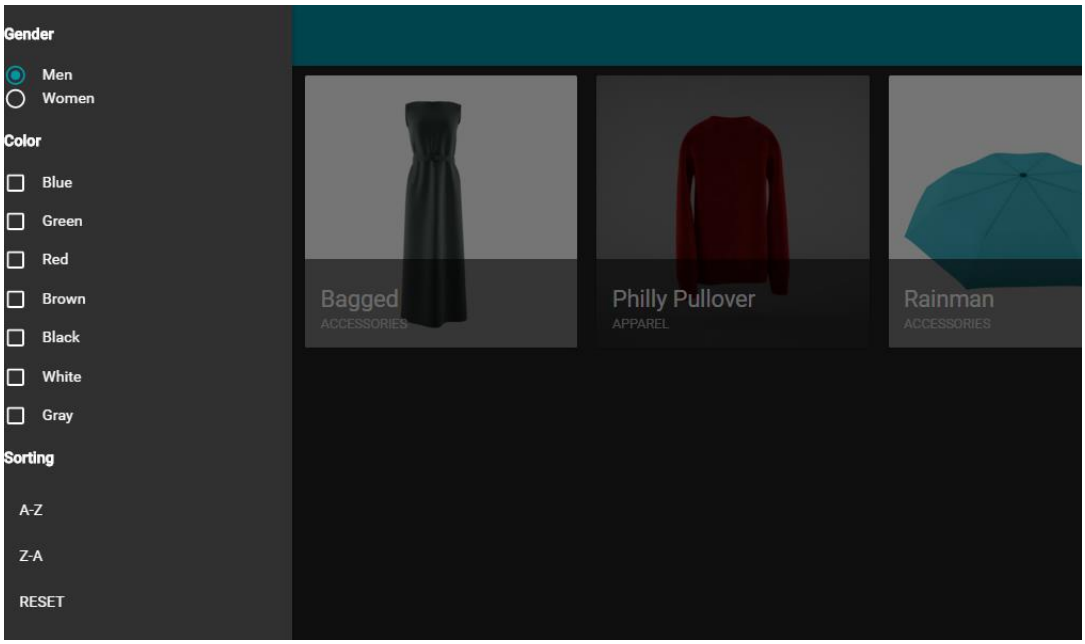
5.3.6. Andre utgave

I dette kapittelet presenterer vi den andre utgaven av katalogen. Figur 14 viser hvordan produkt kortene[32] ikke er mye forskjellig fra første utgave. Bildene er forskjellig fra tidligere og fra hverandre, og vi har prøvde å ha et par bilder med mørkere bakgrunn for å se hvordan det ser ut. Teksten på bildene er nå reelle produkter og teksten under produktnavnet viser nå hvilken kategori produktet tilhører. Hamburger menyen åpner skuffen som inneholder sortering og filtrering og figur 15 viser hvordan denne ser ut. Det går både an å filtrere på farge og på kjønn samtidig og/eller med flere farger. Fargene er redusert til noen kategorier slik at det ikke blir nødvendig med scrolling. Ved å klikke utenfor skuffen vil produktene oppdatere seg etter hva som har blitt sjekket av for. Produktene kan sorteres alfabetisk og det er en knapp for å tilbakestille filteret. Ved å klikke på produktet kalt Bagged åpner dialogen i figur 16 seg og tilsvarende dialog vil komme frem hvis et av de andre produktene blir klikket på. Denne dialogen viser navnet på produktet øverst med bilde av produktet med plass til flere bilder under seg. Produkttype og kjønn blir også hentet inn som ren tekst og vi genererer alle fargene i en dropdown slik at brukeren kan velge hvilken farge han ønsker. Denne siden kan også lukkes ved å klikke på utsiden av selve dialogen.

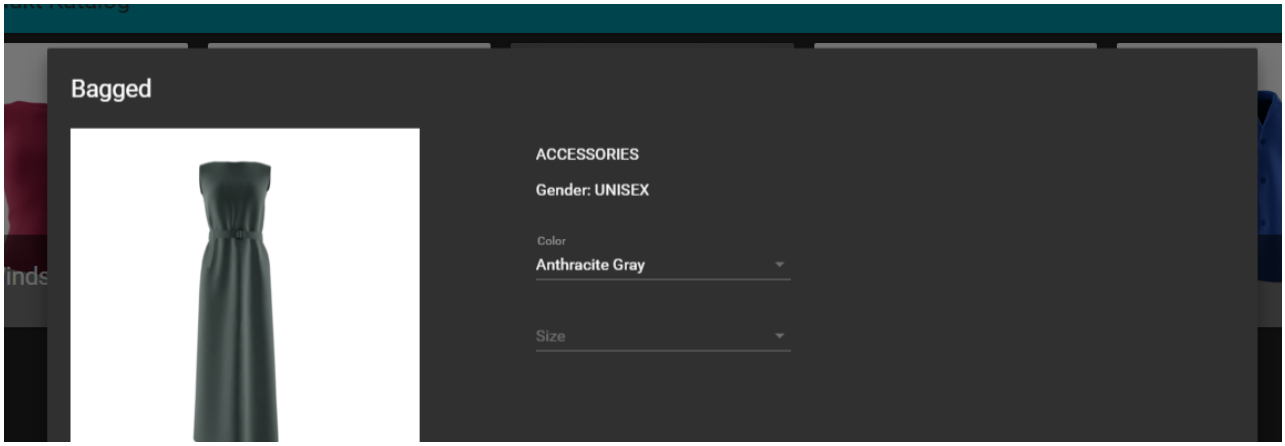


Figur 14: Andre utkast av katalogen, oversikt over alle produktene

[32] **Kort** - hvordan et produkt blir vist i liste over produkter. Navnet kommer fra Material komponenten Card



Figur 15: Filtrering og sortering, her filtrert med kun menn og unisex produkter



Figur 16: Produkt-Siden, viser produktets kategori, kjønn og en dropdown med alle fargene.

Prosessen for å komme til andre utgave var lang, men den hjalp oss virkelig forstå React bedre. Det gamle datasettet ble erstattet med ett vi fikk fra klienten. Produktene fikk nye bilder og nye navn ved å endre en liten kodebit. Vi tok et skritt av gangen og fikk på plass filtrering og sortering, slik at hvis det trengs flere eller andre filtrering funksjoner så kan vi enkelt sette opp dette. Dialogen ble gjort presentabel med ny formatering, slik at bilde og tekst legger seg fint i forhold til hverandre. Samtidig fant vi ut hvordan vi henter ut data fra

datasettet til dialogen, som har gjort arbeidet videre i prosjektet enklere og raskere. Med den andre utgaven ferdig bestemte vi oss for å besøke klienten i Oslo for å vise den andre utgaven og diskutere ting som vi var usikre på slik at vi kunne begynne tredje sprint.

5.4. Tredje sprint

Da vi ble ferdig med den andre sprinten, hadde vi alle hoved funksjonen på plass. For å begynne med den tredje sprinten måtte vi å få tilbakemelding på den andre utgaven. Derfor avtalte vi et møte med klienten for å få tilbakemelding, men også for å diskutere problemer og hva som med i det ferdige produktet. Etter møte med klienten måtte vi først erstatte datasettet for å kunne vise produktene på en ny måte. Vi må også gjøre klar filtreringen og sorteringen klienten ønsker og flytte den inn i selve katalogen. Produktkortene må oppdateres med riktig informasjon også skal de få forskjellige bilder ut ifra farge. Tilslutt skal vi oppdatere måten et produkt vises.

5.4.1. Besøk klienten

Før den siste innspurten dro vi til Oslo for å møte med klienten. Poenget med møte var å få tilbakemelding på den andre utgaven, få hjelp med diverse programmering og få vite hva som må med i den siste utgaven av katalogen. Vi begynte med å diskutere fargene som ble brukt i katalogen. Klienten var fornøyd med at vi har valgt å bruke mørke farger, som de ønsket. De ga oss også beskjed om at ikke vi trenger fokusere mer på farger, fordi de vil tilpasse fargene i katalogen senere når de har bestemt seg for en fargepalett som skal brukes i hele systemet. Det var viktigere for dem at vi fokuserer på funksjonalitet og uthenting av informasjon til forskjellige elementer i katalogen.

På starten pratet vi om hvilken informasjon som skal hentes ut til produkt kortene og dialogen. Vi fant ut at det var viktig å ikke ha med alt for mye informasjon i kortene, men også at det nødvending å få med den viktigste informasjon som er produktnavn, farge og pris. Det vil gi brukere rask oversikt over priser og farger på forskjellige produkter med engang, uten å måtte trykke seg gjennom hvert produkt. For å få mer informasjon om produktet, har vi blitt enige med klienten at det dialogen er best til det. Her får brukeren vite

mer om produktet som for eksempel hvilke farger og størrelser som er tilgjengelige. Gjennom diskusjon med oppdragsgiver fant vi ut at vi trenger å hente ut hvilke sesonger produktet tilhører, pris og størrelser til dialogen. Vi har også avklart at vi skal legge av plass til beskrivelse av produktet, slik at vi slipper å legge til beskrivelse for hvert produkt i JSON fila. Det var et ønske om å legge til en favoritt knapp i stjerne form, en knapp for nedlasting av .csv filer og knapp for å legge produktet i en bestilling. Vi fikk beskjed om at ikke vi trenger å tenke på funksjonaliteten til knappene siden disse blir satt opp senere når katalogen blir implementert i Datorder. Etter dialogen har vi sett på hamburgermenyen, hvor filtrering og sortering ligger. Det ble bestemt at den skal flyttes inn på venstre siden av selve katalogen, hvor produktkortene ligger. Selve hamburger menyen kommer til å bli brukt i Datorder for å navigere seg gjennom resten av systemet. Fordelen med dette er at sortering og filtrering blir mer tilgjengelig når disse funksjonene er en del av selve katalogen og ikke gjemt bort.

Vi tok også opp hvordan sortering og filtrering skal fungere. Klienten ønsker å kunne filtrere produktene på sesong. Sesong er viktig fordi kunden ofte ønsker å se på produkter eller handle ut ifra en spesifikk sesong. Derfor ble vi enige om å implementere dette. Videre falt diskusjonen på måten vi filtrerer på farger. Klienten likte tanken, men da vi så gjennom listen med potensielle farger, fant vi ut at å gruppere hundrevis av farger vil ta utrolig lang tid og det må også brukes mye tid på å vedlikeholde denne listen. Derfor kom vi frem til at det er bedre å kunne sortere fargene alfabetisk slik at like farger blir gruppert sammen.

Til slutt, fant vi ut at vi må bytte måten vi generer ut alle produkt kortene på. Klienten ønsket at i stedet for å genere hvert produkt ut ifra produktkoden, ville de at vi skal ha flere produkter ved å hente ut alle produkter med alle tilgjengelige farger. Det vil si at hvis en genser har to farger vil den vises som to produkter i katalogen. Denne endringen førte til at vi måtte hente data fra JSON fila på annen måte. I utgangspunktet hadde vi brukt map funksjonen (8), men ved bruk av denne funksjonen fikk vi ikke muligheten til å hente ut fargekoden eller fargenavnet som ligger inne i objekter som tilhører et produkt. På Figur 17 kan vi se at på en enkel måte kan vi hente ut navnet til produktet og produktkoden som blir

brukt for å hente bildet til produktet. Map funksjonen generer antall kort ut ifra antall produkter som finnes i datasettet vårt.

```
{tilesData.map(tile => (
    <Card label="Dialog" onTouchTap={this.handleOpen.bind(this)}>
        <CardMedia
            overlay={<CardTitle title={tile.title} subtitle="Overlay subtitle" />}
        >
            <img src={tile.item + ".png"} />
        </CardMedia>
    </Card>
))}
```

Figur 17: Funksjonen map (tilesData.map) går gjennom JSON fila og generer et kort per produkt. Kortene blir generert ut med en tittel (tile.title) og bilde (tile.item + ".png")

Siden funksjonen ikke kunne hjelpe oss med å hente ut informasjonen som vi trenger for å generere flere produkter, bestemte vi oss for å bruke en for loop. Takket være den var det mye lettere for oss å gjennomføre det klienten har ønsket seg. Når vi brukte for loop kunne vi gå dypere i datasettet. Vi måtte bruke flere loops slik at vi kunne komme oss videre til objekter med den informasjonen som vi trenger å ha med i kortene. På Figur 18 kan vi se at tilesData ble byttet ut med dataSet. Vi kan også se at vi bruker to loops for å gå gjennom alle produktene, deretter alle fargene som produktene har. Senere bruker vi informasjon som vi fikk henta ut med for loopene for å generere et nytt kort for hver produktfarge.

```
var card = [];
var cardrows = dataSet.length;
for (var z=0; z < cardrows; z++) {

    for (var x=0; x < dataSet[z].colors.length; x++) {
        card.push(<Card label="Dialog" onTouchTap={this.handleOpen.bind(this)}>
            <CardMedia
                overlay={<CardTitle title={dataSet[z].description} subtitle={dataSet[z].colors[x].description} />}
            >
                <img src={"img/" + dataSet[z].item + ".png"}/>

            </CardMedia>
        </Card>)
    }
}
```

Figur 18: For loops som henter ut produktene med fargene og genererer et kort per farge per produkt. Først går den gjennom alle produktene. Deretter går den gjennom hver farge til hvert produkt og genererer ut et kort.

Løsningen har hjulpet oss å gjennomføre endringene vi trengte å ha på plass og samtidig fikk vi bedre forståelse på hvordan vi kan navigere oss i JSON fila. Vi kan også utvide samme for loopen hvis vi trenger å hente informasjon som ligger enda dypere i datasettet vårt, som for eksempel størrelse eller pris for et produkt. Sakene svi tok opp på møtet hjalp oss med å få konkrete svar på ting vi lurte på, både rundt det teknisk, men også om selve designet. Sammen med klienten fant vi en løsning på hvordan vi skal generere produkt kortene ut ifra farger. Vi har lagd en plan på hva som må bli fikset for å gjøre produkt katalogen brukervennlig og oversiktlig. Vi fikk bedre oversikt over oppgaver som gjensto og hva vi må fokusere på for å bli ferdig med katalogen. Neste steg var derfor å starte med filtrering.

5.4.2. Filtrering og nytt datasett

Da vi begynte arbeidet igjen etter besøket med klienten startet vi med filtrering og sortering, men etter litt testing med den nye måten å vise produktene på, fant vi ut at dette ble problematisk. Sortering og filtrering på kjønn, produktnavn og annen informasjon som var data direkte fra produktet fungerte fortsatt, men når vi måtte dypere i datasettet for å hente for eksempel farge, da returner den hele produktet med alle fargene, istedenfor å kun returnere den ene fargen som matchet. Dette resulterte i at vi fikk at nytt datasett slik at filtrering og sortering skal kunne fungere, men det betydde også at vi måtte gå gjennom koden å endre slik at det nye datasettet kunne fungere.

Når filtreringen var inne i skuffen til venstre så var det når skuffen ble lukket, så oppdaterte produktene seg med all den relevante filtreringen og sorteringen som brukeren hadde valgt. Fordi vi nå hadde flyttet dette til inn i selve katalogen, ble vi nødt til å finne en ny måte å aktivere filtreringen på. Vi valgte løse det med bruk av knapper. Når knappene blir klikket på, så oppdateres produktene. Vi ønsket også å redusere antall sjekkbokser for at filtreringen skulle gå raskere.

Fordi filtrering og sortering ikke fungerer måtte vi diskutere med klienten og vi kom frem til at vi skulle få en ny JSON-fil med litt ny struktur. Når vi skulle filtrere og sortere på det gamle datasettet kunne vi kun sortere produktene, det vil da si at hvis noe filtrering fant for

eksempel en farge, returnerte den hele produktet og den viste frem dette produktet med alle fargene istedenfor kun en farge. Å sortere på fargenavn var også umulig siden den alltid vil viste et produkt med alle fargene før den gikk over til neste produkt. Vi kunne gjort slik at "produkt a" blir sortert med fargenavn, men det ville kun gjort at det blir enklere å lete gjennom for å finne en spesifikk farge i ett produkt, men ikke mange produkter med den fargen. Den nye JSON fila gjorde slik at datasettet inneholdt farger som inneholdt informasjon om et produkt hver, det vil si at det ble litt duplikat, fordi en farge kunne komme flere ganger, men de ble skilt ved at de hadde forskjellig produkt som et under-object. Ved å endre datasettet på denne måten ble vi nødt til å gå gjennom hele koden og endre noen linjer for at det skulle matche. Figur 19 viser hvordan vi måtte gå gjennom å gjøre små endringer i koden for at den skal vise riktig data.

```
dataSet[z].colors[x].description
+ ' ' + dataSet[z].colors[x].sizegroup
+ ' ' + dataSet[z].colors[x].sizes[0].prices['NOK'].listprice
+ ' ' + dataSet[z].colors[x].sizes[0].prices['NOK'].unitprice}
dataSet2[z].description
+ ' ' + dataSet2[z].sizegroup
+ ' ' + dataSet2[z].sizes[0].prices['NOK'].listprice
+ ' ' + dataSet2[z].sizes[0].prices['NOK'].unitprice}
```

Figur 19: Viser hvordan vi måtte endre litt i koden for at det skulle vise fra det nye datasetet. Her viser den hvordan vi byttet ut fargenavnet, "sizegroup" og "list-" og "unitprice", Røde linjer blir erstattet med grønne

Klienten ønsket å ha filtrering på sesongen til produktet. Et produkt kan ha flere sesonger og vi må ganske dypt i JSON-fila for å finne sesonger. Fordi vi må så dypt har hvert produkt, eller hver farge som katalogen nå viser, flere linjer med tekst som ser for eksempel slik ut "#SS15#FW15#SS16". For å få selve filtreringen til å fungere måtte vi velge mellom å generere alle sesongene i katalogen eller å fysisk lage en funksjon for hver sesong. Vi kom frem til å generere dem ut. Å lage en funksjon for hver sesong hadde gitt oss mer kontroll på hvordan det endelige produktet ser ut, men vi risikerer også at den viser sesonger som ikke eksisterer i katalogen eller at den ikke har en sesong som finnes i katalogen. For å generere ut hver sesong ble vi først nødt til å finne alle linjene som inneholdt informasjon om sesonger. Deretter måtte vi dele opp hver enkelt linje der det var en hashtag. Nå satt vi med

et langt array med hundrevis av like verdier, i vårt tilfelle var det fire forskjellige SS15, FW15, SS16 og FW16. For å finne disse fire verdiene ble vi først nødt til å sortere alle disse verdiene, slik at de som var like la seg sammen. Når alle de like verdiene var sammen, kunne vi automatisere en prosess som sjekket om den forrige verdien er lik den nåværende og hvis den ikke er lik, så skrev den den nye verdien til et nytt array og fortsatte prosessen. På den måten kunne vi bruke dette nye arrayet til å automatisk lage en knapp per sesong. Denne knappen deler opp hvert enkelt produkt sine sesonger, på samme måten som knappene ble implementert, men istedenfor å sortere verdiene og fjerne like verdier, leter den gjennom og ser om produktet inneholder den samme sesongen.

Filtrering på kjønn ble problematisk da vi flyttet komponenten med sortering og filtrering. Funksjonaliteten på selve filtreringen var den samme, men fordi vi tidligere hadde oppdatert produktene når skuffen lukket seg måtte vi løse dette på en ny måte. Vi hadde allerede bestemt oss for å bruke knapper. Derfor gjorde vi slik at når en av knappene blir klikket på, så sjekker den hvilken sjekkboks som er aktiv og filtrerer med dette kjønnet. Problemet oppstod hvis brukeren bare ønsket å filtrere med kjønn. I utgangspunktet ville vi ha en knapp sammen med sjekkboksene, men hvis knappen var i samme gruppe som sjekkboksene, slik at den kun vises hvis sjekkboksene vises, så klarte den ikke å sjekke hvilket kjønn som var aktiv. Vi prøvde og feilet og tok kontakt med klienten, uten å finne en løsning. Derfor ble vi enige om å la kjønnene være der slik at problemet kan bli fikset senere.

Vi hadde blitt enige med klienten om at de ikke trenger å ha filtrering på farger, men vil gjerne ha sortering. For å sortere på fargernavnet endret vi bare navn på funksjonen som tidligere hadde sortert produktnavn. Fordi strukturen på datasettet var annerledes enn tidligere, henter vi altså ut fargenavn på samme måte som vi tidligere hentet ut produkt navn. Dette førte til at vi måtte endre måten vi sorterte produktnavnet, men ved å skrive “item[‘description’]” som vist i figur 20, fungerer denne sortering også. For å sortere motsatt vei kan vi enten bytte plass på “<” og “>” eller bytte plass på “-1” og “1”. Med filtrering og sortering som fungerer går arbeidet videre med å få vist ut produktene med dataen klienten ønsket.

```
function alpha(a, b) {
    if (a.item['description'] < b.item['description'])
        return -1;
    else if (a.item['description'] > b.item['description'])
        return 1;
    else
        return 0;
};
```

Figur 20: Viser funksjonen som sorterer produktnavn, produktnavnet blir hentet ut ved "item['description']". Den sammenligner alle produktnavnene med hverandre og gir en verdi ut ifra hvilken som er først i alfabetet. (+1, 0 eller -1). Deretter returnerer den produktene med høyest tall først.

5.4.3. Produktene med nytt datasett

Presentasjonen av alle produktene må gjøres klar slik at de viser den informasjonen klienten ønsker. Det nye datasettet gjorde slik at koden til visning av produktene måtte forandres. Vi måtte gå gjennom koden og erstatte små kodesnutter for at katalogen igjen skulle vise produktene. Fordi vi nå viser ett produkt per farge til et produkt, betyr det at katalogen nå viser endel flere produkter. Dette gjør at de ti bildene som er i katalogen blir vist flere ganger, og fordi produktbildene skulle ha forskjellige farger ble vi nødt til å endre måten bilder blir hentet ut. Vi må også ferdigstille måten kortene blir presentert på slik at de viser den informasjonen som klienten ønsker.

Å gå gjennom koden å erstatte små kodesnutter for at det nye datasettet skulle fungere gikk fort og vi klarte å få katalogen til å vise produktene fra det nye datasettet. Figur 20(som er over) viser hvordan vi gikk gjennom koden og erstattet linjer med kode. Det nye datasettet gjør slik at katalogen har flere produkter og derfor ble vi også nødt til å endre måten bildene blir hentet ut på. Klienten ønsker at bildene skal bli hentet ut ved hjelp av produktkoden og fargekoden, og ikke kun produktkoden som vi hadde brukt tidligere. Å endre koden til å hente bilder var ikke vanskelig og ble gjort ved å endre en linje kode. Figur 21 viser hvordan vi endret fra "produktkode" til "produktkode_fargekode", problemet ved å gjøre dette er at de bildene som allerede var i katalogen ikke lenger blir vist og ingen av produktene hadde

bilde lenger. Vi ble derfor også nødt til å lage nye og flere bilder med det nye måten å skrive bildenavnet på.

```
<img src={"img/" + dataSet[z].item + ".png"}/>
<img src={"img/" + dataSet[z].item + "_" + dataSet[z].colors[x].code +  ".png"}/>
```

Figur 21: Viser hvordan vi henter ut bildet ved bruk av formatet "produktkode_fargekode". Rød linje blir erstattet av grønn

Vi skulle også ferdigstille hvilken informasjon produkt kortene viser. Klienten ønsket å ha navn på produktet, navnet på fargen og pris, både innkjøpspris og utsalgspris. De ønsket også å fjerne størrelsene fra produktet, selv om vi tidligere hadde fått inntrykk av at dette var viktig. Først ønsket vi å legge til de to prisene, men vi oppdaget at teksten som definerte prisene var skrevet "list-price" og "unit-price". Problemet med dette er at når vi skal hente ut disse to så må vi skrive .list-price og .unit-price, men når vi skriver "-" blir det registrert som at vi ønsker si ".unit minus price". Vi prøvde oss litt frem ved å finne ulike metoder å gjøre at katalogen leser "-" som en bindestrek, men de forskjellige forsøkene hadde alltid problemer, enten så fungerte det ikke eller så ga den oss informasjon som var uleselig. Derfor konsulterte vi oss med klienten og kom frem til at vi skulle endre datasettet fra å være "list-price" og "unit-price" til å være "listprice" og "unitprice" uten bindestrek. På den måten kan vi hente prisene ut på samme måte som om vi skulle hentet ut noe annet. Fordi produktnavnet og fargenavnet allerede var definert, manglet vi bare å formatere dataen slik at den blir presentert slik vi ønsker.. Vi beholdt produktnavnet som en tittel og la fargen til venstre under tittelen fordi vi leser fra venstre til høyre og da faller blikket naturlig på fargenavnet før de to prisene som vi valgte å legge til høyre. Da vi hadde ferdigstilt visningen av alle produktene fokuserte vi på å gjøre klar visningen av et produkt.

5.4.4. Dialogen

Når nytt datasett var koblet opp og kortene ferdigstilt har vi bestemt oss å fortsette arbeidet med dialogen. Som tidligere, måtte vi oppdatere koden slik at informasjon fra nytt datasettet blir hentet på riktig måte. For å finne ut hva som trengtes i dialogen har vi gått

gjennom tilbakemelding som vi har fått fra klienten. Siden vi hadde en klar liste over ting som mangler i dialogen, viste vi hva som måtte gjøres for få alt på plass.

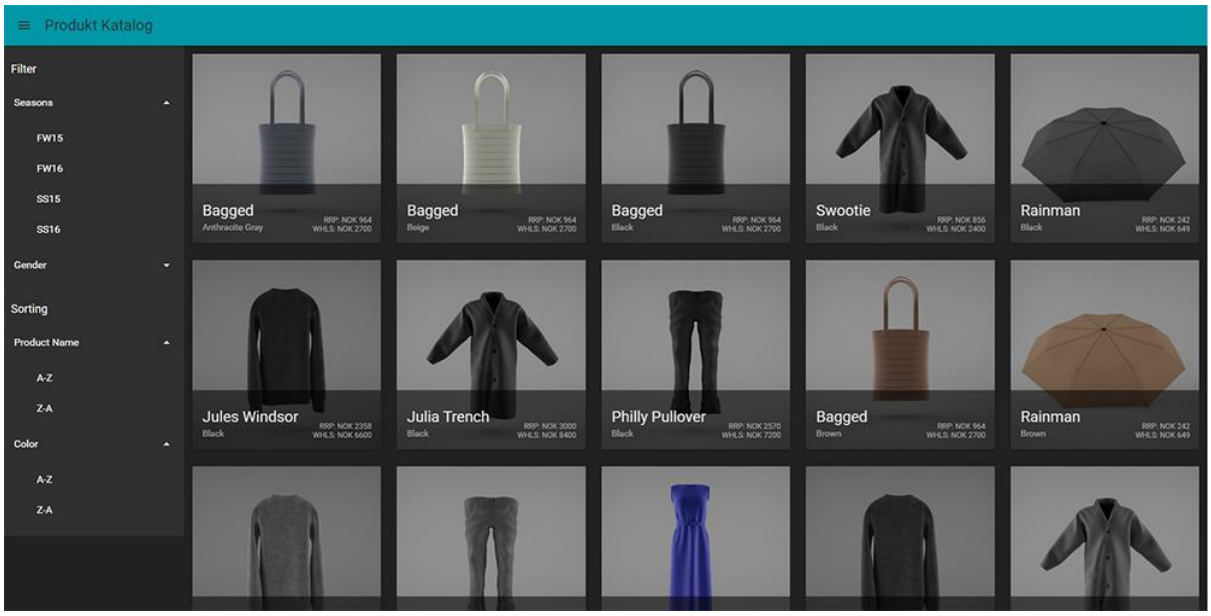
På starten oppdaterte vi bildene som blir vist i dialogen. Vi har løst problemet på samme måten som vi har gjort tidligere med produktene ved å oppdatere linken til bildet fra produktkode til produktkode med produktfarge. Videre fokuserte vi på å hente sesong, størrelse og pris til dialogen. Siden vi har jobbet med uthenting av data tidligere i prosessen, var det ikke lenger problematisk å hente ut den dataen vi trenger. Det vi måtte ha på plass i dialogen var alle sesongene produktet tilhører, kjønn, farge, størrelser og pris. I tillegg fikk vi ordnet alle dummy knappene som klienten hadde ønsket seg i dialogen. Under på figur 22 kan vi se at nedlasting knappen sammen med "add to draft" ble plassert nederst i høyre hjørne og favoritt øverst i høyre hjørne.

Figur 22: Knappene for nedlasting, "add to draft" og favoritt som ble plassert i dialogen. I tillegg ser vi hvordan verktøytips fungerer på favoritt knappen.

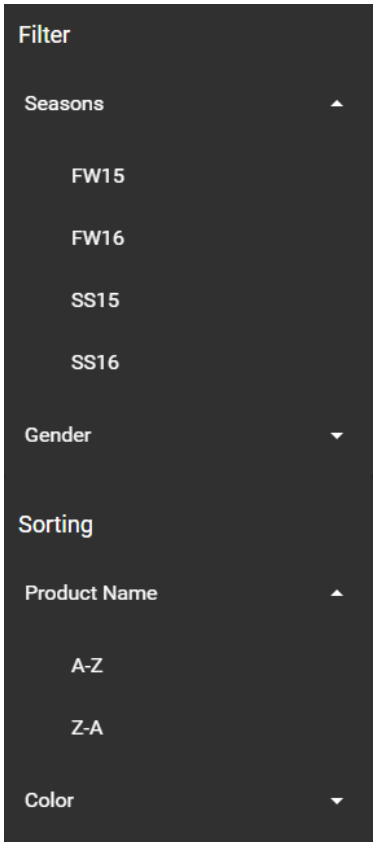
For nedlasting og favoritt har vi bestemt å benytte ikoner som er brukt i Material Design. Da vi satt opp ikonene, så vi at det passet bedre i dialogen siden de tok mindre plass og samtidig er det enkelt for brukeren å forstå hva disse indikerer istedenfor å ha store knapper med tekst. Allikevel hvis brukeren er ikke kjent med systemet eller katalogen og er usikker på hva disse knappene gjør, vil det komme et verktøytips om hva disse gjør i form av en grå firkant med hvit tekst. Da vi hadde hentet ut all nødvending informasjon til dialogen og satt opp dummy knappene, var dialogen ferdig.

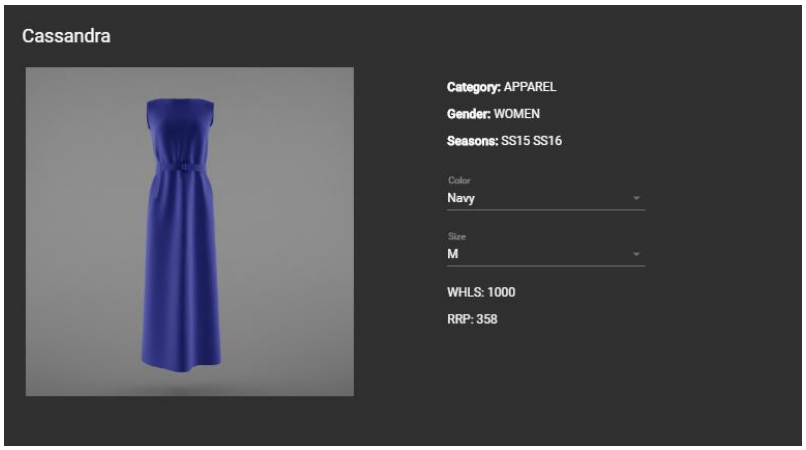
5.4.5. Siste utgave

I den siste versjonen av katalogen flyttet vi sortering og filtrering inn i selve katalogen. Figur 23 viser hvordan vi har gitt plass på siden av produktene til filtreringen. Hvert produkt blir vist en gang per farge og i figur 23 er det for eksempel flere av produktet "bagged", men de har forskjellige farger og forskjellige bilder. Hvert kort ble designet slik at de viser pris til høyre og farge til venstre, slik at øynene først faller på navnet til produktet og fargen. Figur 24 viser filtreringen og sorteringen. Hver av de forskjellige måtene å sortere og filtrere på, kan åpnes og lukkes slik at det ikke blir for mye informasjon på skjermen samtidig. Filtreringen på sesong er åpen når katalogen starter for å gi en indikasjon til brukeren på hva som ligger der. Figur 25 viser hvordan dialogen til et produkt viser frem den informasjonen klienten ønsket seg. I tillegg til å vise kategori, kjønn og farge, viser den nå også pris og størrelse. Dialogen har også fått knapper for å legge produkter til favoritter og ordre, slik at klienten selv kan implementere dette senere.



Figur 23: Siste versjon av katalogen, oversikt over alle produktene





Figur 24 (venstre): Filtrering og sortering, De forskjellige måtene å sortere og filtrere på kan åpnes og lukkes for å vise mer informasjon, Seasonger er åpen når katalogen starter

Figur 25 (høyre): Produkt-Siden, viser produktets kategori, kjønn, sesong, pris og dropdown til fargene og størrelsene.

Ved å besøke klienten, fikk vi svar på hva som gjensto for at vi kunne anse prosjektet som ferdig. Vi fant ut hva som måtte gjøres med kortene, hvilken informasjon trenger vi i dialogen og hvordan vi skal endre funksjonalitet rundt filtrering og sortering og samtidig fant vi ut hvor skal selve filtrering og sortering plasseres i katalogen. Med alt av tilbakemeldinger vi har fått kunne vi jobbe effektivt gjennom tredje sprinten siden vi visste nøyaktig hva som må gjøres. Gjennom sprinten jobbet vi hardt med å få alt som gjenstår på plass. Samtidig har vi tilpasset katalogen slik at klienten kan på enkelt måte videre utvikle den og samtidig koble den til Datorder. Videre skal vi se nærmere på hvordan vi har passet på å ha god kvalitet på det vi har utviklet gjennom alle sprintene.

5.5. Kvalitetssikring

For å sikre kvaliteten i prosjektet, har vi testet løsningen vår etter hver sprint. Vi har fulgt Google Material Design for å opprettholde bra design på siden. Vi har brukt tid på å gjennomføre intervju med klienten og potensielle brukere for å bli bedre kjent med hele tema rundt handel ved bruk av kataloger i forskjellige form og samtidig få bedre innsikt i hva som må fremstilles i katalogen. Etter hver sprint vi var ferdig med, delte vi resultatene av vår arbeid med klienten for å få tilbakemelding slik at vi kan ta den med til neste sprint. På denne måten kunne vi jobbe effektivt med klienten og nå kravene som ble satt opp for prosjektet.

5.5.1. Intervju

I løpet av designprosessen gjennomførte vi tre intervjuer, ett med klienten og to med potensielle brukere av systemet. I intervjuet med klienten fikk vi bedre forståelse for hvordan Datorder fungerer og samtidig fikk vi se en applikasjon laget med React og Material-UI. Vi fikk vite at vi har ganske frie tøyler når det gjelder designet, og vi fikk en litt mer detaljert forklaring på hva oppgaven går ut på. Da vi gjorde de to andre intervjuene med potensielle brukere ønsket vi å få vite hva de ser etter når de velger produkter, hva slags informasjon som er viktig og hvordan prosessen med å bestille varer er. Vi fikk vite at pris ofte er det viktigste, både for intervjuobjektene og kunden, men at bildene som blir brukt i katalogen ofte også kan være viktige for hvilket produkt som blir kjøpt. Når kunden bruker papirkataloger kan det enkelt skje feil, ved at for eksempel brukeren skriver feil produktkode eller at brukeren misforstår produktbilde til produktkode. Ved hjelp av disse intervjuene fikk vi mulighet til å kartlegge hva som var viktig å ha med når vi skulle utvikle digital løsning og vi fikk bedre forståelse av hva som forventes fra en løsning som Datorder.

5.5.2. Testing/tilbakemelding

Vi har gjort to brukertestinger, en etter vi hadde laget prototype og en etter den første sprinten. Fordi funksjonaliteten og designet ikke har forandret seg så mye i løpet av de to siste sprintene, så vi heller ikke nødvendigheten av å teste produktet med personer hentet utenfra. Størrelsen på løsningen gjorde at testing gikk fort, men førte også til at testing ikke

ga oss de store resultatene. Da vi testet prototypen fikk vi fire personer til å klikke seg litt rundt. Vi fikk lite ut av denne brukertesten, fordi prototypen hovedsakelig var laget som en layout som vi kunne bruke i implementerings prosessen. Filtreringen på denne prototypen var også inne i selve katalogen og derfor ble den sett av testpersonen med engang. Tre av testpersonene fant mer informasjon om et produkt og siste personen bommet da han prøvde å klikke på en av filtreringene og prototypen lyste opp på de stedene som kunne bli klikket på. Vi kjørte andre runde med testing etter den første sprinten. Vi gjorde det ganske likt som med prototypen bare denne gangen hadde vi seks personer. Det vi håpte få ut av denne testen var om testpersonene klarte å finne de ulike komponentene. Det var liten tvil om hvordan de kunne få mer informasjon om et produkt, og alle forsto de kunne klikke på et produkt for å få vite mer. Det var litt mer problematisk å finne filtreringen. Tre personer fant frem til hamburgermenyen og klikket på denne, mens de tre resterende ikke fant denne. Vi valgte å ha dette i bakhodet til senere, og etter den tredje sprinten flyttet vi filtreringen inn i selve katalogen. Denne testen ble også ganske liten, men vi tok med oss det lille vi fikk ut av den.

5.5.3. Google Material Design

Når vi har designet katalogen har vi fulgt Google Material Design prinsippene. Material-UI komponenter har gjort slik at de komponentene vi har brukt har blitt designet innenfor disse prinsippene. Menylinjen i toppen er en meny som er definert av Google, det samme gjelder kortene og måten seksjonen til filtrering og sortering er satt opp. Produktene blir vist frem ved hjelp av cards som vi har redigert litt, men vi har beholdt funksjonaliteten til kortet som er definert som "*A card is a sheet of material that serves as an entry point to more detailed information*." (9). Selv om designet er litt forandret fra slik det ble levert fra Material-UI, har vi passet på at de ikke bryter med prinsippene definert av Google. Når vi genererte ut kortene har vi brukt en komponent kalt Grid List. Grid List er relatert med kortene og er optimalisert for å vise elementer fra JSON filer eller fra array slik vi har gjort.

5.5.4. Samarbeid med klienten

På grunn av Slack hadde vi gjennom prosjektet god kontakt med klienten. Vi ønsket å opprettholde et godt samarbeid og derfor har det vært viktig for oss å kunne holde kontakt på en litt uformell måte, i form av chat. I løpet av prosjektet var det enkelt å spørre om hjelp eller andre ting vi lurte på. I designprosessen fikk vi tilsendt linker til konkurrenter og inspirasjon da vi spurte om dette. Vi fikk også tilbakemelding på prototypen og produktet underveis. Når vi hadde problemer med kode kunne vi spørre om hjelp. Fordi det å holde kontakten med klienten ble litt uformelt gjorde det at det var enklere for begge parter å ytre meningene sine. Dette førte til at vi kom til diskusjoner som ga oss bedre forståelse og fikk begrunnelser for hvorfor en løsning var bedre enn en annen.

5.6. Konklusjon

I løpet av implementasjon prosessen har vi lært hvordan vi kan bruke React og JavaScript til å løse forskjellige problemer. Vi ble også bedre kjent med Google Material Design prinsippene og fikk bedre forståelse av hvordan vi kunne benytte disse i katalogen. Gjennom tre sprinter har vi utviklet tre utgaver av løsningen. Disse ble utviklet med stort fokus på tilbakemelding fra klienten i løpet av sprintene. Den første sprinten ga oss en katalog som la grunnlaget for katalogen. Komponentene var på plass og riktig data ble hentet til produktdialogen. Til den andre sprinten fikk vi en reell JSON-fil, slik at vi kunne bytte ut dummy dataen vi brukte i første sprint til å presentere produktene. I den andre utgaven fikk vi gitt hvert produkt et bilde, samt hentet ut relevant informasjon til hvert kort. En del filtreringsfunskjoner kom på plass og dialogen viser nå frem produktkategori, kjønn, farge og bilde. Vi byttet også fargetema fra lyst til mørkt. I den tredje sprinten lagde vi den siste utgaven av katalogen. I denne versjonen ønsket klienten at katalogen skulle vise et kort per farge per produkt. Dette førte til at vi måtte få en ny JSON-fil og endre litt av strukturen på koden. Produktkortene ble videre designet for å vise frem navn, farge og pris. Filtreringen ble flyttet fra hamburgermenyen inn i selve katalogen og vi implementerte filtrering for sesong, navn og fargenavn. Vi har også filtrering på kjønn, men denne er fortsatt ikke helt

funksjonell. Dialogen fikk mer informasjon og viser nå sesong, størrelser og diverse knapper uten funksjon som klienten ønsket, slik at de kan knytte disse knappene til sitt system senere, i tillegg til det dialogen allerede viste fra andre utgaven.

6. Avslutning

I dette kapitelet skal vi drøfte utviklingsprosessen, hvordan vi har jobbet sammen og sluttresultatet. Samtidig skal vi komme med forslag for videre utvikling for å forbedre produktkatalogen. Vi skal også se på hva som må gjøres for å koble produktkatalogen til Datorder, slik at alt fungerer optimalt.

6.1. Evaluering av oppgaven

Vi er fornøyd med prosessen og måten vi har gjennomført oppgaven. Det å lære React og NPM har vært lærerikt og noe vi kan ta med oss videre. Vi er skuffet over at katalogen ikke ble ferdig, men fornøyde med det vi kan ta med oss videre.

Designprosessen ble litt som forventet, fordi vi har mye erfaring fra tidligere med å lage skisser, prototype og gjøre intervjuer. Vi valgte å ikke lage wireframes, men laget en prototype istedenfor pga. størrelsen på applikasjonen. Da vi hadde en prototype valgte vi å gjøre litt testing. Etter denne testingen satt vi med en følelse av at dette var unødvendig å gjøre, fordi testen var så liten og rask gjorde dette at testpersonen omtrent ikke rakk satt seg ned før han var ferdig og vi fikk derfor lite ut av testen. Selv om vi satt med dette i bakhodet valgte vi også å kjøre en test etter den første sprinten for å se om vi kunne få mer ut av det denne gangen. Vi valgte å ha flere testpersoner denne gangen for å få bedre resultat. Det eneste resultatet vi fikk fra den andre testen var at filtreringen var litt for godt gjemt. Vi valgte å ikke gjøre noe med dette umiddelbart. Vi satt igjen med litt følelse av at testen ikke ga oss så mye, men sa oss fornøyde med å ha fått litt tilbakemelding. Etter andre sprinten valgte vi derfor ikke å ha en test, men ønsket å ha en etter den siste sprinten for å ha litt tilbakemelding å videreformidle til klienten slik at de hadde litt konkret å jobbe ut ifra.

Uheldigvis rakk vi ikke å gjennomføre denne testen da fokus ble rettet på rapporten den siste tiden av prosjektet.

Vi er hovedsakelig mest fornøyd med den første og andre sprinten, selv om vi ikke fikk gjort så mye som vi i utgangspunktet hadde ønsket. Prosessen med å sette opp React og Material-UI tok veldig mye lengre tid enn vi hadde regnet med. Vi hadde erfaring med JavaScript fra tidligere og trodde React var mer likt JavaScript enn det var. I utgangspunktet trodde vi at vi bare trengte laste ned React og Material-UI rammeverkene fra nettsidene deres, men det viste seg at vi ble nødt til å lære oss å bruke NPM for at Material-UI skulle fungere. React kunne lastes ned fra nettsiden, men fordi Material-UI måtte bli installert via NPM valgte vi å gjøre det samme med React. Uten erfaring med NPM fra tidligere tok det mye tid før vi klarte få alt til å fungere skikkelig og vi fikk derfor ikke til det vi utgangspunktet hadde håpet på. Da vi startet på den andre sprinten var vi fortsatt ikke helt vant med React og ting tok lang tid. Derfor var forventingene til den andre sprinten ikke helt optimistiske, men etterhvert som vi jobbet forstod vi rammeverkene bedre og tempoet på programmeringen gikk drastisk opp halvveis ut i sprinten. Den andre sprinten er den vi er mest fornøyd med, fordi vi forstod rammeverkene bedre og kom lengre enn forventet, selv om det var litt surt å måtte fjerne filtreringen på farger i tredje sprint. Den tredje sprinten ble veldig hektisk fordi deadline kom veldig fort. Vi visste vi måtte begynne å skrive på rapporten, men ønsket å fullføre katalogen så mye som mulig. Klienten var informert om at vi ikke hadde mye tid igjen og er klar over at produktet de får, ikke er ferdig. Klienten er selv utviklere slik at de har kontroll på hvordan de kan ferdigstille løsningen.

Det produktet vi leverer til klienten har vi blandede følelser for. Vi er fornøyde med prosessen og det vi leverer, men vi ønsket å kunne levere en ferdig versjon. De nye rammeverkene gir oss erfaring som er nyttig til senere og vil gjøre at det blir lettere å lære nye JavaScript rammeverk i fremtiden. NPM er også en viktig løsning som vi tar med oss videre.

6.2. Gruppearbeid

Vi har jobbet mye sammen gjennom bacheloroppgaven og var kjent med arbeidsmetodene til hverandre. Vi fant fort ut at det var mye lettere å samarbeide ved å møte der vi kan jobbe sammen, både under designprosessen og implementeringen. Hvis vi var uenige om noe, kunne vi alltid diskutere og argumentere hva som vil være best for prosjektet vårt. Siden vi bare var to var det lettere å inngå kompromiss som begge to var fornøyde med. Av og til var det best for oss å høre med klienten for å finne den beste løsningen.

Vi følte at møter der vi jobbet sammen passet oss best, det var lettere å fordele oppgaver mellom hverandre og samtidig passe på at vi gjør det vi skal. Vi passet på at begge alltid hadde arbeidsoppgaver og gjorde det vi skulle. Ved å alltid sitte sammen å jobbe var vi flinke til å passe på så den andre ikke arbeidet med urelevante funksjoner og når vi så den andre slet kunne vi avlaste hverandre. Vi lærte også endel fra hverandre når vi arbeidet slik. Begge to har god forståelse av koden som den andre har skrevet, og vi har brukt mye tid på å forklare koden vi har skrevet til hverandre. Fordi vi har sittet så mye sammen har vi også hatt perioder vi har blitt irriterte på hverandre, men vi har løst det med å ta noen minutters pause fra hverandre. Det har aldri vært store uoverensstemmelser, slik at vi har kommet sammen igjen og spøkt litt med det. Vi har vært flinke til å si ifra når vi er uenige og når vi har kommet over kode vi ikke har forstått, så vi har tatt dette opp ved at vedkommende forklarer koden sin. Når vi har vært uenige om koden har vi ikke vært redde for å si ifra, fordi vi kjenner hverandre godt.

6.2.1. Arbeidsfordeling

Siden vi bare var to personer på gruppa, var det naturlig for oss å gjennomføre de fleste oppgavene sammen, slik at vi kunne bearbeide resultatene av disse i fellesskap. Siden vi var lite kjent med React så bestemte vi oss for å lese litt om det på egenhånd, men vi jobbet sammen når vi skulle begynne med å bruke det i prosjektet. Siden vi måtte sette opp alt i NPM for å få programmering i gang, satt vi i kort tid med samme oppgave. Etterhvert da vi begynte å få disse tekniske tingene på plass, kunne vi endelig fordele oppgaver mellom

hverandre slik at begge to kunne fokusere seg på forskjellige elementer i katalogen. Vi har hovedsakelig hatt ansvar for forskjellige komponenter, men vi har engasjert oss i hverandres arbeid. Katalogen er bygd opp av tre hovedkomponenter, filtrering og sortering, produktkortene og produktdialogen. Vi har delt arbeidet slik at én har hovedsakelig arbeidet med filtrering og sortering, mens den andre har jobbet med kortene. Dialogen har blitt fokusert på når den første ble ferdig med komponenten sin. Fordi vi har arbeidet sammen har begge jobbet mye med hverandres komponenter. Det at vi har arbeidet på denne måten har ført til at vi har jobbet like mange timer, det har vært nok arbeidsoppgaver slik at vi har hatt minimalt med dødtid.

6.3. Videre utvikling

Vi synes at katalogen vår har mange potensielle muligheter for videre utvikling, men det er også en del som kan endres for å gjøre utgaven vår bedre. Samtidig skal vi se nærmere på ting som må gjøres før løsningen kan implementeres i systemet til klienten.

6.3.1. Hva som må være på plass

Et av ønskene til bedriften er at det skal være mulig å vise fram flere bilder av ett produkt i dialogen. Vi har satt av plass under bilde i dialogen slik at det enkelt kan plasseres inn flere bilder der hvis det flere bilder av ett produkt. Bildene må også gjøres klikkbare slik at brukeren kan endre hvilket bilde som er i fokus. Det kan også være lurt å endre litt på elementene som å plassere elementer litt annerledes og eventuelt endre litt på skriftstørrelse for å fremheve den informasjonen som er viktig. Vi har masse tom plass i dialogen som er tilberegnet produktbeskrivelse etter ønske fra klienten, men vi har ikke fylt dette område med fylltekst eller en placeholder til denne teksten. Vi har også nevnt tidligere at filtrering for kjønn ikke fungerer optimalt. Det må implementeres en knapp for at det skal være mulig å kun filtrere på kjønn, vi hadde problemer med at når knappen var implementert sammen med sjekkboksene, så fungerte den rett og slett ikke. Filtreringen på kjønn fungerer hvis brukeren klikker på en knapp for å filtrere på sesong.

6.3.2. Hva som kan være med

Det er en del funksjoner som kan implementeres for å gjøre katalogen bedre. For å gi brukere mulighet til å legge produkter raskere til favoritt, kan det være aktuelt å sette opp favorittknapp i hvert produktkort. I utgangspunktet ønsket vi å ha den samme stjernen i hvert produkt som vi har i dialogen. Hvis brukeren liker å kunne bla gjennom produkter som i en papirkatalog, kan det være relevant med en funksjon som gjør at man kan hoppe til neste produkt når en dialog er allerede åpen. Det kan være en nyttig løsning hvis man i tillegg kobler funksjon til piltastene på tastaturet for å bytte mellom produkter. Klienten har også snakket om muligheten til å vise produktene i form av en liste uten bilder istedenfor kortene. Dette kan brukes til å fremstille flere detaljer om et produkt og man kan navigere seg gjennom alle produktene enda raskere uten å måtte klikke seg gjennom hvert produkt for å se mer detaljert informasjon. Siden vi stort sett har fokusert på å bli ferdige med en skrivebordsversjon, og vi har bare testet katalogen i mobil- og nettbrett-oppløsning via nettleseren. Vi har sett at det må bli gjennomført en del tilpasning for at katalogen skal fungere optimalt på forskjellige enheter. For eksempel må katalogen vise flere eller færre produkter ut ifra oppløsning, dialogen må optimaliseres og filtreringen må kunne gjemmes bort eller plasseres på toppen for å ikke bruke for mye plass. Slik katalogen er i dag viser den sortering i form av knapper som sier enten “A-Z” eller “Z-A” og disse tar en linje hver. Vi vurderte å bytte dem til knapper som tok en halv linje hver og bytte teksten til Google Material Design symboler, men vi valgte å ikke gjøre dette og heller fokusere på å gjøre ferdig viktigere deler av katalogen.

6.3.3. For implementering

For å implementere katalogen i Datorder, må klienten gjennomføre noen tilpasninger til katalogen. Klienten må erstatte menylinjen vår med den som er i resten av systemet som blir brukt til å navigere seg gjennom resten av systemet. I tillegg må klienten sette opp en knapp i sitt system for å komme til katalogen. Katalogen har en del dummy knapper som vi har satt opp etter ønske fra klienten. Knappene i seg selv gjør ikke noe, derfor må klienten få funksjonalitet for favoritter, bestilling og .csv-nedlasting. Disse funksjonene kommer til å ha kobling til brukersystemet som Datorder benytter. Klienten har satt opp forskjellige valuta i

JSON-fila, noe som vi har diskutert med bedriften og kommet fram at vil være en del av innstillinger som brukeren kommer til å ha i systemet. For at fargene i katalogen skal være tilpasset til de som er brukt i Datorder, er det viktig at klienten kobler fargepaletten som er brukt i systemet også til katalogen.

6.4. Konklusjon

Alt i alt har dette vært et nyttig og lærerikt prosjekt. Vi har fått erfaring med nye rammeverk og fått erfaring med NPM, som kan være nyttig å ta med seg i arbeidslivet. Det å arbeide med en reell kunde har gitt oss en smakebit på hvordan arbeidslivet er. Vi klarte dessverre ikke å fullføre katalogen, men er dog fornøyd med prosessen og det vi har lært. Det er essensielle funksjoner som må fikses eller implementeres for at katalogen skal kunne implementeres i Datorder. Vi ønsket også å ta en test til for å gi klienten tilbakemelding på hva som bør fikses eller forandres på når de tar over arbeidet, men tiden strakk dessverre ikke til. Vi kunne ha avsluttet arbeidet tidligere for å få tid til dette, men vi syntes det var bedre jobbe med katalogen så lenge vi kunne. Vi tar erfaringene med oss videre og lærer av både oppturer og nedturer.

Litteraturliste

1. CoreTrek AS. Scrum i et nøtteskall. CoreTrek [sitert: 12.05.16]. Tilgjengelig fra: http://www.coretrek.no/om-oss/utviklingsmetodikk/scrum/

2. Sandip Das, What is React.js and Why I recommend it to other JavaScript Developers? LinkedIn, 2015 [sitert 09.05.2016]. Tilgjengelig fra: https://www.linkedin.com/pulse/what-reactjs-why-i-recommend-other-javascript-sandip-das

3. NPM, Getting Started, 2016 [sitter 13.05.2016] Tilgjengelig fra: https://docs.npmjs.com/how-npm-works

4. NPM, package.json – Specifics of npm's package.json handling , 2016, [sitter 09.05.2016] Tilgjengelig fra: https://docs.npmjs.com/files/package.json

5. React, JSX in Depth, 2016 [sitert: 11.05.2016]. Tilgjengelig fra: http://facebook.github.io/react/docs/jsx-in-depth.html

6. Helge Østbye, Knut Helland, Karl Kanpskog, Leiv Ove Larsen, Hallvard Moe, [sitert 12.05.2016] Metodbok for mediefag. 4 utg. Fagbokforlaget, 2013, side 103

7. Helge Østbye, Knut Helland, Karl Kanpskog, Leiv Ove Larsen, Hallvard Moe, [sitert 12.05.2016] Metodbok for mediefag. 4 utg. Fagbokforlaget, 2013, 104-105

8. Mozilla Developer Network, Array.prototype.map(), 2016, [sitert 04.05.2016]. Tilgjengelig fra: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

9. Google, Material Design - Card, 2016, [sitert 14.05.2016]. Tilgjengelig fra: https://www.google.com/design/spec/components/cards.html

Vedlegg A

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

SIC AS v/ Ib Christiansen (oppdragsgiver), og
Halvard Meijer
Maksymilian Sadowski
(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11/1 - 2016 til 05/6 - 2016.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra HiG. Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik

offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan/prodekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Når HiG også opptrer som oppdragsgiver trer HiG inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): EMIL BAKKE

Oppdragsgivers kontaktperson (navn): IB CHRISTIANSEN

Student(er) (signatur): _____ dato 27/1 -2016

_____ dato 27/1 -2016

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur): _____ dato 27/1-2016

IMT Dekan/prodekan (signatur): _____ dato _____

Versjon Januar 2011hb

Vedlegg B

2016

Prosjektplan

IKATALOG

HALVARD MEIJER
MAKSYMILIAN SADOWSKI

Contents

1. Mål og rammer 2
1.1. Bakgrunn 2
1.2. Prosjektmål 2
1.2.1. Effektmål 2
1.2.2. Resultatmål 2
2. Omfang 2
2.1. Fagområde 2
2.2. Avgrensning 3
2.3. Oppgavebeskrivelse 3
3. Prosjektorganisering 3
3.1. Ansvarsforhold 3
3.2. Grupperegler 4
4. Planlegging, oppfølging og rapportering 4
4.1. Hovedinndeling av prosjektet 4
4.2. Plan for statusmøter og beslutningspunkter 5
5. Organisering av kvalitetssikring 5
5.1. Dokumentasjon, standardbruk og kildekode 5
5.2. Konfigurasjonsstyring 5
5.3. Risikoanalyse 6
5.4 Verktøy 7
6. Plan for gjennomføring 9
7. Referanser 10

1. Mål og rammer

1.1. Bakgrunn

SIC AS leverer et B2B (business-to-business) system og ønsker en interaktiv katalog i denne løsningen. De ønsker å effektivisere måten bedrifter kjøper klær og sko slik at de minsker feil fra brukeren og hjelper bedriften spare tid og penger. Den gamle måten å drive med engroshandel er å fylle ut regneark eller skjemaer. Det skjedde ofte menneskelige feil i slike måter å drive handel på, for eksempel at man skriver feil produktkode eller hvis det blir gjort for hånd at det er utydelig skrevet. Ved å ha en slik webbasert katalog løsning gjør det at brukeren har vanskeligere for å gjøre feil. Bedriften vil også spare penger fordi de slipper bruke ekstra timer på arbeid man ellers ville brukt. Det er også viktig for SIC at katalogen ser moderne ut og de ønsker at den skal følge prinsippene til Google Material Design. Ved å tilby en løsning som er både enkel å bruke, ser bra ut og sparer bedriften penger, kan SIC også levere en bedre løsningen til bedrifter som allerede har en digital løsning.

1.2. Prosjektmål

1.2.1. Effektmål

- Lære og mestre React.js
- Forstå og benytte Google Material Design til å lage bedre design
- Oppleve samarbeidet med en reell kunde

1.2.2. Resultatmål

- Ferdigstilt produktkatalog som SIC kan bruke i sitt system
- Ferdigskrevet rapport med beskrivelse av arbeidet vårt gjennom prosjektet
- Lage en god og responsive UI med Google Material Design elementer

2. Omfang

2.1. Fagområde

- Brukergrensesnitt lagd ved bruk av React.js
- Clojurescript
- UX-Design
- Produktkatalog
- Typografi

- Material Design UI

2.2. Avgrensning

- Ingen native løsning for katalogen, som fører til at brukeren kun har mulighet til å benytte produktkatalogen online.
- Vi skal kun teste på de moderne nettleserne

2.3. Oppgavebeskrivelse

Bedriften ønsker seg en god og oversiktlig produktkatalog som de skal integrere i deres system. Katalogen skal være for klær og sko, og det er bedrifter som driver med engroshandel som skal bruke katalogen. Det skal være en webbasert løsning som følger Google Material Design prinsippene og den skal lages ved bruk av React.js.

Det er viktig for bedriften at katalogen kommer til å vise produktene på oversiktlig, konsis og pen måte. Hovedpoenget er at målgruppen til løsningen skal navigere seg gjennom produktene på en enkelt og rask måte. Det skal ikke være nødvendig å ha opplæring for å ta katalogen i bruk. Det er også et krav om at det skal være mulig for brukere å sortere produktene etter sitt eget ønske. Produktene skal kunne kategoriseres etter produktkategori, merke, kjønn, pris, sesong, farge, navn, produktgruppe og tilgjengelighet. Den skal også kunne sorteres i ulike hierarkier det vil si at brukeren skal kunne klare å finne produkt kategorier som genser og vinter både ved å gå via Vinterklær til vintergensere eller via gensere til vintergensere. Det skal også være mulig å legge til produkter i favoritter eller legge til ordre. Produkter i katalogen skal kunne ha bilder, flere bilder eller ingen bilder.

3. Prosjektorganisering

3.1. Ansvarsforhold

- **Gruppeleder** - Halvard Meijer
  - Kan ta avgjørelser hvis det er usikkerhet, men til store protester skal veileder kontaktes
  - Ansvar for at gruppemøtene ikke sporer av
  - Ansvar for utdeling av arbeidsoppgaver
  - «Scrum-master»
- **Sekreter** - Maksymilian Sadowski
  - Loggføre møter
- **Veileder** - Emil Bakke

- **Arbeidsgiver** - SIC AS
- **SIC AS kontaktperson** - Ib Christiansen

3.2. Grupperegler

§1 - Fravær fra fastemøtetider må meldes ifra om dagen før og ett nytt møte skal arrangeres

§2 - Avgjørelser skal gjøres ved å diskutere fordeler og ulemper: Vi skal deretter ta avgjørelsen ut fra dette. Hvis vi ikke kommer til enighet skal avgjørelsen bli gjort i samarbeid med veileder eller bedriften

§3 - Alle skal loggføre arbeidet sitt med antall timer og beskrivelse av hva som ble gjort

§4 - Etter faste gruppemøter skal vi utdele arbeidsoppgaver

§5 - Alle kan underskrive på vegne av gruppa, men alle må være klar over signeringen

§6 - Ved brudd på denne kontrakten skal veileder kontaktes

4. Planlegging, oppfølging og rapportering

4.1. Hovedinndeling av prosjektet

Design prosessen skal gjennomføres ved hjelp av skisser, prototyping, brukertesting, god kommunikasjon med bedriften og ved å følge retningslinjene i Google Material Design. De første skissene blir bare på papir, men etterhvert som vi begynner lande på et design vil vi bruke prototypeverktøyet proto.io. Slik at vi enkelt kan teste og se hvordan brukere reagerer til designet vårt. Google Material Design gir oss retningslinjer for at designet skal se moderne og bra ut, slik at vi kan fokusere mer på å gjøre katalogen bra å bruke.

Vi skal gjennomføre selve programmeringen i prosjektet med utgangspunkt i Scrum. Der vi skal jobbe i sprinter på en uke og skal ha møte med bedriften annenhver uke. Scrum anbefaler sprinter på 2-4 uker, men vi velger å ha oppfølging hver uke for at vi alltid skal ha arbeidsoppgaver utdelt og kan hjelpe hverandre hvis vi har problemer. Gruppeleder har også rollen som “Scrum Master”. I Scrum skal alle arbeidsoppgavene i utgangspunktet være ferdig når sprinten er ferdig, men vi vil gå litt bort ifra dette fordi noen av arbeidsoppgavene kommer til å ta lengre enn en uke. Vi skal ha Skype møte med bedriften hver 14. dag hvor vi kan gå igjennom og få en tilbakemelding på det som vi har gjort.

4.2. Plan for statusmøter og beslutningspunkter

Vi har bestemt oss for å ha gruppemøter ukentlig, hver onsdag. Dersom vi finner ut at det nødvendig å ha flere møter kan vi arrangere dette. På møtene skal vi diskutere fremdriften og fordele oppgaver mellom gruppemedlemmene. Vi har også bestemt at vi skal bruke disse møtene til å jobbe sammen hvis det er noe som må bli gjennomført eller diskutert i fellesskapet. Problemene som gruppa møter gjennom prosjektet blir også tatt opp på møtene slik at vi kan bli enig om hvordan disse skal løses.

Sammen med veileder har vi bestemt oss for å ikke ha faste tidspunkter for statusmøter. Gruppa kommer til å holde kontakt med veileder via e-post. Vi vil ha hyppige møter i starten, litt sjeldnere utover i prosjektet og mer hyppig igjen når vi nærmer oss slutten av prosjektet. Vi skal bruke disse møtene for å ta opp ting som gruppa sliter med eller om det er noe vi er usikker på. Dersom det blir behov for flere møter med veileder så skal vi ta kontakt for å sette opp en plan med faste møtetider.

Større beslutninger og uenigheter skal bli tatt opp med arbeidsgiver eller veileder for å avklare hva som må fikses eller gjøres. Ting blir ofte diskutert skriftlig via Slack, men vi har bestemt oss for å ha møter på Skype annenhver uke med bedriften, slik at vi kan vise og diskutere resultatet av sprintene. Vi kommer også til å ha ett par vanlig møter med bedriften i Oslo.

5. Organisering av kvalitetssikring

5.1. Dokumentasjon, standardbruk og kildekode

Alle våre dokumenter kommer til å bli skrevet og lagret på Google Drive slik at alle medlemmene har tilgang til all dokumentasjon. Den siste versjonen av rapporten og prosjektplanen kommer til å bli formatert ved hjelp av Microsoft Word. Dokumentene blir levert i PDF format.

Sekretæren skal loggføre alle møtene i ett dokument i Google Drive. Hvert møte blir registret med dato, start og slutt tid og hva som ble gjort i løpet av et møte. For å loggføre individuell arbeid skal vi benytte Google Spreadsheet.

Koden kommer til å bli skrevet på engelsk med betydningsfulle navn. Det er også krav at koden blir kommentert slik at den blir forståelig for andre gruppemedlemmer og arbeidsgiver.

5.2. Konfigurasjonsstyring

For å ha god kontroll over kildekoden skal gruppen benytte BitBucket som konfigurasjonsstyringsverktøy. Sammen med dette skal vi bruke SourceTree som vil gi alle

medlemmene god visuell oversikt over arbeidet og samtidig som vi får tilgang til kommandolinjen hvis det blir nødvendig.

Google Docs vil hjelpe oss med konfigurasjonsstyring for prosjektdokumentasjon. Tjenesten kommer også med en logg som viser endringene som ble gjort med navn og dato.

5.3. Risikoanalyse

	Sannsynlighet	**Utfall (8 timer per dag)**
Lav	0 til 1 ganger i løpet av prosjektet	Opptil 1 dag ekstra arbeid
Middels	2 til 3 ganger i løpet av prosjektet	1 - 3 dager ekstra arbeid
Høy	Mer enn 4 ganger i løpet av prosjektet	Mer enn 4 dager ekstra arbeid eller prosjektet må revurderes (kontakt veileder)

#	**Risiko**	**Sannsynlighet**	**Utfall**
1	Sykdom	Høy	Middels
2	Mister data	Lav	Høy
3	Gruppe medlem dropper ut	Lav	Høy
4	Bitbucket/Google Drive/Slack er nede	Lav	Lav
5	Beregner for dårlig tid	Høy	Middels
6	Dårlig kontakt med arbeidsgiver/veileder/innad i gruppa	Lav	Middels
7	Dårlig fordeling av oppgaver	Høy	Middels

Risiko	Forebyggende tiltak	Hvis det skjer
1	Utsette møter. Ved mindre alvorlig sykdom kan vi jobbe hjemmefra.	Gi beskjed i så fort som mulig slik at vi kan organisere arbeidet ut ifra dette.
2	Lagre alltid backup av data på et sted som alle gruppemedlemmer har tilgang til (Google Drive, Bitbucket)	Hente tidligere backups hvis det er mulig og jobbe effektivt for å unngå stort tap av tid
3	Gruppereglene definerer dette	Kontakt veileder og arbeidsgiver
4	Alltid ha siste versjonen av dokumenter/filer tilgjengelig lokalt. Alltid sjekke nye meldinger på Slack	Fortsette å jobbe med det vi kan til programvaren er oppe igjen.
5	Diskutere hvor mye tid vi tror ting tar og legge til noen ekstra timer.	Fokusere på det som holder oss igjen og dele det opp slik at vi effektivt kan fullføre. Hvis det er en mindre viktig funksjon, kan vi velge å hoppe over den.
6	Ha alle kommunikasjonsverktøy tilgjengelig på alle enheter. Faste møtetider	Ha ett møte med bedriften/veileder om hvordan vi kan forbedre kommunikasjonen
7	Planlegge hvem som gjør hva og si ifra hvis vi møter problemer. Hjelpe hverandre hvis det er nødvendig	Ha et felles møte og jobbe sammen med problemet eller gi den mer tid

5.4 Verktøy

Google Drive - skytjenesten levert av Google. Verktøyet blir brukt for å lagre alle viktige dokumenter på et sted slik at alle gruppemedlemmene har tilgang.

Google Docs - et online tekstbehandlingsprogram som er integrert med Google Drive. Verktøyet vil hjelpe oss med felles skriving av rapporten og andre viktig dokumenter.

Slack - kommunikasjons verktøyet som kommer til å bli brukt som et kontaktpunkt med arbeidsgiver. Slack er en enkel chat løsning til både pc og mobil slik at kontakten med bedriften blir lettere og raskere enn å måtte bruke epost.

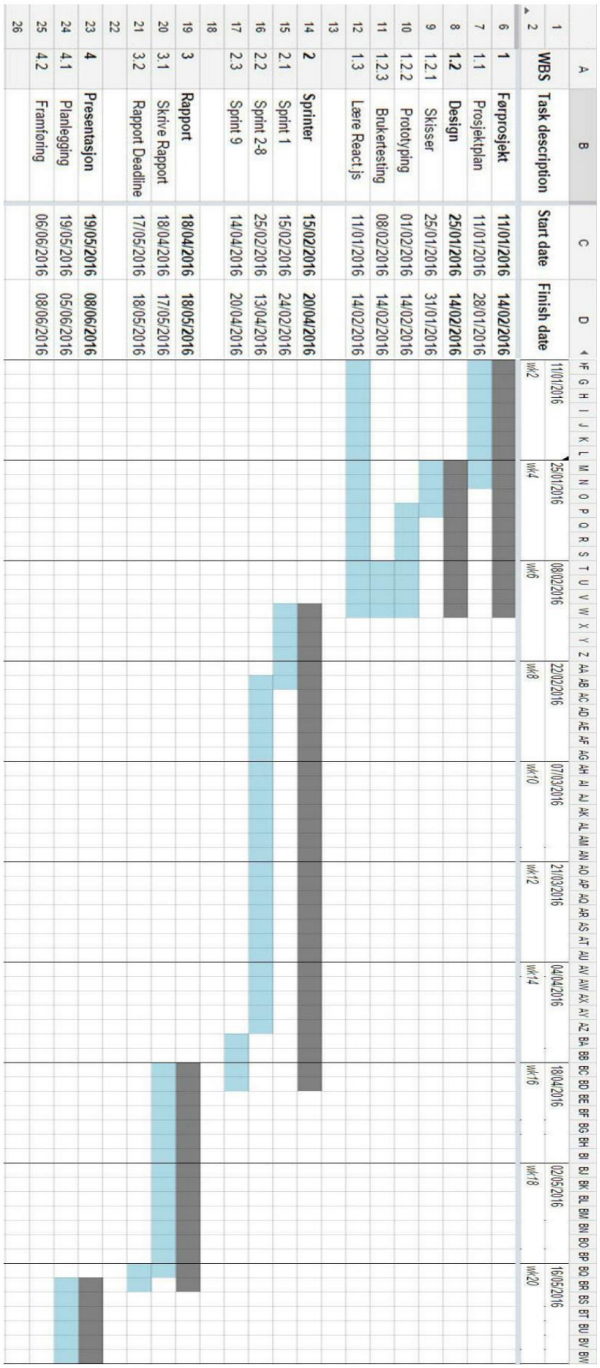
Skype - programmet som kommer til å bli brukt for å gjennomføre møter med arbeidsgiver. Skype gir oss mulighet til å dele skjermen slik at vi enkelt kan vise hva vi har gjort.

YouTrack - et programvare levert av JetBrains som arbeidsgiveren vår bruker. Her kan både bedriften og gruppa notere ting som må gjøres eller fikses i prosjektet. YouTrack vil hjelpe oss med organisering av arbeidsoppgaver og gi oss en oversikt over hva som må bli gjort med deadline og prioritering.

Microsoft Word - tekstbehandlingsprogrammet vi skal bruke til å formatere den siste versjonen av rapporten og prosjektplanen.

Webstorm - programvare som blir brukt til programmering av katalogen. Programmet er i hovedfokus rettet mot utvikling i JavaScript.

6. Plan for gjennomføring



WBS	Task description	Start date	Finish date
1	**Førprosjekt**	**11/01/2016**	**14/02/2016**
1.1	Prosjektplan	11/01/2016	28/01/2016
1.2	**Design**	**25/01/2016**	**14/02/2016**
1.2.1	Skisser	25/01/2016	31/01/2016
1.2.2	Prototyping	01/02/2016	14/02/2016
1.2.3	Brukertesting	08/02/2016	14/02/2016
1.3	Lære React.js	11/01/2016	14/02/2016
2	**Sprinter**	**15/02/2016**	**20/04/2016**
2.1	Sprint 1	15/02/2016	24/02/2016
2.2	Sprint 2-8	25/02/2016	13/04/2016
2.3	Sprint 9	14/04/2016	20/04/2016
3	**Rapport**	**18/04/2016**	**18/05/2016**
3.1	Skrive Rapport	18/04/2016	17/05/2016
3.2	Rapport Deadline	17/05/2016	18/05/2016
4	**Presentasjon**	**19/05/2016**	**08/06/2016**
4.1	Planlegging	19/05/2016	05/06/2016
4.2	Framføring	06/06/2016	08/06/2016

7. Referanser

Scrum Alliance Inc - 2012 | Agile Atlas Core Scrum

http://agileatlas.org/images/uploads/CoreScrum_NO.pdf

Vedlegg C

Uke 1

- Avtalt møte med arbeidsgiver den 13.01.2016

Uke 2

11.01 Bachelor Lynkurs 10.00-12.00

- Hvordan skrive prosjektplan
- https://fronter.com/hig/links/files.phtml/1602890066$71119491$/FagstoffRessurser/01+Lynkurs+og+informasjonsm_prcent_F8te/lynkurs1_bopg_v_prcent_E5r2016_infomatikk_prosjektplan.pdf

12.01 Møte 14:00-16:00

- Satt opp prosjektplan dokumentet
- Satt opp referat dokumentet
- Kontaktet veileder for videre planlegging av samarbeidet
- Satt opp dokumentet med viktige med linker

Uke 3

19.01 Møte 13:00-14:00

- Begynte å tenkte på prosjektplan, notert hva som må bli med i planen

20.01 Møte 11:00-12:40

- Skulle møte med veileder, men borte pga sykefravær
- Leste på gamle oppgaver for å få oversikt hvordan ting skal bli gjort
- Notert litt mer om prosjekt planen for å ha vite hva vi skal skrive om

Uke 4

25.01 Møte 11:00-15:00

- Felles skriving på prosjektplanen - risikoanalyse og bakgrunn
- Diskutere hva vi må ta opp på møtet med Emil for å fullføre prosjektplanen

26.01 Møte 12:30-18:30

- Felles skriving på prosjektplanen - Fylle ut punkter som mangler
- Lagde gantskjema
- Møte med veileder ble utsatt til 27.01. Forbereder det mer

27.01 Møte 13:00-19:00

- Skype møte med veileder

- Kommentarer rundt prosjektplanen fra veileder rundt hva som må fikes
- Kontaktet arbeidsgiver for å høre om forskjellige ting som vi skulle ha i prosjektplanen
- Retting opp av prosjektplanen
- Skriving av ting som vi har manglet i prosjektplanen
- Oppdatering av Gantt-skjema

28.01 Møte 12:30-18:00

- Prosjektplanen - innlevering
- Gikk gjennom hele planen og rettet opp feil
- Formatering i Word

Uke 5

01.02 Møte 11:00-16:00

- Tegnet skisser
- Begynte på prototyping med proto.io
- Diskusjon rundt design
    - Hvordan blir data fremstilt
    - Hvordan ting vil fungere
- Analyse av datasett i JSON fra SIC AS

02.02 Møte 11:00-16:00

- Satt opp første prototype i proto.io
- Diskusjon rundt design, valg av farger osv.
- Fått fargepalett fra SIC AS
- Sendt prototype til bedriften for å få tilbakemelding

03.02 Møte 11:30-16:00

- Analysere ting rundt prototypen
- Kartlegging av funksjoner som må implementeres i katalogen (programmering)
- Kartlegging av ting som vi kan skrive om i rapporten
- Satt opp repository på BitBucket
- Installert React.js og Material UI for prosjektet

Uke 6

09.02 Møte 11:30-16:00

- Videre planlegging av design prosessen
- Installert node.js på pcene
- Installasjon av React og Material-UI for prosjektet

10.02 Møte 11:00-16:30

- Fikk fiksa prosjektet slik at alt fungerer i browser
- Ble bedre kjent med node.js/npm og bruk av modules

- Testing av React og Material-UI om det fungerer på riktig måte nå

11.02 Møte 11:00-16:00

- Vidre testing av React og Material-UI
- Testing av forskjellige npm packages

Uke 7

16.02 Møte 11:00-16:00

- Felles arbeid med Material-UI og React

17.02 Møte 11:00-16:00

- Satt opp lista over packages som vi trenger i prosjektet
- Satt opp package.json med riktig innstillinger og "scripts" for forbedre arbeid

18.02 Møte 11:00-16:30

- Testet om ting fungerer på riktig måte
- Satt opp ny repository på BitBucket med alle packages vi trenger
- Problemer med ny SourceTree
    - Mange bugs i programmet som stoppet arbeidet
    - Bitbucket via terminal

19.02 Møte 11:30-16:30

- Begynte å notere ting som vi må skrive om i rapporten
- Fant kilder på ting som vi skal skrive om

Uke 8

22.02 Møte 12:20-16:30

- Snakket med bedriften om dataoppsettet som vi kommer til å bruke i katalogen
    - Det vil følge med forklaring hvordan vi kan generere bilder i forskjellige fargeverianter
    - Snakket hvordan det går med prosjektet
    - Videretesting av React og Material-UI

23.02 Møte 11:00-16:00

- Prøvde å få sidebar til å fungere
- Noterte kilder som vi kan bruke i rapporten på ting som vi tester ut nå

24.02 Møte 12:00-16:00

- Fikk endelig sidebar til å fungere
- Koblet det opp med App Bar
- Brainstorming rundt dataoppsett i JSON format

25.02 Møte 11:00-16:30

- Videretesting av forskjellige components fra Material-UI
- Satt opp plan med hva som vi må ha på plass
- Leste om JSON for å ta dataoppsett i bruk

26.02 Møte 12:00-16:00

- Jobbing med GridList og JSON dataoppsett

Uke 9

29.02 Møte 12:00-17:30

- Jobba rundt med å sette opp dialog på GridList
- Leste på React og JSON hvordan vi går generere GridList

01.03 Møte 11:00-16:00

- Bytta utt GridList med Cards, flere muligheter, mer oversiktlig
- Analyse av dataoppsett som vi fikk fra Sic AS

02.03 Møte 11:00-16:00

- Jobbing rundt dialogen
    - Dialog som henter ut data fra JSON fil

03.03 Møte 11:00-16:00

- Fiksa endeling dialogen
- Lagde en plan
    - Må bli ferdig med prototype i React
    - Mulighet for å jobbe videre med prototypen
    - Noe som kan vises fram
    - Lærer koding, jobber samtidig med design

Uke 10

07.03 Møte 11:00-16:00

- Satt opp rapport dokumentet
- Kobla JSON fila til React
- Diskusjon rundt møte med veileder

09.03 Møte 12:15-14:00

- Lynkurs om rapportskriving
    - Notater med viktig info
- Snakket med arbeidsgiver
    - Avtalt workshop etter påske
    - Fikk ny dataoppsett med bilder
        - Bedriften skal gjøres den om til JSON
- Satt opp liste med ting som må gjøres i programmering

10.03 Møte 10:15-11:30

- Lynkurs om rapportskriving
- Avtalt ny dato for veiledning, tirsdag 15.03 klokka 10:00
    - Spørsmål rundt rapportkskriving

11.03 Møte 11:00-17:30

- Jobber med ting som må fikses i programmering
- Spørsmål til møte med veileder

Uke 11

14.03 Møte 11:00-17:00

- Sortering av produkter
- Fikk dbprodukt.json fra SIC AS sammen med GIMP bilder som kan redigeres
- Snakket med bedriften om dbprodukt.json

15.03 Møte 11:00-17:30

- Møte med veileder
- Kobla opp dataoppsettet
- Fant ut hvordan bildesystemet fungerer
    - Lagde en del dummy bilder som vi kan bruke for testing

Uke 13

30.03 Møte 11:00-16:00

- Satt opp planen for hva som må gjøres i katalogen
    - Halvard - Sortering og filtrering
    - Maks - Style dialog og hente data til dialog
- Begynte å skrive på rapport
    - Halvard - npm og SourceTree problemer

31.03 Møte 11:00-16:00

- Rapport skriving
    - Maks - package.json i npm
- Utvikling av sortering og filtrering
    - Brainstorming rundt hvordan vi fikser filtrering med flere filtrering muligheter samtidig

01.04 Møte 11:00-16:00

Uke 14

04.04 Møte 11:00-16:00

- Var i kontakt med bedriften ang. møte/workshop
    - Fikk hjelp med å hente ut data fra json fila
        - Dbproduct_keyed.json
- Jobbing med oppgavene som vi har fordelt mellom hverandre

05.04 Møte 11:00-17:00

- Kontaktet veileder og avtalt møte torsdag 07.04 kl 12:00-12:30
- Prøve å finne ut hvordan vi kan hente ut data til dialogen som f. eks alle størrelser for et produkt

06.04 Møte 12:00-17:00

- Prøvde ut katalogen i mørke farger
    - Brukte standard farger som følger med Material-UI
        - Vi mener at katalogen så ut bedre med lyse ferger
    - Kan bli bedre når vi får farger fra bedriften
- Jobbing med filtrering som vil fungere via checkboxes

07.04 Møte 11:00-17:00

- Fikset forloop som går gjennom ting i json fila for å hente ut farger for et produkt i dialogen
- Møte med veileder ang. rapportskriving og videreutvikling av prosjektet
- Jobbing med filtrering
- Begynte å gå gjennom gamle bacheloroppgaver for å finne ut oppsettet for rapporten

08.04 Møte 12:00-16:00

- Fikset henting ut av størelser for et produkt
    - Brainstorming rundt henting av data til dialog
      Kundene skal i utgangspunktet bestille flere størellser og farger av et produkt (må høre med bedriften)
- Fikset filtrering for kjønn og farger
- Ryddet koden etter merge i SourceTree

Uke 15

11.04 Møte 12:00-17:00

- Sortering med seasons
- Jobber med å hente ut seasons ut i fra hvilken farge man har valgt for et produkt
- Planlegging til møte med bedriften

12.04 Møte 11:40-17:00(+reise)

- Møte i Oslo med bedriften
- Diskusjon rundt koden
    - Splitting av main.js til mindre filer
- Avklaring hvilken sortering er det vi trenger
- Snakk om vi skulle ha søkefeltet
- Fant ut hva som vi må hente ut fra json fila til dialog og cards
- Flytting av sortering og filtering til egen div inni katalogen i stedet for å ha det i NavBar
    - Bedriften skal ha sin egen NavBar for hele systemet

14.04 Møte 12:00-20:00

- Begynte på ting som må gjøres som vi har avklart med bedriften
- Katalogen genererer en card for hver product color
- Printer ut seasons til dialogen
    - Små ting som må fikses med duplikasjon

15.04 Møte 11:00-18:00

- Fiksa duplikasjon på seasons
- Jobber med å legge til star icon for favourit funksjonen
    - Fikk lagd star icon, styling rundt hele dialogen
- Prøver å fikse sortering og filtrering mot nye cards

Uke 16

18.04 Møte 11:30-17:00

- Endra på productdb.json
    - Flytta size-groups til color av et produkt siden hver size for et color produkt har samme size-group
- Prøver å få filtrering til å fungere
- Avtalte TeamViewer med bedriften på tirsdag (19.04.2016)
- Henter ut size-group til cards

19.04 Møte 12:00-20:00

- Flytta sortering og filtrering til applikasjonen
- Henter ut list og unit pris til card
- Begynte på styling rundt cards og left-nav (sortering og filtrering)
- Delte opp class/component LeftNav til ApplicationBar og LeftNav
- Flyttet på js filer til src/javascript og json til /src/json
    - Prøver å splitte components fra main.js
- Fikk oppdatert json fila for å fikse filtrering

20.04 Møte 12:00-17:00

- Splita main.js til flere filer slik at main.js er ikke en lang wall of text
- Fikk forslag fra bedriften hva vi skal endre på i katalogen
- Fjernet size-group fra cards
-

21.04 Møte 12:00-20:00

- Prøver å fikse filtrering
- Må hoppe tilbake til en main.js fil for å prøve seg på filtrering og sortering
- Koblet opp ny json fila som vi har fått fra bedriften
- Fiksa små problemer som har oppstått under endring av productdb.json
- Begynte å hente ut prisen til dialogen
- Fjernet seasons fra Cards
- Begynte å sette opp filtrering etter ønske til bedriften
    - Dropdown meny for hver filtrering eller sortering
- Stylet pris på cards

22.04 Møte 12:00-19:00

- Ble ferdig med alle bilder for hver product color
- Fortsetter med styling

Uke 17

25.04 Møte 12:00-20:00

- Begynte å skrive på rapporten
- Rapportorganisering
- Gikk gjennom tidligere bacheloroppgaver

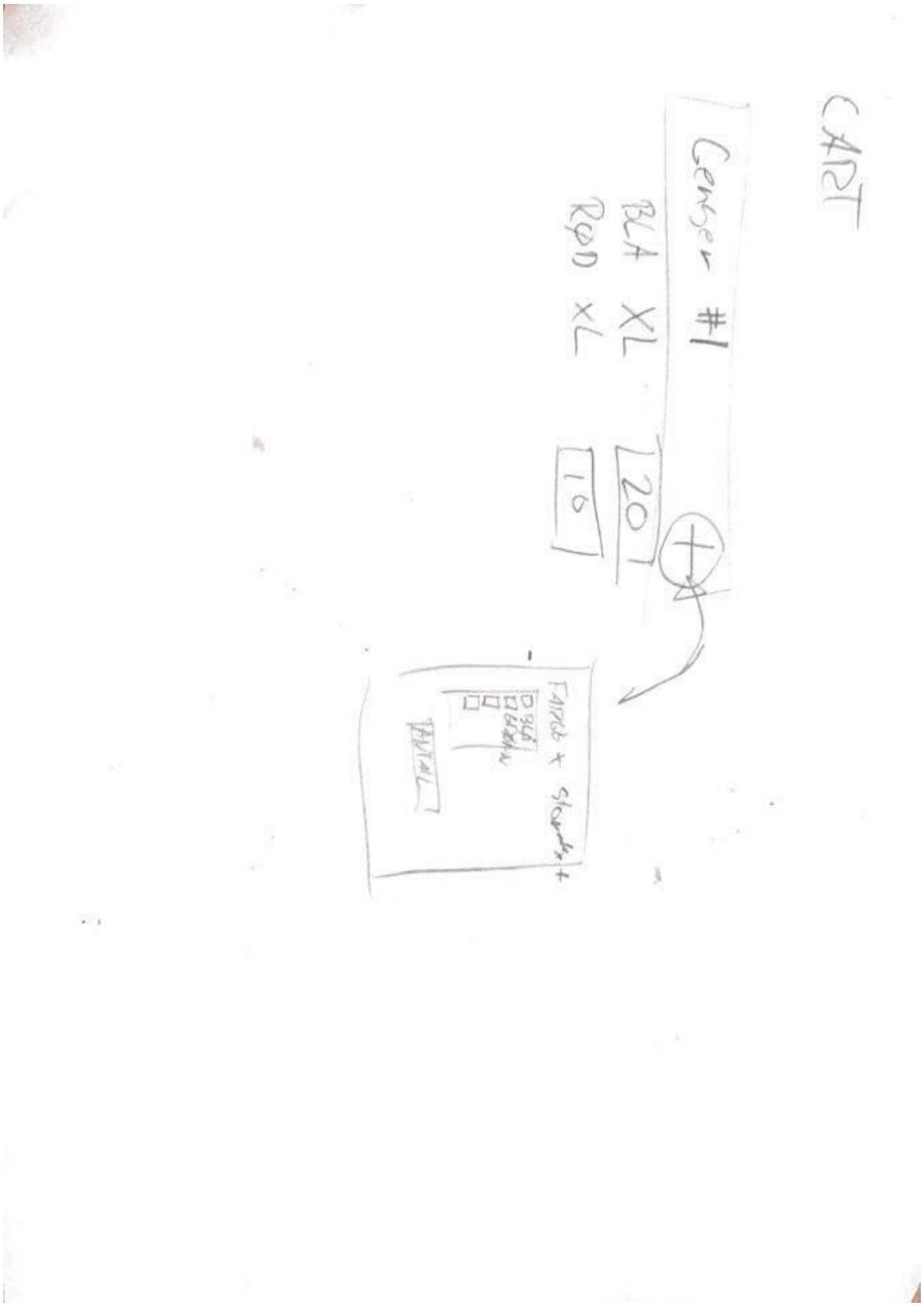
26.04 - 18.05

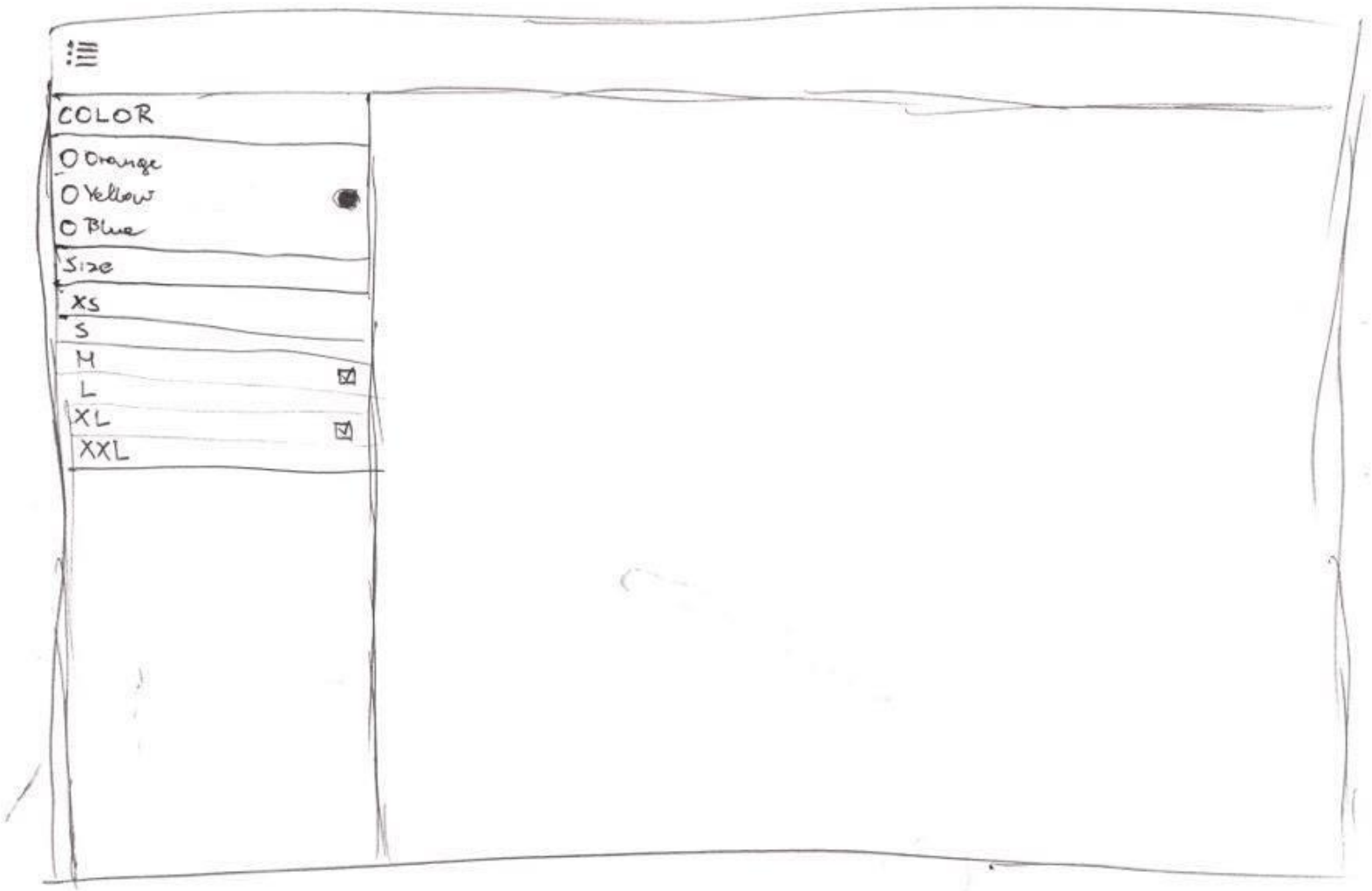
Rapportskriving

Vedlegg D

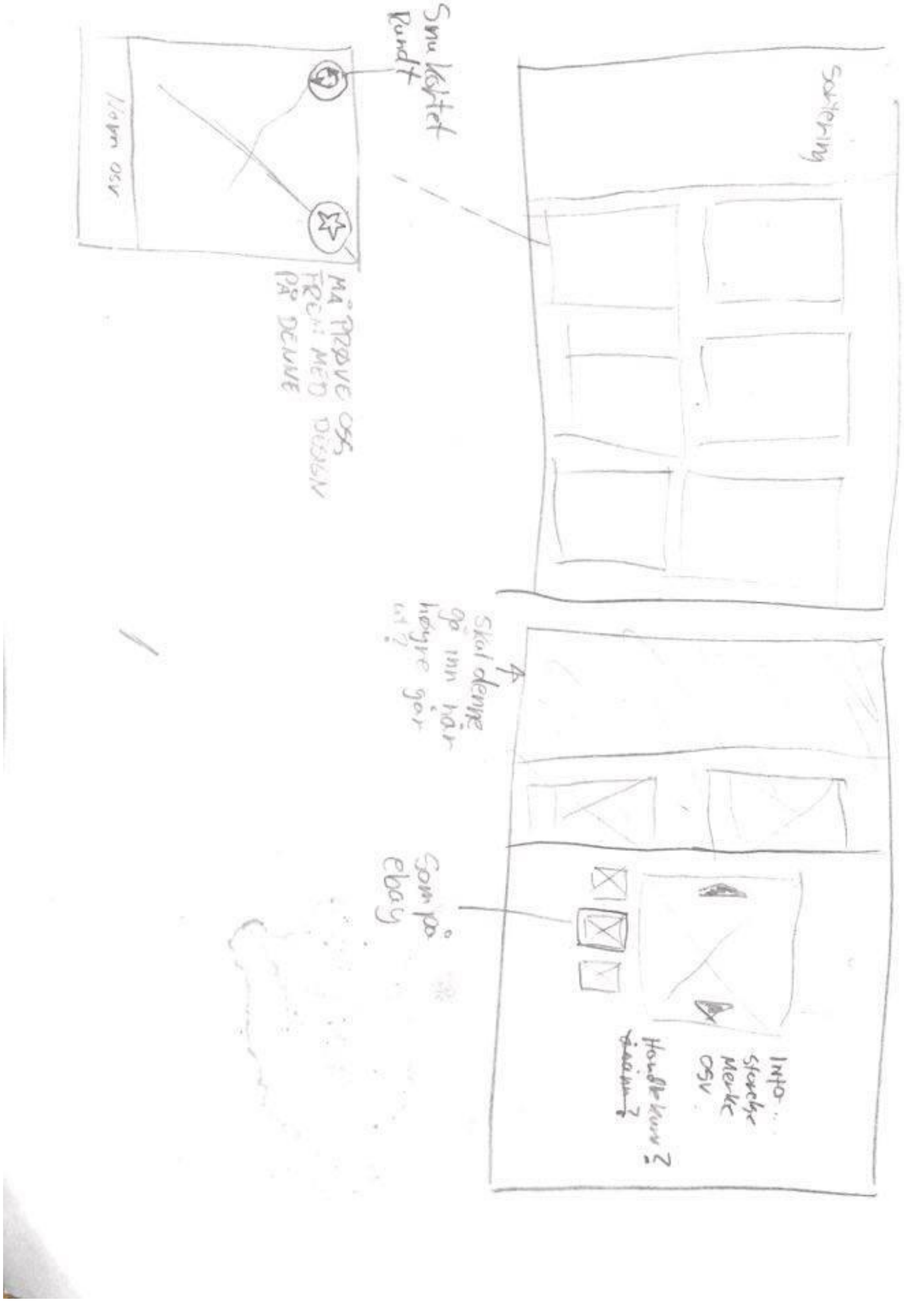
- **Produktbaklogg** - liste over funksjoner som skal utvikles i katalogen
- **Datorder** - navnet til klientens systemet
- **JSON -** står for JavaScript Object Notation. En JSON fil består av objekter med tabeller, hvor man kan lagre informasjon som kan bli hentet ved hjelp av JavaScript.
- **NPM -** Node Package Manager
- **DOM -** Document Object Model, en beskrivelse hvordan HTML skal se ut som kan bli endret ved bruk av JavaScript
- **Commit -** velge filer som vi skal pushe og gi en kommentar
- **Push** - laste opp endringene vi har gjort i repository
- **Main.js -** hovedfil hvor ligger hele koden til katalogen som vi har skrevet ved hjelp av React, Material-UI og JavScript
- **Bundle.js –** JavaScript fila katalogen bruker. Denne blir lagd av Browserify og generert ut ifra main.js
- **For loop** – en funksjon som går gjennom hver objekt i et array
- **If statement -** et utsagn som er true eller false. Hvis true så funksjonen vil gjøre noe
- **Kort –** hvordan et produkt blir vist i liste over produkter. Navnet kommer fra Material komponenten Card.

Vedlegg E

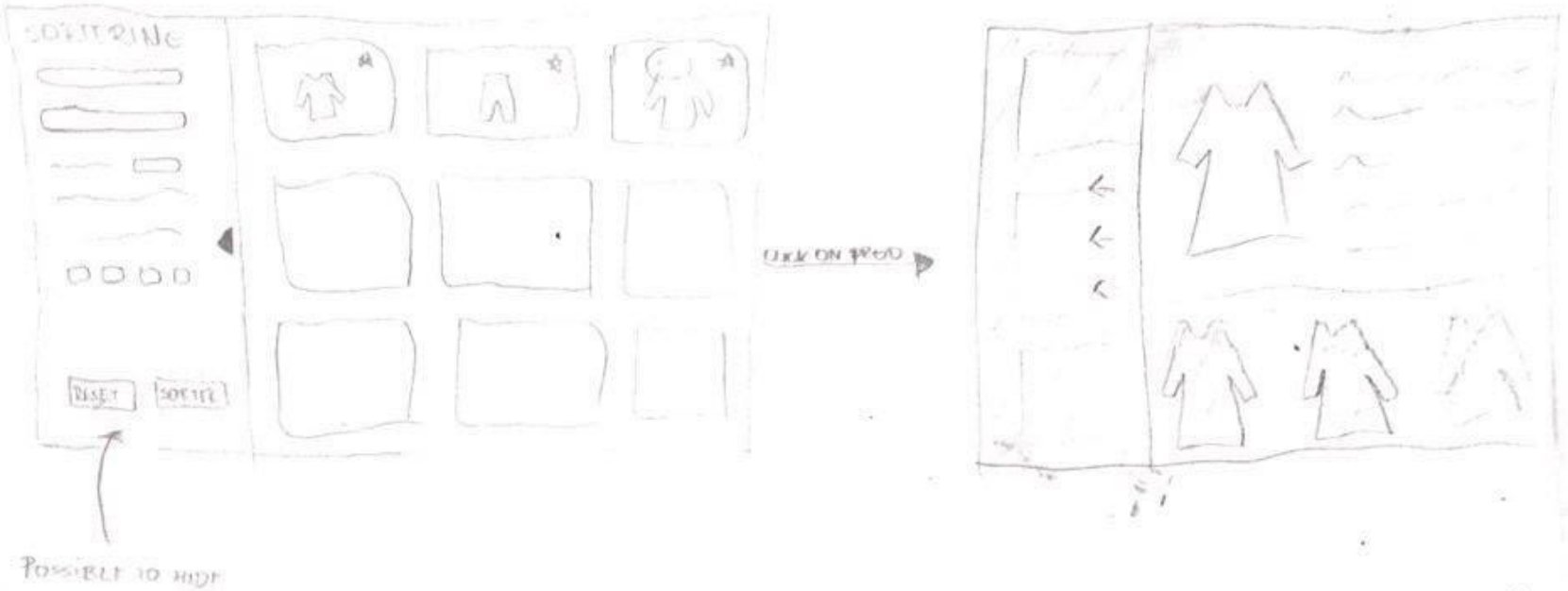



COLOR
Orange
Yellow
Blue
Size
XS
S
M
L
XL
XXL

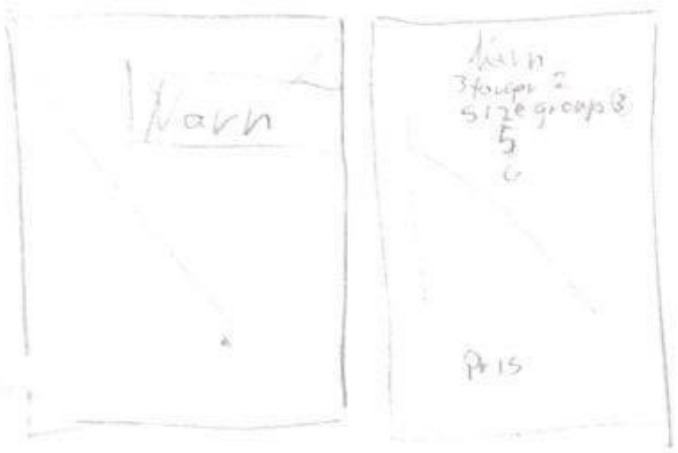

Sesong
SS 16
FW 16
SS 17
PRODUKT GRUPPE
Klær
Sko
Accessories
Bags
Kjønn
Menn
Dame
Klær
Jakker
Gensere
Bukser
Kjole
Sko
Accessory
klær
Jakker
Sko
Accessorys



Sortering
Zoom osv
MÅ PRØVE OG FREM MED DESIGN PÅ DENNE
Som på ebay


RESET
SORTER
CLICK ON PROD
POSSIBLE TO HIDE

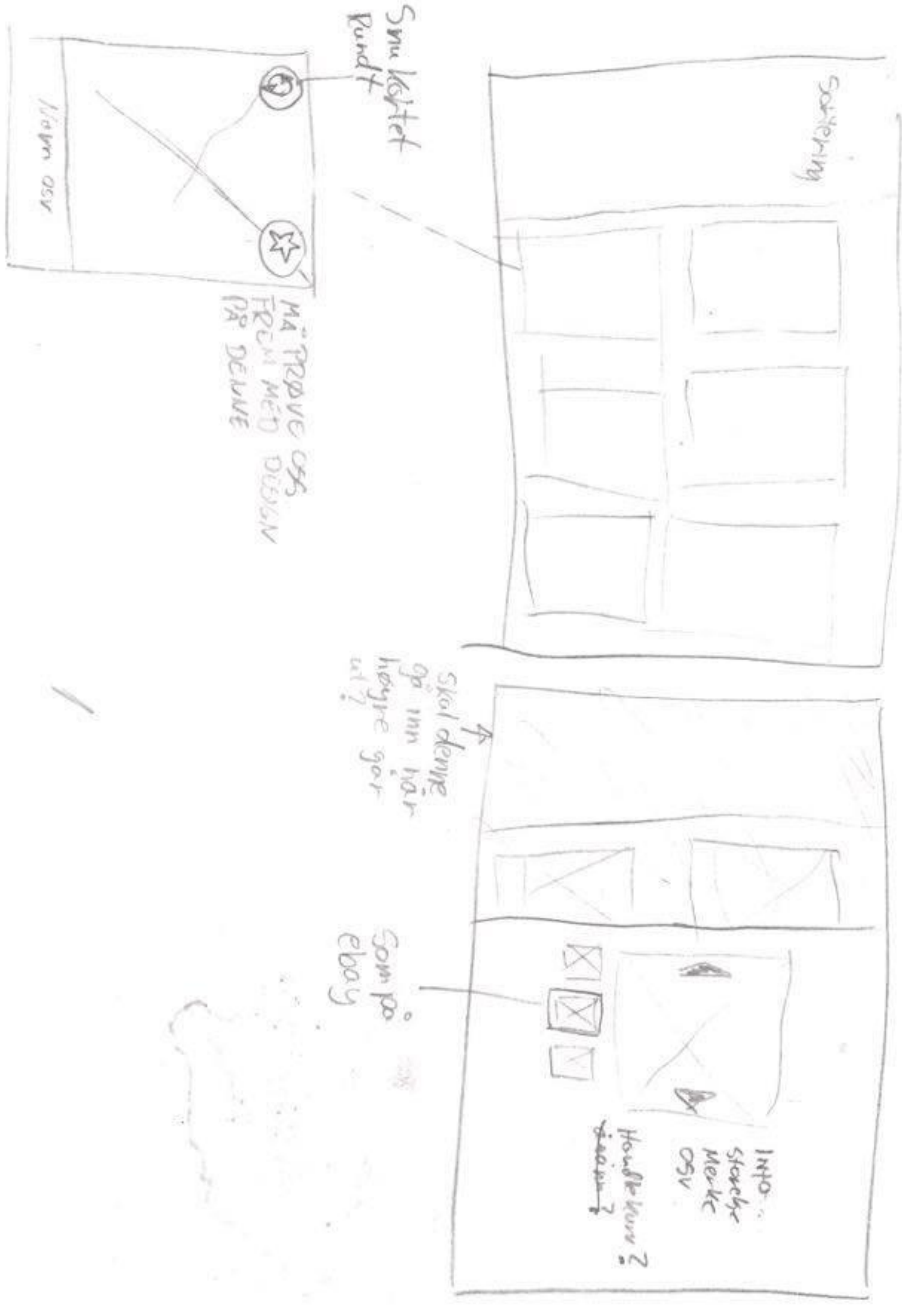
Produkt



Google Mat Design box

Innholder
1 Bilde
2 Antall farger
3 Antall Størrelser
4 PRIS
5 Sesong
6 Blond
7 Kjønn

Eventuelt piler til bytte farge/størrelse


Sortering
Som på ebay

Vedlegg F

Koden ble lagt ved som .zip fil ved innlevering. For å åpne prosjektet, start index.html i nettleseren.

Vedlegg G

Figur 1 (Skjermbilde): Material-UI, App Bar [Internettside] Sitert 17.05.2016. Tilgjengelig fra: http://www.material-ui.com/#/components/app-bar

Figur 2-3 (Skjermbilde): Proto.io, Skjermbilder av prototypen [Internettside] Sitert 17.05.2016. Tilgjengelig fra: https://shabbyabby.proto.io/share/?id=b04791d2-60aa-4152-9303-515275c5c189&v=1

Figur 4 til 25 er gruppens egne skjermbilder, figurer og modeller av katalogen