Oliver Nilssen
Silje Tanemsmo
Joakim Tronseth

Bacheloroppgave

Utvikling av Android applikasjon for trafikklærere

Med rammeverket React Native

Mai 2021

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Bacheloroppgave

2021

NTNU

Kunnskap for en bedre verden

Oliver Nilssen
Silje Tanemsmo
Joakim Tronseth

Utvikling av Android applikasjon for trafikklærere

Med rammeverket React Native

Bacheloroppgave
Mai 2021

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Rapport

NTNU
Kunnskap for en bedre verden

Utvikling av Android-applikasjon for trafikklærere

Med rammeverket React Native

VERSJON

1.0

DATO

20.05.2021

FORFATTERE

Oliver Elias Nilssen
Joakim Heitmann Tronseth
Silje Tanemsmo

ANTALL SIDER OG VEDLEGG

153 sider

7 vedlegg

SAMMENDRAG

Dette dokumentet representerer bacheloroppgaven som er utført av gruppe 087 på linjen *Informatikk med spesialisering i drift av datasystemer* på NTNU. Problemstillingen for oppgaven var å utvikle et verktøy på Android enheter for å hjelpe trafikklærere å effektivisere arbeidet sitt i læring av elever. Oppgaveløsningen blir i dette dokumentet representert i form av dokumenter som er utarbeidet i løpet av prosjekttiden, 7. januar 2021 til 20. mai 2021.

Prosjektgruppen har utviklet en applikasjon, illusTrafikk, som kan lastes ned gratis via Google Play Store og benyttes for å illustrere trafikksituasjoner under trafikkopplæring. Bacheloroppgaven er utviklet ved bruk av rammeverket React Native og kan kjøres på de fleste Android enheter, men hvor hovedfokuset har vært på større enheter som nettbrett.

Dokumentene som er produsert i løpet av prosjekttiden blir presentert i denne samlede PDF filen. Dette dokumentets innholdsfortegnelse vil kunne navigere leseren til hver av disse dokumentene, som også er tilgjengelig individuelt som vedlegg. Dokumentasjonen som blir representert i denne PDF-filen er: Forstudierapporten, kravdokumentet, driftsrapporten og sluttrapporten. Alle vedlegg som tilhører dette dokumentet og de individuelle delene i dette dokumentet ligger i en egen separat ZIP-fil. Kildekoden som er skrevet og utviklet av prosjektgruppen ligger også tilgjengelig via et repository på nettet og i ZIP-filen hvor alle vedlegg ligger.

NTNU
Kunnskap for en bedre verden

Historikk

VERSJON	DATO
1.0	20.05.2021

VERSJONSBESKRIVELSE

Ferdigstilt versjon av oppgaverapporten. Spesifikk revisjonshistorie for de ulike delene (delrapportene) er spesifisert i starten av hver delrapport.

INNHOLDSFORTEGNELSE

FORSTUDIERAPPORT

BACHELORPROSJEKT 087: UTVIKLING AV ANDROID-APPLIKASJON FOR TRAFIKKLÆRERE

Oliver Elias Nilssen, Joakim Heitmann Tronseth og Silje Tanemsmo

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
29.01.2021	0.1	Første utkast	Joakim H. Tronseth, Oliver E. Nilssen og Silje Tanemsmo
29.04.2021	0.2	Retting av skrivefeil	Silje Tanemsmo
15.05.2021	1.0	Endelig versjon.	Joakim H. Tronseth, Oliver E. Nilssen og Silje Tanemsmo

INNHOLDSFORTEGNELSE

1. INTRODUKSJON – HENSIKTEN MED DOKUMENTET

Hensikten med dette dokumentet er å få en felles forståelse av prosjektet og hva det innebærer for alle berørte. Med utgangspunkt i dokumentet skal både vi som skal gjøre jobben (prosjektgruppen) og oppgavestiller/oppdragsgiver få en samstemt oppfatning av hva dette prosjektet dreier seg om og hva som kommer til å skje fremover, både tidsmessig og resultatmessig.

Dette er et bachelorprosjekt i studiet Bachelor i Informatikk med spesialisering i drift av datasystemer ved NTNU. Oppgaven består av å utvikle en applikasjon som trafikklærere kan benytte som et verktøy i kjøreopplæringen.

I dette dokumentet belyser vi og gir svar på følgende spørsmål:

- Kapittel 2 - Hva som er bakgrunnen for at prosjektet er startet
- Kapittel 3 - Hva som skal gjøres i dette prosjektet, hvor lang tid det tar og hvor mye det vil koste (prosjektmål og prosjektplaner)
- Kapittel 4 - Hvem som er interessentene for prosjektet og hvilke suksesskriterier disse har, samt hvilke rammebetingelser prosjektet er underlagt
- Kapittel 5 – Kritiske suksessfaktorer for at prosjektet skal lykkes
- Kapittel 6 – Hvilke utfordringer og risikoer prosjektet vil møte, og hvordan prosjektgruppen skal håndtere disse
- Kapittel 7 – Tar for seg lønnsomheten av prosjektet ved å sette forventet nytte opp mot forventede kostnader
- Kapittel 8 – Teknologivalgene som er tatt internt i gruppen for å forsikre god arbeidsflyt
- Kapittel 9 – Ulike retningslinjer og standarder til prosjektet og prosjektgjennomføringen
- Kapittel 10 – Hvem som skal gjøre arbeidet og hvem som styrer arbeidet

1.1. FORKORTELSER OG BEGREPER

React Native – Rammeverk for å utvikle plattformavhengige applikasjoner for Android og iOS. Kombinerer plattformavhengig utvikling med React, et JavaScript-bibliotek for å bygge brukergrensesnitt [1].

MVP (Minimum Viable Product) – Innenfor produktutvikling er det det produktet som gir størst avkastning på investering sett i forhold til risikoen [2].

Alpha-produkt – En veldig tidlig versjon av en programvare som ikke nødvendigvis inneholder alle funksjonene som den endelige versjonen er planlagt å ha. En alpha-versjon er ufullstendig eller bare delvis fullstendig [2].

2. BAKGRUNNEN FOR PROSJEKTET

Bedriften Sopra Steria Trondheim ønsket en gruppe bachelorstudenter til å gjøre en utviklingsoppgave som bachelorprosjekt, det var opp til studentene selv å komme opp med en ide til applikasjon. Samtidig ønsket en trafikklærer hos Way trafikkskole en applikasjon for å kunne illustrere trafikksituasjoner for sine elever, dette ble da vårt forslag til utviklingsoppgave og oppgaven ble godkjent.

2.1. BESKRIVELSE AV PROBLEMER OG BEHOV

Utfordringer/problemstillinger med dagens løsning:

- Illustrering av trafikksituasjoner med penn og papir blir lite presist. Skal man for eksempel illustrere en situasjon i et veikryss må man først tegne opp selve veikrysset, deretter veioppmerkinger, og til slutt selve situasjonen. Dette blir lite presist. Skal man gjøre dette detaljert for hånd, vil det ta for lang tid.
- Forskjeller i «tegneferdigheter» blant trafikklærere kan gjøre det utfordrende for elevene å forstå hva som forklares.
- Papir som benyttes for å illustrere på, forsvinner og man får ikke mulighet til å gå tilbake og se hva man har gjort før (gjenbruke vanlige situasjoner).
- Det kan ta lang tid å illustrere en situasjon med penn og papir slik at eleven forstår det.
- Tap av tid og penger for trafikkskolen på grunn av manglende system for illustrering av trafikksituasjoner.

Muligheter programvaren kan gi:

- Ferdigdefinerte «trafikksituasjoner» man kan ta utgangspunkt i, som reduserer tiden som benyttes til å tegne opp situasjonen. Skal man for eksempel illustrere en situasjon i et veikryss, får man raskt opp et ferdig opptegnet veikryss med riktige veioppmerkinger o.l., som man deretter kan tegne over.
- Ferdigdefinerte komponenter som biler, sykler, fotgjengere o.l. man kan sette inn i «trafikksituasjonen» - Reduserer tiden det tar å tegne, pluss at det standardiserer illustreringen slik at det blir mer forståelig for elevene (for eksempel mot at en lærer bruker sirkel for å illustrere en bil, mens en annen benytter en trekant).
- Trafikkskilt som kan plasseres ut i trafikksituasjoner, gjør det mulig for trafikklæreren å illustrere komplekse situasjoner på en kjapp og oversiktlig måte.
- Ved å gi mulighet til å også tegne selv, over de ferdigdefinerte trafikksituasjonene, kan man enkelt illustrere akkurat den spesifikke situasjonen.
- Mulighet til å lagre illustrasjonen gjør at man kan gå tilbake og gjenbruke det man har illustrert tidligere

2.2. KORT OM DAGENS SYSTEMER OG RUTINER

Det eksisterer pr dags dato ikke et formelt system for illustrasjon av trafikksituasjoner, og det er opp til hver enkelt trafikklærer hvordan man velger å gjøre dette. Trafikklæreren vi har vært i kontakt

med bruker penn og papir for å illustrere tenkte trafikksituasjoner som hen benytter i sin undervisning. Disse tegningene brukes både i klasserommet og under kjøretimene. Tegningene må utføres kjapt og presist og det er ikke en enkel oppgave når man sitter i en bil. Det eksisterer en applikasjon for iOS med funksjoner for trafikkundervisning, men den ble sist oppdatert i 2017. Den er med andre ord veldig utdatert både i design og funksjonalitet. Et annet problem er at trafikklærerens bedrift i hovedsak benytter seg av Android-baserte nettbrett i sin undervisning, og de kan dermed ikke benytte seg av den allerede eksisterende applikasjonen uansett.

3. PROSJEKTMÅL

Prosjektets mål er grunnlag for:

- Å bli enige med oppdragsgiver om hva som skal bli resultatet av prosjektet
- Å kunne planlegge og styre
- Å kunne vurdere i ettertid om resultatet av prosjektet ble som planlagt
- Å få en felles forståelse i prosjektgruppa for hva jobben går ut på

Prosjektmålene er formulert med utgangspunkt i de problemene og behovene vi har kartlagt og beskrevet i kapittel 2 om bakgrunnen for prosjektet. Enkelte av målene tar også utgangspunkt i generelle krav til bacheloroppgaven og hva prosjektgruppen ønsker å få ut av oppgaven.

3.1. EFFEKTMÅL

Effektmålene beskriver den ønskede effekten av oppgaven og prosjektet. Vi har satt oss følgende effektmål:

- Forenkle måten trafikklærere illustrerer trafikksituasjoner for sine elever
- Forbedre utbyttet av trafikkopplæringen ved at illustrering av trafikksituasjoner blir mer forståelig for elevene

3.2. RESULTATMÅL

Resultatmålene beskriver hva som konkret skal foreligge som resultat når prosjektet er ferdig. Vi har satt oss følgende resultatmål:

- Ha utviklet en Android-applikasjon i løpet av prosjektperioden, der trafikklærere kan illustrere ulike trafikksituasjoner
- Publisere applikasjonen på Google Play Store

3.3. PROSESSMÅL

Siden dette er et studentprosjekt, anser vi at det er hensiktsmessig å formulere mål som ikke er direkte knyttet til det prosjektet skal produsere, men til den prosessen vi skal gjennom. Vi har satt oss følgende mål for den ønskede effekten prosjektarbeidet skal ha på prosjektdeltakerne:

- Erfaring og økt kompetanse om smidig utvikling og spesifikt Scrum
- Erfaring og økt kompetanse om applikasjonsutvikling med rammeverket React Native
- Erfaring med fjernsamarbeid
- Oppnå karakter B eller bedre som resultat i emnet IDRI3001 Bacheloroppgave i drift av datasystemer

3.4. PROSJEKTETS OMFANG

Prosjektet skal handle om å utvikle en applikasjon for trafikklærere, som de kan velge å benytte i opplæringen for å enklere illustrere ulike trafikksituasjoner for elevene. Applikasjonen skal fungere på enheter som kjører Android operativsystem.

Hva prosjektet ikke skal handle om:

- Utvikle en applikasjon som kan selges – Vi vil ha fokus på å i første omgang utvikle en MVP eller eventuelt en beta og gjøre denne tilgjengelig i Google Play Store.
- Det finansielle aspektet i en utviklingsoppgave. Dette er en studentoppgave, og vi har ingen økonomiske ressurser å benytte oss av.
- Hva trafikkopplæring består/skal bestå av – Vi skal kun utvikle et verktøy trafikklærere kan benytte seg av.
- Utvikling av applikasjon for enheter som kjører iOS
- Prosjektet skal ikke drive brukeropplæring

3.5. Produktets funksjonelle egenskaper

Produktet bør ha følgende funksjonelle egenskaper for å løse de utfordringene med dagens situasjon, som ble beskrevet i kapittel 2.1 Beskrivelse av problemer og behov.

Problem	Berører	Konsekvens	Vellykket løsning
Illustrering med penn og papir blir lite presist	Elever Trafikklærere	Utfordrende for elevene å forstå hva som illustreres. Skaper rom for misforståelser mellom trafikklærer og elev, som kan føre til farlige situasjoner i trafikken.	Ferdigdefinerte «situasjoner» å ta utgangspunkt i som inneholder detaljert og korrekt veioppmerking og skilting, som man deretter kan tegne på/over.
Forskjeller i tegneferdigheter blant trafikklærere	Elever Trafikkskolen	Utfordrende for elevene å forstå hva som illustreres (særlig i ettertid). Bruker mye av tiden på forklaring (av elevenes tid, som de betaler for). Tap av omdømme for trafikkskolen hvis de får mange misfornøyde elever.	Ferdigdefinerte «trafikksituasjoner» å velge i. Eks rundkjøring, lyskryss, veikryss, landevei o.l. Ferdigdefinerte komponenter som biler, syklister og fotgjengere som man kan «dra inn» i bildet. Standardisering av illustrasjoner gjør det enklere for eleven.
Papir som benyttes for å illustrere på, forsvinner	Trafikklærere	Får ikke mulighet til å gjenbruke tidligere arbeid, illustrasjoner som kanskje har vært spesielt gode.	Mulighet til å lagre illustrasjoner man har gjort.
Benytter lang tid på å illustrere en situasjon med penn og papir	Elever Trafikkskolen	Går av tiden til eleven, tid som kunne blitt brukt bedre.	Ferdigdefinerte «trafikksituasjoner» å velge i, og mulighet til å tegne over disse. Viktig at dette går like fort som å gjøre det samme med penn og papir,

		Skaper rom for misforståelser mellom lærer og elev. Tap av omdømme for trafikkskolen, hvis de får mange misfornøyde elever.	men at det blir mye mer *detaljert*.
Manglende system for illustrering av trafikksituasjoner	Trafikkskolen	Tap av tid og penger.	Et system for illustrering av trafikksituasjoner

TABELL 1 - FUNKSJONELLE EGENSKAPER TIL PRODUKTET

IKKE-FUNKSJONELLE EGENSKAPER OG KRAV

Produktet bør i tillegg ha følgende ikke-funksjonelle produktegenskaper og krav:

Egenskap/krav	Berører	Beskrivelse	Vellykket løsning
Brukervennlighet	Brukere	Stort aldersspenn blant trafikklærere gjør at applikasjonen må være brukbar både for yngre og eldre brukere. Dårlig brukervennlighet kan gjøre at enkelte kvier seg for å ta i bruk appen, da det krever for mye å sette seg inn i.	Applikasjonen bør følge etablerte standarder for Android-applikasjoner og universell utforming. Benytte komponenter som er kjent fra andre applikasjoner. Fokus på et enkelt og oversiktlig oppsett.
Responstid	Brukere	Hvis applikasjonen bruker for lang tid på å laste inn, ofte «fryser» eller lignende, vil det bli mer kost enn nytte for brukerne å ta i bruk applikasjon i forhold til å holde seg til dagens løsning med penn og papir.	Fokus på å begrense størrelsen på applikasjonen: Ta i bruk SVG-filer i stedet for JPG/PNG. Holde antall tredjepartbiblioteker til et minimum, hvis et bibliotek benytter mye lagringsplass – se på andre alternativer i stedet.
Brukbarhet	Brukere Elever	Å benytte applikasjonen til å illustrere situasjoner må ikke ta lengre tid enn det gjør med penn og papir. Det må gå like raskt å gjøre det i applikasjonen, men det må være mer *presist*.	Ferdigdefinerte situasjoner med korrekt og detaljert veioppmerking og skilting, slik at alt man trenger å gjøre er å velge riktig «situasjon», for eksempel veikryss, og deretter

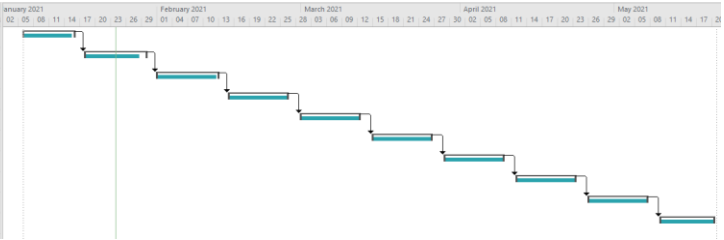
			tegne inn bilen og annet man trenger for å illustrere. Appen må være oversiktlig, og det må ikke være for mange «ledd» man må gå gjennom får å finne fram til riktig situasjonsbilde.

TABELL 2 - IKKE-FUNKSJONELLE EGENSKAPER OG KRAV TIL PRODUKTET

3.6. PROSJEKTETS MILEPÆLER OG HOVEDAKTIVITETER

Prosjektet er delt opp i ti sprinter à 2 uker. Innholdet i hver sprint er avhengig av den foregående sprinten, og hva man klarte å få unnagjort der. Det er enkelte hovedmomenter som skal med i de forskjellige sprintene, og disse vil vi vise til her. På grunn av antall sprinter og antall oppgaver innenfor hver sprint vil kun oversikten over de planlagte sprintene komme med her. Vi viser til vedlagt prosjektplan – *Vedlegg 1: Initiell prosjektplan* - for full oversikt over de planlagte momentene.

	Task Name	Duration	Start	Finish	Predecessors
1	Sprint 1	8,2 days	Wed 06.01.21	Fri 15.01.21	
8	Sprint 2	10 days	Mon 18.01.21	Fri 29.01.21	1
17	Sprint 3	10 days	Mon 01.02.21	Fri 12.02.21	8
24	Sprint 4	10 days	Mon 15.02.21	Fri 26.02.21	17
31	Sprint 5	10 days	Mon 01.03.21	Fri 12.03.21	24
38	Sprint 6	10 days	Mon 15.03.21	Fri 26.03.21	31
45	Sprint 7	10 days	Mon 29.03.21	Fri 09.04.21	38
52	Sprint 8	10 days	Mon 12.04.21	Fri 23.04.21	45
59	Sprint 9	10 days	Mon 26.04.21	Fri 07.05.21	52
66	Sprint 10	9 days	Mon 10.05.21	Thu 20.05.21	59



FIGUR 1 - PROSJEKTPLAN

4. INTERESSENTER OG RAMMEBETINGELSER

Her vil vi først presentere hvem som er interessentene for prosjektet og hvilke suksesskriterier disse har. Deretter beskriver vi hvilke absolutte krav som stilles til prosjektets gjennomføring og til resultatet, som vil ha avgjørende innflytelse på planer og valg i prosjektet.

4.1. INTERESSENTANALYSE

En presentasjon av hvem som er interessentene for prosjektet, hvilke suksesskriterier de har og hva de skal bidra med i prosjektet.

Selv om dette prosjektet er basert på en oppgave vi i prosjektgruppen har definert selv, er det Sopra Steria som har rollen som oppdragsgiver eller oppgavestiller. Oppdragsgiver skal godkjenne resultatet av prosjektet i tillegg til å bidra med veiledning gjennom oppgaven.

Suksesskriterier for oppdragsgiver er at prosjektgruppen viser god arbeidsinnsats, jobber selvstendig med oppgaven og holder tidsfrister. I tillegg vil et suksesskriterium være at vi gjennom prosjektet viser at vi har en god kompetanse innen prosjektarbeid og utvikling.

Brukerne av applikasjonen er trafikklærere i Norge. På grunn av ulikheter i både veiutforming, veimerking, trafikkregler og trafikkopplæring mellom ulike land, vil det bli et for omfattende prosjekt å rette seg inn mot flere land. Det vil i all hovedsak være trafikklærere som berøres i det daglige, men de som er elever i kjøreopplæringen vil være indirekte berørt. Sistnevnte vil derimot ikke ha noen rolle hverken i prosjektet eller i bruk av applikasjonen.

Suksesskriterier for brukerne av applikasjonen, trafikklærere, vil være en brukervennlig og stabil applikasjon som de kan benytte til å illustrere trafikksituasjoner for elevene sine. Illustrering av trafikksituasjoner i appen må gå like raskt som å benytte penn og papir, men det må være mer *presist* og detaljert. Produktet må være et mer effektivt, enklere og bedre verktøy enn det de fleste benytter i dag – penn og papir.

Veileder fra NTNU er også en interessent i prosjektet. Veileder skal godkjenne resultatet av prosjektet sammen med oppdragsgiver Sopra Steria. Veileder fra NTNU skal også bidra med veiledning gjennom hele oppgaven, sammen med veileder fra Sopra Steria.

Suksesskriterier for veileder er at vi benytter 500 arbeidstimer (+/- 5 %) hver på prosjektet, at vi viser god arbeidsinnsats, jobber selvstendig med oppgaven og holder tidsfrister. I tillegg må vi gjennom prosjektet utarbeide all nødvendig dokumentasjon. Dette innebærer forstudierapport, kravdokument/kravspesifikasjon, driftsdokumentasjon, sluttrapport, møteinnkalling og møtereferater for alle veiledningsmøter, samt ukes- og statusrapporter. Det vil også være viktig at vi gjennom prosjektet benytter en stor del av den kunnskapen og kompetansen vi har tilegnet oss gjennom studiet.

Det er vi i prosjektgruppa, eller utviklingsteamet, som utfører alt av arbeid i dette prosjektet.

Suksesskriterier for prosjektgruppa er å ha utvikle en applikasjon som fungerer godt til det formålet den skal ha. Målet er at applikasjonen faktisk blir benyttet av trafikklærere til det som er tiltenkt. Vi må utvikle applikasjonen på en slik måte at den er skalerbar og brukervennlig, og få gjort applikasjonen tilgjengelig i Google Play Store (eller gjøre slik at den kan gis ut der med litt ekstra innsats). Det at vi klarer å få til et godt samarbeid gjennom hele prosjektet, vil også være et suksesskriterium.

Både oppdragsgiver, intern veileder og brukerne av produktet, vil ha behov for å vite noe om prosjektet. Det er derimot kun oppdragsgiver og intern veileder som vil ha behov for å vite noe om selve prosjektforløpet. Tabellen under viser en kort oppsummering av interessentanalysen.

Interessent	Suksesskriterier	Bidrag til prosjektet
	Eksterne	
Oppdragsgiver	Prosjektgruppen viser god arbeidsinnsats, jobber selvstendig med oppgaven og holder tidsfrister. At vi gjennom prosjektet viser at vi har en god kompetanse innen prosjektarbeid og utvikling.	Vil hovedsakelig bidra med veiledning. Er med på å ta beslutninger i prosjektet og vil være med på å godkjenne resultatene som oppnås gjennom prosjektet.
Sluttbruker	Brukervennlig og stabil applikasjon som de kan benytte til å illustrere trafikksituasjoner for elevene sine. Illustrering av trafikksituasjoner i appen må gå like raskt som å benytte penn og papir, men det må være mer *presist* og detaljert Et mer effektivt, enklere og bedre verktøy enn det de fleste benytter i dag – penn og papir.	God kjennskap til dagens situasjon og hvilke behov produktet må innfri.
	Interne	
Prosjektgruppen	Utvikle en applikasjon som fungerer godt til det formålet den skal ha. At applikasjonen faktisk blir benyttet av trafikklærere til det som er tiltenkt. Utvikle applikasjonen på en slik måte at den er skalerbar og brukervennlig. Utvikle applikasjonen på en slik måte at den kan tilgjengeliggjøres i Google Play Store. At vi klarer å få til et godt samarbeid gjennom hele prosjektet.	Ansvar for selve gjennomføringen av prosjektet innenfor tidsrammen.
Veileder	Prosjektgruppen benytter ca. 500 arbeidstimer hver på prosjektet. Prosjektgruppen utarbeider den nødvendige prosjektdokumentasjonen på en god måte: • Forstudierapport • Kravdokument/kravspesifikasjon	Veiledning til prosjektgruppen underveis i prosjektet, samt være med på å ta beslutninger og godkjenne resultatene som oppnås gjennom prosjektet.

	• Driftsdokumentasjon • Sluttrapport • Møteagenda og møtereferater for alle veiledningsmøter • Statusrapporter etter hver sprint Prosjektgruppen benytter en stor del av den kunnskapen og kompetansen de har tilegnet seg gjennom studiet. Prosjektgruppen viser god arbeidsinnsats, jobber selvstendig med oppgaven og holder tidsfrister.	

TABELL 3 - INTERESSENTENES SUKSESSKRITERIER OG BIDRAG TIL PROSJEKTET

4.2. RAMMEBETINGELSER

Rammebetingelsene setter absolutte krav for hvordan prosjektet skal gjennomføres og hvilke resultater som skal leveres. Meningen er å synliggjøre kravene tidlig i prosjektet ettersom disse vil være viktige i utformingen av andre rapporter.

- Det skal utvikles en Androidapplikasjon for trafikkopplæring ved bruk av JavaScript og React Native
- Applikasjonen skal erstatte penn og papir ved illustrering av trafikksituasjoner
- Git skal benyttes for kildekodeadministrasjon og versjonskontroll
- Prosjektgruppen skal utvikle applikasjonen ved hjelp av Scrum-metodikk
- Hver sprint skal være på henholdsvis 2 uker
- Prosjektet skal være ferdig og levert 20. mai, ved et senere tidspunkt skal det også gjennomføres en presentasjon
- Hvert prosjektmedlem skal jobbe 500 timer +- 5%
- Prosjektgruppen skal utarbeide forskjellige dokumenter i løpet av prosjektet (forstudierapport, kravdokument, driftsdokumentasjon og sluttrapport)
- Prosjektgruppen skal også dokumentere arbeidsprosessen ved å benytte møtereferater og sprintrapporter

5. KRITISKE SUKSESSFAKTORER

Her beskriver vi faktorer som vil være kritisk avgjørende for om prosjektets resultat vil bli vellykket eller ikke. Vi beskriver også hvilke informasjonsbehov de ulike interessentene har.

5.1. SUKSESSFAKTORER

Kritiske faktorer for suksessfull gjennomføringen av prosjektet:

- Det må opprettholdes god kommunikasjon mellom teamet og oppdragsgiver, samt mellom medlemmene i teamet.
- Teamet må benytte seg av de valgte verktøyene for å sikre fremdrift og for å kunne dokumentere arbeidet.
- Alle i teamet må ta del i alle deler av prosjektet.
- Grundig testing av applikasjonen gjennom hele utviklingen.
- Utvikle en applikasjon som er oversiktlig, lett å bruke og gjør trafikkundervisningen mer effektiv.
- Benytte Scrum som utviklingsmetodikk.
- Lettere arbeidsdag for trafikklærere.

5.2. INFORMASJONSBEHOV

For å kunne formidle informasjon til alle interessentene i prosjektet på en effektiv måte, har vi analysert hver enkelt interessent og hvilke informasjonsbehov de har. Ved å spesifisere informasjonsbehovet vil man kunne unngå å sende ut unødvendig informasjon til interessentene.

Interessent	Informasjonsbehov	Tidspunkt for informasjon	Formål
		Eksterne	
Oppdragsgiver	Prosjektets gang og hvordan prosjektet ligger an i henhold til milepæler, ressurser og tidsrammer.	Faste møter hver 2. uke fram til 17. februar, deretter hver 3. uke. fram til 12. mai.	Oppdragsgivers formål med informasjonen er å vite hvordan prosjektet ligger an i forhold til satte milepæler, ressurser og tidsrammer.
Sluttbruker (Trafikklærer)	Informasjon om eksisterende og nye funksjoner.	Etter hver sprint for å utføre test av produktet og de implementerte funksjonene.	Sluttbrukerne må være informert om de forskjellige implementerte funksjonene, og kunne gi tilbakemelding for videre utvikling.
		Interne	

Prosjektgruppen	Sprintrapporter hver 2. uke, aktive sprint-oppgaver og milepæler.	Kontinuerlig tilgang til backlog i Jira, sprint review hver 2. uke, 10-15 min «stand-up» tirsdag, onsdag og fredag.	Prosjektgruppen må alltid være informert om hvilket arbeid som gjenstår og om det går i henhold til milepælene som er satt.
Veileder	Rapporter, oversikt over gjennomførte og gjenstående oppgaver.	Faste møter hver 2. uke fram til 17. februar, deretter hver 3. uke. fram til 12. mai.	Veileder må holde styr på prosjektgruppen, være et bindeledd mellom gruppen og oppdragsgiver og andre eksterne grupper/personer. Ansvar for å holde gruppen innenfor satte milepæler.

TABELL 4 - INFORMASJONSBEHOV I PROSJEKTET

6. RISIKOANALYSE

Risikoene knyttet til prosjektet og utføringen av det, har for det meste sammenheng i hvor godt vi har planlagt prosjektet vårt og hvorvidt vi klarer å få utført alle nødvendige deler for å få en fullstendig MVP. Gruppen har lite erfaring med teknologien som skal brukes, React Native, som betyr at vi må bruke litt ekstra tid i begynnelsen på kursing, opplæring og testing.

Et annet moment er distansearbeid. Oppgaven blir utført ved å kjøre hjemmekontor hvor alle deltakere kommuniserer over plattformer som Teams og Slack. Dette i seg selv har en risikofaktor som må vurderes og reflekteres over. Det er viktig at alle medlemmene i gruppen er innforstått med hva dette innebærer og hvordan håndtere eventuelle interne konflikter ved bruk av videosamtaler istedenfor å kunne møtes fysisk. Risikoanalysen vil ta for seg dette elementet samt andre som har med interne konflikter å gjøre.

Gruppen har intensjoner og håp om å kunne publisere MVP-versjonen eller en Alpha-versjon via Google Play mot slutten av prosjekttiden. Dette innebærer at vi må arbeide effektivt mot de spesifikke målene våre, ha god kommunikasjon og stå til bistand om et gruppemedlem har problemer med deres arbeidsoppgaver. Publisering kan også ta tid, da det må godkjennes og noen retningslinjer må legges til grunn. Derfor er det viktig at alle på gruppen er innforstått med målene som blir satt til hver sprint og at disse arbeides mot. For å unngå at MVP-versjonen blir for tynn, må vi ha satt oss oppgaver hver uke som vil medføre et positivt produkt som gruppen ser seg fornøyd med.

Andre risikofaktorer er interne konflikter, konflikter mellom veiledere og teamet samt fravær i gruppen. Gruppen består bare av 3 medlemmer som arbeider godt sammen (fra tidligere erfaring), men om noen av disse skulle oppstå må vi ha klare retningslinjer for å håndtere disse tilfellene. Det vil være viktig med hyppig kommunikasjon, oppdateringer mellom medlemmene og møter med veiledere for å sikre at vi holder oss tritt med satte forventinger. Om konflikter skjer internt vil universitetets veileder kunne bistå som mellomledd for å avvikle eventuelle konflikter. Gruppen vil også iverksette tiltak for å forbedre arbeidsmiljøet, spesielt siden det skjer over distanse, slik at vi har en god arbeidskultur hvor alle medlemmer føler de kan ytre sin mening. Det vil være viktig at vi kan snakke sammen om urettferdigheter og problemer knyttet til utføringen av dette prosjektet.

Forsinkelser i prosjektet kan også oppstå av ulike grunner, det meste avhenger av overnevnte risikoer som har med kompetanse og distansearbeid å gjøre. Vi vil fokusere på å bruke god nok tid på planlegging i starten av prosjektet for å kunne unngå de fleste risikoelementene, samt at alle i gruppen har en forståelse av disse elementene. Det viktigste for gruppen er å få et produkt vi er fornøyde med og med de kravspesifikasjonene som blir satt i kravdokumentet senere. God kommunikasjon gjennom hele prosjektet og samarbeidsvilje er noe som blir satt høyt på listen. Alle medlemmer vil også kunne bistå med å hjelpe hverandre om noen ser seg låst på oppgaven sin, så gruppen kommer til å arbeide for å ha en god arbeidskultur som støtter alles meninger.

Tabellen under viser en oversikt over alle risikomentene i prosjektet. Sannsynligheten og konsekvensen er et tall mellom 1 og 5, der 5 er veldig høy sannsynlighet/veldig store konsekvenser.

H	Moment	Sannsynlighet	Konsekvens	Risiko	Tiltak
1	MVP ikke ferdig eller ufullstendig i forhold til planen	3	5	15	Kommunikasjon i teamet, spørre etter hjelp om nødvendig samt forhøre oss med veiledere når vi står låst.
2	Manglende kompetanse i prosjektgruppen	4	4	16	Bruke ekstra tid på opplæring internt av bla. React Native.
3	Distansearbeid fører til at gruppen jobber mindre effektivt	3	2	6	Faste tidspunkter hvor vi har «arbeidstid», innenfor disse tidspunktene skal alle være tilgjengelig. Samt korte møter på starten av dagen som «stand-up» for å holde alle oppdatert.
4	Uenigheter og misforståelser mellom veiledere og gruppemedlemmer	2	3	6	Møter annenhver til hver tredje uke med statusgjennomgang.
5	Konflikt internt i prosjektgruppen.	2	2	4	Ha en åpen kultur hvor alles meninger har verdi og bruke eventuelle tredjepartsressurser for å avvikle konflikter.
6	Fraværende prosjektmedlemmer	1	3	3	Arbeide for en prosjektkultur der man tar vare på hverandre og kan snakke åpent sammen.
7	Mål blir satt til side for andre mindre viktige oppgaver	2	4	8	Møter hvor vi diskuterer arbeid gjort de siste dagene og hvorvidt dette er riktig fokus akkurat nå.

TABELL 5 – RISIKOANALYSE

Tabellen under viser en visualisert oversikt over risikomentene og deres alvorlighetsgrad.

Konsekvens -> Sannsynlighet	1 Ufarlig	2 Viss fare	3 Farlig	4 Kritisk	5 Svært kritisk
5 - Svært sannsynlig					
4 – Meget sannsynlig				Moment 2 Svært alvorlig	
3 – Sannsynlig		Moment 3 Alvorlig			Moment 1 Svært alvorlig
2 – Mindre sannsynlig		Moment 5 Lite alvorlig	Moment 4 Mindre alvorlig	Hendelse 7 Alvorlig	
1 – Lite sannsynlig			Hendelse 6 Lite alvorlig		

TABELL 6 - TABELL OVER ALVORLIGHETSGRAD PÅ HENDELSER

7. KOST/NYTTE-ANALYSE

En kost/nytte-analyse benyttes for å avgjøre om prosjektet skal gjennomføres eller ikke. Hvis kostnadene ved å gjennomføre prosjektet overstiger nytten ved prosjektet, bør det ikke gjennomføres.

Måten vi gjennomfører denne analysen på, er å først beskrive nytten ved prosjektet. Dette inkluderer både den nytten som kan kvantifiseres og den som ikke kan det. Deretter beskriver vi kostnadene ved prosjektet, før vi sammenstiller kostnadene og nytten.

7.1. KVANTIFISERBAR OG IKKE-KVANTIFISERBAR NYTTE

Kvantifiserbar nytte:

- Flere kunder på grunn av undervisning utført på en effektiv og kvalitetsbevisst måte

I snitt koster det 30 000 NOK å kjøre opp til førerkort i klasse B [3]. Inkludert i disse beregningene er også priser for teoriprøve (680,-), førerprøve (1140,-), utstedelse av førerkort (150,-) og bilde (80,-) [4]. I tillegg kommer lærebok (390,-). Ved å fjerne prisen for disse vil vi kun sitte igjen med gjennomsnittssummen en elev betaler for kjøreopplæringen hos en trafikkskole. Dermed blir den samlede kvantifiserbare nytten:

Antall elever	Pris per elev	Sum	% overskudd	Sum overskudd
10	27 560 NOK	275 600 NOK	25 %	68 900 NOK

TABELL 7 - ESTIMERT KVANTIFISERBAR NYTTE

Ikke-kvantifiserbar nytte:

- Verdien av mindre frustrasjon hos trafikklærerne
- Verdien av mindre frustrasjon hos elevene
- Verdien av en bedre forståelse av trafikkregler hos elevene
- Verdien av bedre omdømme hos kundene og fremtidige ansatte

7.2. ESTIMERTE KOSTNADER

Siden dette er et bachelorprosjekt, får vi ikke lønn i kroner for det arbeidet vi gjør. Kostnadene våre forbundet med prosjektet er derimot bruk av tid. For å tallfeste dette har vi satt en fiktiv timepris for hver av oss, der det er estimert at vi benytter 500 arbeidstimer hver.

Under er en tabell med oversikt over kostnader relatert til prosjektet. Tabellen viser en oversikt over kostnader per time, totalsummen per ressurs og totalsummen for hele prosjektet. Prosjektet er estimert til å være 67 arbeidsdager.

Ressurs	Timepris	Totalsum prosjekt
Silje Tanemsmo	600 NOK	300 000 NOK

Oliver E. Nilssen	600 NOK	300 000 NOK
Joakim H. Tronseth	600 NOK	300 000 NOK
Totalsum		900 000 NOK

TABELL 8 - ESTIMERTE KOSTNADER FOR PROSJEKTET

Vi ser også for oss at det vil kreve videre utvikling og forbedringer av applikasjonen, og tenker oss en årlig kostnad på 50 000 NOK for denne oppgaven.

7.3. SAMMENSTILLING KOST/NYTTE

I tabellen under er det gjort et estimat av kost og nytte for **én** trafikkskole over en periode på 5 år. Utviklingskostnadene til selve prosjektet er i år 1. Kostnadene til videreutvikling vil fordeles utover de gjeldende årene utviklingen utføres. Vi ser for oss en gradvis økning i antallet elever per år med 50 % (med basis i utregningen vår fra kvantifiserbar nytte) som følger av applikasjonen. Vi runder ned i tilfeller hvor desimaler inntreffer.

	År 1	År 2	År 3	År 4	År 5	SUM
Antall elever		10	15	22	33	80
Kvantifiserbar nytte	0	68 900	103 350	151 580	227 370	551 200
Utviklingskostnader	900 000					900 000
Videreutvikling		50 000	50 000	50 000	50 000	200 000
Sum kostnader	900 000	50 000	50 000	50 000	50 000	1 100 000
Beregnet nytte	-900 000	18 900	53 350	101 580	177 370	-548 800

TABELL 9 - ESTIMERT KOST/NYTTE

Det er viktig å presisere at denne utregningen er kun gjort med tanke på én bedrift. Utvider man denne sammenstillingen til å innefatte flere vil utviklingskostnadene være like, men den kvantifiserbare nytten vil øke.

IKKE-KVANTIFISERBAR NYTTE

Vi vil også beskrive den ikke-kvantifiserbare nytten ved prosjektet, disse må tas med ettersom de kan være avgjørende for at et prosjekt skal kunne bli gjennomført. I dette prosjektet er den kvantifiserbare nytten mindre enn utviklingskostnadene, så det faller på den ikke-kvantifiserbare nytten om prosjektet blir godkjent eller ikke.

- Mindre frustrasjon blant ansatte, økt trivsel og bedre arbeidsmiljø
- Mindre frustrasjon blant elevene, økt trivsel og mindre frafall
- Økt kvalitet på undervisning, bedre trafikkforståelse blant elevene
- Bedre omdømme fører til at bedriften blir mer ettertraktet, både hos kunder og fremtidige ansatte

8. TEKNOLOGIER

I dette kapittelet beskriver vi ulike teknologier vi har valgt å bruke i utføringen av prosjektet. Disse teknologiene er viktige da dette er et utviklingsprosjekt hvor teknologivalg har en høy betydning for utføringen av prosjektet.

8.1. REACT NATIVE

React Native er et åpent kildekode-rammeverk utviklet av Facebook. Rammeverket muliggjør utvikling av native applikasjoner for iOS og Android ved hjelp av JavaScript og de samme mekanismene som blir brukt i React. Mye av koden som blir skrevet i React Native kan bli gjenbrukt på tvers av plattformene, som kan effektivisere utviklingen og gjøre at vedlikehold av applikasjonene blir lettere.

Rammeverket er delt som åpen kildekode-biblioteker gjennom NPM (Node Package Manager). NPM er en pakkehåndterer for JavaScript og et datterselskap av Git [5], som tilbyr hosting for programvareutvikling og versjonskontroll sammen med Git. NPM er en kommandolinjeklient, som vil si at funksjonene brukes via terminalen på datamaskinen for å installere, oppdatere og kjøre de forskjellige pakkene som blir brukt under utviklingen.

Gruppen har valgt å bruke React Native akkurat fordi det er et fint rammeverk å bruke for utvikling av applikasjoner som skal utvikles på tvers av ulike plattformer. I første omgang kommer gruppen til å ha fokus på Android, men ved å bruke dette rammeverket kan vi lett adoptere det til iOS-enheter også.

Vi har også valgt å bruke React Native Cli istedenfor den mer brukte Expo, som er metoder for å starte et React Native prosjekt på. Vi valgte Cli over Expo da vi planlegger å bruke SVG-filer istedenfor en database som blir lagret lokalt på enheten til brukeren, denne funksjonen er ikke tilgjengelig via Expo.

8.2. SLACK

Slack er et kommunikasjonsverktøy for bruk under utvikling. Vi bruker det for å kunne ha forskjellige samtaler angående de forskjellige temaene som blir tatt opp under prosjektet. Vi har også Jira og GitHub koblet opp mot denne plattformen slik at vi får oppdateringer på Slack ved endringer i en av de andre plattformene.

Gruppen har delt Slack inn i flere kanaler; Design, kode, dokumentasjon og generelt. Disse kanalene skal bli brukt respektivt av hva de handler om, slik at vi lettere kan finne frem til tidligere diskusjonsemner og få hjelp til spesifikke problemer.

8.3. MICROSOFT TEAMS

MS Teams er et fantastisk verktøy for gruppearbeid og kommer i utgangspunktet til å bli brukt av prosjektgruppen for å holde oversikt over dokumentasjon, videosamtaler m.m. Teams blir brukt for å holde møter, sende møteinnkallinger og for å lagre dokumenter som kommer til å bli brukt under utviklingen av applikasjonen vår.

8.4. JIRA

Jira er et verktøy utviklet av Atlassian, som brukes til å holde oversikt over oppgaver som må gjøres, hva som gjenstår og for å registrere hvor mye tid som er brukt av hvert medlem på de forskjellige oppgavene. Jira er et fantastisk verktøy å bruke under Scrum-utvikling, da man kan dele oppgavene inn i forskjellige sprinter. Teamet vårt har valgt å dele hver sprint inn i to-ukers intervaller, hvor vi annenhver uke vil gjennomgå utførte og gjenstående oppgaver, og opprette en ny sprint for de to neste ukene.

I Jira kan man opprette en «backlog», som inneholder fremtidige oppgaver som skal utføres og legge til disse i sine respektive sprinter når det ansees som nødvendig. Vi vil også her ha en fin oversikt over hvem som jobber med hvilke oppgaver, samt at vi får et oppgavenummer som skal brukes når vi oppdaterer «repository» i GitHub slik at alle vet hvilke oppgaver som er blitt fullført.

8.5. GITHUB

Git er et system for versjonskontroll som holder styr på modifikasjoner som blir gjort i programkode. Det er et verktøy for å kunne ha oversikt over endringer gjort av andre gruppemedlemmer, alt fra små til store endringer [6]. Git gjør dette på en oversiktlig og systematisk måte. GitHub er hvor disse endringene blir lagret, samt koden. Dette blir lagret i en såkalt «repository» og ulike grener («branches»), slik at to utviklere kan arbeide på samme kode uten at det påvirker det den andre arbeider med. Vår gruppe kommer til å ta i bruk GitHub ved å ha en hoved-gren (branch) som all ferdig kode blir lagt til, men før vi legger koden vår inn her så arbeider vi i separate grener som blir navngitt etter Jira-oppgaven de utarbeides fra. Før et gruppemedlem kan legge koden sin inn på hoved-grenen, så må den bli godkjent av minst et annet medlem, dette er for å sikre bedre kodekvalitet og mindre sammenslåingsfeil.

9. RETNINGSLINJER OG STANDARDER

I dette kapittelet skal vi kortfattet ta med de retningslinjene og standardene som prosjektet må forholde seg til.

9.1. KRAV TIL DOKUMENTASJON

Her beskrives kort hvilke dokumenter som skal produseres i løpet av prosjektet, når de skal foreligge, på hvilken form de skal foreligge og evt. rutiner for godkjenning og revisjon av dokumentene.

Dokument	Beskrivelse	Når de skal foreligge	Rutiner for godkjenning
Forstudierapport	Beskriver hva prosjektet dreier seg om, hvilke krav som stilles til gjennomføringen, prosjektmål- og planer.	I løpet av den andre sprinten i prosjektet, som er i uke 3-4.	Rapporten skal sendes til oppdragsgiver og intern veileder, som enten godkjenner rapporten eller kommer med endringsønsker.
Kravdokument	Inneholder kravspesifikasjoner til produktet, og beskrivelse av brukergrensesnittet.	Arbeidet med dokumentet vil foregå iterativt gjennom store deler av prosjektet, med endelig versjon ved prosjektslutt 20. mai. Gjeldende utkast gjøres tilgjengelig for interessentene etter hver sprint.	Gjeldende versjon gjøres tilgjengelig for oppdragsgiver og intern veileder etter hver sprint. De kan deretter komme med tilbakemeldinger.
Driftsdokumentasjon	Beskriver det tekniske ved prosjektet. Er laget på en slik måte at det er mulig for andre å sette seg inn i og videreutvikle applikasjonen ut ifra dette dokumentet.	Ved prosjektslutt 20. mai.	Ingen godkjenning.
Møteinnkallinger	Innkalling og agenda for alle veiledningsmøter og brukermøter i prosjektet.	Sendes på e-post til møtedeltakere senest kvelden før møtet.	Godkjenning/forslag til endring gjøres på starten av møtet innkallingen gjelder for.
Møtereferat	Referat fra alle veiledningsmøter og	Sendes på e-post til møtedeltakere etter	Godkjenning/endringsønsker gjøres senest på neste møte,

	brukermøter i prosjektet.	avholdt møte samme dag.	men kan også gjøres fortløpende av alle møtedeltakere etter mottatt referat.
Sprintrapport	Inneholder Sprint Review, Sprint retrospektiv, tabell over timer jobbet på sprinten og timer jobbet hittil i prosjektet.	Etter fullført sprint, før ny sprint startes.	Ingen godkjenning.
Sluttrapport	Gjennomgang av prosjektet, om det har gått slik vi planla.	Ved prosjektslutt 20. mai.	Rapporten gjøres tilgjengelig for oppdragsgiver og intern veileder ved prosjektslutt.

TABELL 10 - KRAV TIL DOKUMENTASJON I PROSJEKTET

9.2. KRAV TIL KVALITETSGJENNOMGANGER

De resultatene som skal ha kvalitetsgjennomgang er alt av dokumentasjon og rapporter, design og til slutt selve koden. Gruppen har valgt å kvalitetssikre spesielt alle dokumenter før de blir sendt inn til veilederne. Dette gjøres ved at alle i gruppen først leser gjennom rapportene og dokumentene på egen hånd og kommenterer der det føles nødvendig. Deretter har vi en gjennomgang sammen hvor vi kan diskutere/drøfte innholdet. Dette er for å sikre at kvaliteten på produktet vi leverer inn holder den standarden som gruppen har satt seg. Gjennomgangen av dokumentene skjer i fellesskap, men lesingen av dokumentet må hver deltaker ta ansvar for å gjøre selv.

Når det gjelder gjennomgang av kode, så har vi en standard hvor minst en annen på gruppen ser gjennom koden som er skrevet og kommenter der det sees nødvendig, før noe av koden blir lagt sammen med «hoved»-koden. Dette er for å sikre at koden som blir slått sammen står i samme stil, og ikke påvirker eksisterende kode negativt. Hvert teammedlem har selv ansvar for å skrive kode som følger riktig struktur og at den ikke inneholder feil som kan føre til dobbeltarbeid for andre. Gjennomlesning av kode og «pull-requests» er veldig viktig for at gruppen skal få en kodebase som kan gjenbrukes og leses av andre utviklere.

Veileder Stein Meisingseth er den som har ansvar for å godkjenne arbeidet utført av gruppen, sammen med veileder fra Sopra Steria, Georg Kippernes.

9.3. KRAV TIL STANDARDER OG METODER

Tabellen under viser konkrete navne- og programmeringsstandarder vi har satt, verktøy vi skal benytte i prosjektet, og metoder vi benytter under gjennomføringen.

Type	Gjelder	Beskrivelse
	Metoder	

Utviklingsmetode	Vi benytter Scrum som metode i gjennomføringen av prosjektet.	Vi kjører sprinter med varighet på 2 uker. Etter hver sprint og før oppstart av neste sprint skal vi gjennomføre Sprint Review og Sprint Retrospective. Dette er for å sikre at vi kontinuerlig retter opp i eventuelle problemer som skulle oppstå.
Verktøy		
Jira	Utarbeiding og oversikt over backlog, nåværende sprint og gjennomførte sprinter.	Vi benytter Jira som prosjektstyringsverktøy. Her legger vi inn alle oppgaver som skal utføres i en backlog, som blir oppdatert kontinuerlig gjennom hele prosjektet. Vi har også oversikt over nåværende sprint og tidligere sprinter.
Clockify	Individuell føring av timer.	Vi benytter Clockify som en add-on i Jira, som gjør at vi kan føre timer på oppgaver direkte fra Jira. Prosjektdeltakerne har selv ansvar for å føre timer på oppgaver. Ved slutten av hver sprint eksporterer alle sin egen timeliste fra Clockify til Excel.
Excel	Timeliste for hele teamet, fordelt på sprint og hovedoppgave.	Vi benytter en felles timeliste i Excel for å ha oversikt over antall timer hele teamet har benyttet. Oversikten viser timer brukt per sprint per aktivitet, i tillegg til timer brukt per aktivitet hittil i prosjektet.
Microsoft Teams	Samskriving på dokumenter, dokumentdeling og digitale møter.	Både prosjektgruppen, oppdragsgiver og veileder har tilgang til teamet i Teams. Her legges alle dokumenter som er utarbeidet i prosjektet. Alle digitale møter blir også avholdt i Teams.
Slack	Intern kommunikasjon mellom medlemmene i prosjektgruppen.	Vi benytter Slack for intern kommunikasjon innad i prosjektgruppen, både formell og uformell kommunikasjon.
GitHub	Versjonskontroll og samskriving på kode.	Vi har opprettet en privat repository i GitHub. Denne repositorien er delt opp i to hoved-brancher: *main* og *development*. Vi oppretter nye brancher for hver oppgave vi skal utføre, og pusher deretter denne til *development* når vi er ferdige. Vi pusher endringer til *main* først når endringene har blitt godkjent av noen andre i teamet.
Navnestandarder		
Navn på oppgaver	Oppgaver (issues) som opprettes i Jira backlog	Nye oppgaver (issues) navnsettes med en «prefix» ut ifra hvilken type oppgave det er. Dette gjøres for å lette arbeidet med å hente ut hvilken type oppgave det er ført timer på. Vi benytter følgende prefixer: ▪ *D – Dokumentasjon (forstudierapport)* ▪ *A – Prosjektadministrasjon*

		▪ *DE – Design av applikasjonen* ▪ *R – Research* ▪ *KD – Kravdokument* ▪ *K – Koding (Alpha og MVP)* ▪ *M – Møtevirksomhet* ▪ *TB – Testing og forbedring* ▪ *BT – Brukertesting* ▪ *DR - Driftsdokumentasjon* ▪ *SD – Sluttrapport*
Navn på brancher	Vi benytter Git til versjonskontroll, og oppretter nye brancher for hver oppgave som utføres.	Nye brancher navnsettes slik at man ser hvilken oppgave i Jira den tilhører. Oppgaver (issues) i Jira starter alltid med *BADR* og et nummer. For oppgaven *BADR-1* skal navnet på branchen starte med *BADR-1*.
Programmeringsstandarder		
Programmeringsspråk	Vi benytter rammeverket React Native for utvikling av applikasjonen.	Applikasjonen blir utviklet med React Native og JavaScript. Mer spesifikt benytter vi *react-native-cli*, som blant annet støtter bruk av SVG-filer og tredjepartsmoduler via Node.js.
Navn på variabler, klasser og funksjoner	Gjelder for selve koden.	Variabler og funksjoner navnsettes på måten *camelCase*. Klasser og komponenter navnsettes på måten *CamelCase*, dvs. med stor bokstav først.

TABELL 11 - KRAV TIL STANDARDER OG METODER I PROSJEKTET

9.4. ENDRINGSHÅNDTERING

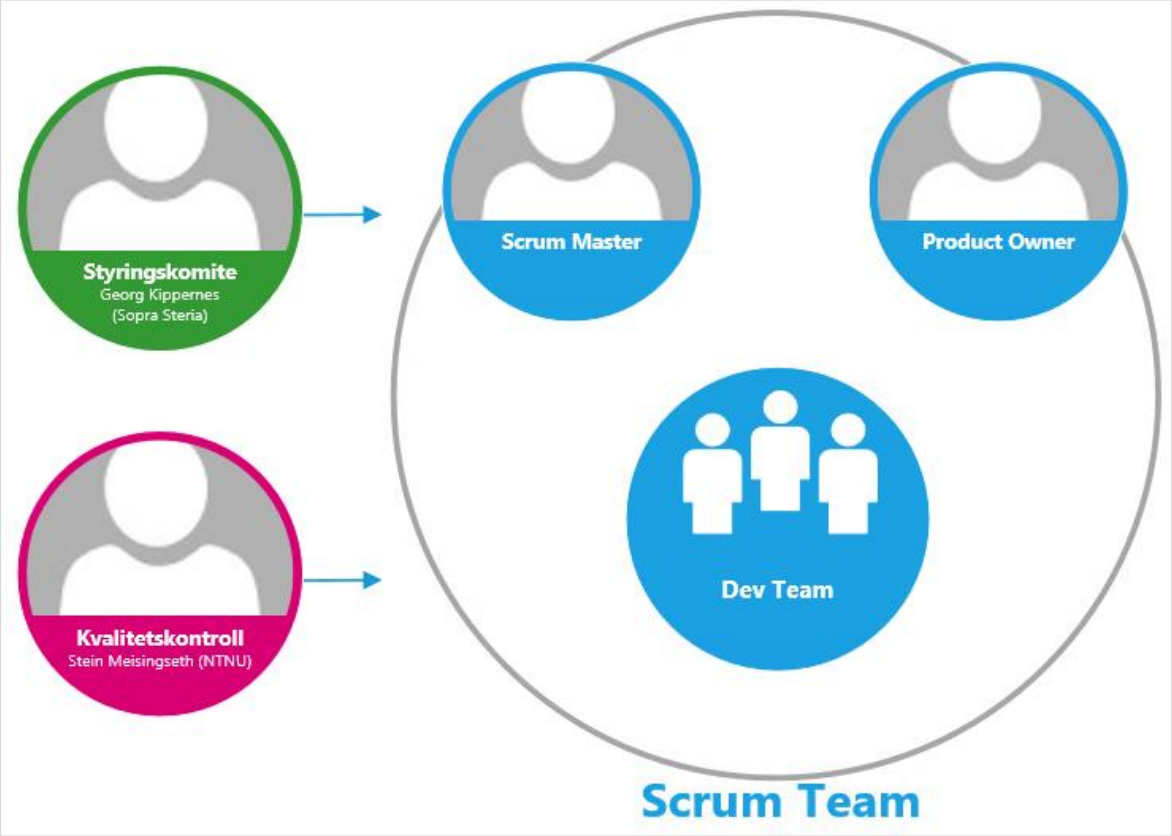
Ønsker om endringer i planen for prosjektet, som er presentert i denne forstudierapporten, kan komme fra oppdragsgiver, veileder, brukere og prosjektdeltakerne selv. Endringsønsker er vanlig og skal behandles formelt og forretningsmessig. Framgangsmåten for håndtering av endringsønsker er:

1. Dokumenter endringens innhold
2. Analyser konsekvensene for prosjektet
3. Beregne eventuell kost/nytte
4. Godkjennelse og aksept
5. Logg endringen
6. Juster planene
7. Informer interessentene
8. Gjennomfør endringen

10. PROSJEKTORGANISERING

I dette kapittelet skal vi vise hvem som er med i prosjektet og hvilke roller de har.

Siden dette er en bacheloroppgave og dermed et studentprosjekt, vil prosjektorganisasjonen avvike noe fra det som er vanlig i andre prosjekter. Siden vi benytter vår egen implementering av Scrum som utviklingsmetode, vil også organiseringen av Scrum Teamet avvike fra en standard Scrum Team-organisasjonsstruktur.



FIGUR 2 - PROSJEKTORGANISASJONEN

For å komme fram til prosjektorganisasjonen har vi kartlagt følgende:

- Styringskomite: Sopra Steria, representert av Georg Kippernes, har rollen som oppdragsgiver. Vedkommende fungerer som veileder og fyller rollen som styringskomite for prosjektet.
- Kvalitetskontroll: Rollen som kvalitetskontroll fylles av veileder v/ NTNU, Stein Meisingseth.
- Referansegruppe: Vi har vurdert at det ikke er behov for at en representant for brukerne fyller rollen som referansegruppe. Vi vil i stedet foreta brukertester som vil dekke behovet for å vurdere kvaliteten på resultatet, sett fra oppdragsgivers side.
- Produkteier: I Scrum har produkteier det siste ordet når det kommer til prioritering av oppgaver. Siden det er vi studenter som skal ha det hele og fulle ansvaret for dette prosjektet, kan ikke denne rollen fylles av hverken oppdragsgiver eller veileder v/ NTNU. Vi har dermed besluttet at rollen som produkteier innehas av hele teamet, med både veileder fra Sopra Steria og NTNU som bidragsytere.

- Scrum Master/prosjektleder: Rollen som prosjektleder eller Scrum Master vil gå på rullering mellom alle medlemmene i utviklingsteamet. Tabell 12 - Rullering av rollen som Scrum Master viser en oversikt over hvem som innehar rollen som Scrum Master når.

Sprint	Ukenummer	Scrum Master
1	1-2	Oliver
2	3-4	Joakim
3	5-6	Silje
4	7-8	Oliver
5	9-10	Joakim
6	11-12	Silje
7	13-14	Oliver
8	15-16	Joakim
9	17-18	Silje
10	19-20	Oliver

TABELL 12 - RULLERING AV ROLLEN SOM SCRUM MASTER

- Utviklingsteam: Vi har besluttet å ikke ha bestemte roller, men at vi fordeler arbeidet mellom oss etter hvert som oppgavene klargjøres.

11. ANBEFALING OM VIDERE ARBEID

Prosjektet anbefales videreført med de rammene og planene som er lagt frem her i dette dokumentet. Etter mye diskusjon har gruppen konkludert med å ta i bruk React Native for å forenkle plattformspesifikk programmering. Ved å bruke dette språket får vi tilgang til viktige pakker som vil forenkle arbeidet for prosjektgruppen i løpet av prosjekttiden. Gruppen anbefaler også å følge satte standarder som er beskrevet i dette dokumentet, dette vil føre til at videre arbeid vil være forutsigbart og uten unødvendig dobbeltarbeid.

Vi ser i kost-nytte kapittelet nytten av å videreføre arbeidet på dette prosjektet. Det vil føre til bedre samarbeid mellom elev og lærer, samt mindre tid brukt av lærere for å lage egne illustreringer, forklaringer og oppsett. I dag bruker trafikklærere tunge eller utdaterte metoder for å forenkle undervisningen for elevene sine, som er en kost trafikkskolene og lærerne selv påtar seg. Ved å videreføre prosjektet som nevnt over vil kapasiteten til lærerne økes, noe som vil gi dem mulighet til å sette fokuset sitt på andre mer viktige aspekter ved læringen.

Vi i prosjektgruppen ser fram til videre arbeid med Sopra Steria, veileder Georg Kippernes og studieveileder Stein Meisingseth.

12. REFERANSER

[1] React Native, "React Native - A framework for building native apps using React," [Online]. Available: https://reactnative.dev/. [Accessed 15 Januar 2021].

[2] J. Jones and S. Waddell, "The 5 Stages of Product Prototyping," 26 Februar 2019. [Online]. Available: https://uxplanet.org/the-5-stages-of-product-prototyping-ebb276004640. [Accessed 15 Januar 2020].

[3] Altomførerkortet.no, "Hva koster det å ta lappen," 2020. [Online]. Available: https://www.altomforerkortet.no/forerkort-priser. [Accessed 28 januar 2021].

[4] Statens Vegvesen, "Priser på teoriprøve, oppkjøring, utstedelse og foto," Statens vegvesen, 5 januar 2021. [Online]. Available: https://www.vegvesen.no/forerkort/ta-forerkort/priser. [Accessed 28 Januar 2021].

[5] S. Nguyen, "What is npm? A Node Package Manager Tutorial for Beginners," 16 Juli 2020. [Online]. Available: https://www.freecodecamp.org/news/what-is-npm-a-node-package-manager-tutorial-for-beginners/. [Accessed Januar 2021].

[6] K. Brown, "What is GitHub, and What is it Used For?," 31 November 2019. [Online]. Available: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/. [Accessed Januar 2021].

KRAVDOKUMENT

BACHELORPROSJEKT 087: UTVIKLING AV ANDROID-APPLIKASJON FOR TRAFIKKLÆRERE

Oliver Elias Nilssen, Joakim Heitmann Tronseth og Silje Tanemsmo

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
04.03.2021	0.1	Første utkast	Joakim H. Tronseth, Oliver E. Nilssen og Silje Tanemsmo
10.03.2021	0.2	Lagt inn "Myndighetspyramiden" og "innstillinger" i UC UML-diagrammet. Lagt til UC tekstlig beskrivelse av UC 8. Lagt inn Wireframes for "Trafikkskilt"-siden.	Joakim H. Tronseth Oliver E. Nilssen Silje Tanemsmo
24.03.2021	0.3	Lagt inn Wireframes for "Læreplanmål"-siden, og oppdatert listen med kravene for å reflektere fullførte oppgaver.	Silje Tanemsmo
11.04.2021	0.4	Lagt inn Wireframes for "Myndighetspyramiden"-siden og "Google Maps"-siden.	Silje Tanemsmo
15.04.2021	0.5	Lagt inn Wireframes for "Innstillinger"-siden.	Oliver E. Nilssen
05.05.2021	0.6	Oppdatert liten over kravene for å reflektere fullførte oppgaver. Oppdatert innholdsfortegnelsen.	Silje Tanemsmo
15.05.2021	1.0	Endelig versjon.	Joakim H. Tronseth, Oliver E. Nilssen og Silje Tanemsmo

INNHOLDSFORTEGNELSE

1. INNLEDNING

I dette dokumentet beskrives de funksjonelle kravene til applikasjonen – En Android-applikasjon for trafikklærere. Vi har beskrevet de overordnede kravene til produktet og prosjektet i forstudierapporten.

Hensikten med dokumentet er å gi en oversikt over logikken til applikasjonen, og sikre at de funksjonene vi implementerer stemmer overens med de kravene brukerne har til applikasjonen.

I kapittel 2 – Bakgrunn og oversikt – har vi en prioritert liste over kravene til applikasjonen. Kravene med høyest prioritet vil implementeres først, og her vil vi ha god dialog med brukerne for å sikre at vi har prioritert riktig. Ut ifra kravene til applikasjonen presenterer vi her et Use Case-diagram over applikasjonens bruksområder.

Kapittel 3 – Detaljerte krav – inneholder detaljerte tekstlige beskrivelser av hvert Use Case. Dette er et supplement til Use Case-diagrammet.

Kapittel 4 – Beskrivelse av brukergrensesnitt – inneholder bilder og beskrivelse av prototypen til applikasjonen. Dette er et forenklet bilde av hvordan applikasjonen skal være, med hovedfokus på hvordan applikasjonen skal oppføre seg og hvor brukeren skal finne de ulike funksjonene.

1.1. DEFINISJONER OG FORKORTELSER

Use Case (UC):	Use Case kan oversettes til *bruksmønster*, og er en beskrivelse av hvordan programvaren og brukeren skal samhandle. Det viktigste med et Use Case er de tekstlige beskrivelsene, som har til hensikt å oppdage og fange opp brukerens mål for systemet [7].
Use Case UML-diagram	Et Use Case UML-diagram er et diagram basert på UML (Unified Modeling Language). Det brukes for å visualisere hvordan systemet og systemets brukere skal samhandle, og hvilke handlinger brukeren skal kunne utføre i systemet [2].

2. BAKGRUNN OG OVERSIKT

I denne delen vil vi, ved hjelp av en prioritert liste over krav og et Use Case UML diagram, gi en oversikt over hvilke funksjoner applikasjonen skal ha og hvordan vi tenker å implementere dem.

2.1. PRIORITERT LISTE OVER KRAV

Vi har fordelt kravene i to ulike lister eller tabeller, ut ifra om de er *hovedfunksjoner i appen* eller *mindre viktige funksjoner*. For å enkelt vise hvilke funksjoner/krav vi har implementert, har vi satt en grønn kolonne til venstre i tabellen for de vi har implementert.

2.1.1. HOVEDFUNKSJONER

Oversikt over de viktigste kravene til appen. Dette er funksjoner for å velge mellom ulike trafikksituasjoner og tegne på, samt en menyoversikt over disse funksjonene og mulighet til å navigere mellom de ulike sidene i appen.

	KRAV	PRIORITET	FORESLÅTT LØSNING
+	En hovedmeny med tilgang til alle appens navigasjonssider	Høy	Startmeny med ikoner og tekst for hver funksjon. Samle naturlig tilhørende funksjoner sammen, slik at det blir mer ryddig på startmenyen. Følgende skal ligge på startskjermen: • Veikryss (tegnefunksjon) • Rundkjøring (tegnefunksjon) • Vei (tegnefunksjon) • Trafikkskilt • Læreplanmål • Myndighetspyramiden • Nyttige lenker
+	Enkelt gå tilbake til startskjermen fra andre sider	Høy	Ikon for "Hjem" øverst på verktøylinjen. Dette ikonet er tilgjengelig på alle sider, og ved å trykke på dette blir man sendt direkte til startmenyen.
		Veikryss	
+	Velge veikryss og tegne med "frihånd" over dette	Høy	Velger "Veikryss" fra startmenyen, og kommer direkte til en side der man kan begynne å tegne oppå dette krysset. Alt av statiske komponenter (riktig veioppmerking, skilting) må være på plass i bildet.
+	Under veikryss, kunne velge mellom kryss med	Høy	Velger "Veikryss" fra startmenyen. Kommer direkte til en side der man kan begynne å tegne.

	høyreregel, forkjørskryss og lyskryss.		En verktøylinje nederst på skjermen der man kan velge mellom krysstypene. Kryss med høyreregel er valgt som standard, da det er den mest brukte krysstypen. Alt av statiske komponenter (riktig veioppmerking, skilting) må være på plass i bildet ut ifra hvilken type kryss det er.
+	Under veikryss, kunne velge mellom X-, T- og Y-utforming på krysset.	Medium	Velger "Veikryss" fra startmenyen. Kommer direkte til en side der man kan begynne å tegne. En verktøylinje der man kan velge mellom krysstypene. Kryss med høyreregel er valgt som standard, da det er den mest brukte krysstypen. Etter krysstype er valgt, kan man deretter velge ønsket utforming på krysset med en annen knapp på verktøylinjen nederst på skjermen. Her er X-utforming valgt som standard, da det er den vanligste utformingen. Alt av statiske komponenter (riktig veioppmerking, skilting) må være på plass i bildet ut ifra hvilken type kryss det er.
+	For veikryss: Mulighet til å velge sykkelfelt, busslomme eller gangfelt.	Medium	I verktøylinjen nederst på skjermen er det en knapp (Drop Down-meny) med mulighet til å velge enten sykkelfelt, busslomme eller gangfelt. Dette blir da tegnet inn i krysset (ut ifra valgt krysstype og kryssutforming) med riktig veioppmerking og skilting.
	For veikryss: Mulighet til å velge flere kombinasjoner av sykkelfelt, busslomme og gangfelt (for eksempel busslomme og sykkelfelt).	Lav	I verktøymenyen nederst på skjermen er det knapper (eventuelt "radio-buttons") for busslomme, sykkelfelt og gangfelt. Velger man en av dem, blir dette da tegnet inn i krysset (ut ifra valgt krysstype og kryssutforming) med riktig veioppmerking og skilting. Velger man en til, blir dette også tegnet inn i tillegg til den som er valgt fra før.
		Rundkjøring	
+	Velge rundkjøring og tegne med "frihånd" over dette	Høy	Velger "Rundkjøring" fra startmenyen. Kommer direkte til en side der man kan begynne å tegne.
+	Under rundkjøring, kunne velge mellom ett- og tofelts rundkjøringer	Medium	Velger "Rundkjøring" fra startmenyen. Kommer direkte til en side der man kan begynne å tegne. En verktøylinje nederst på skjermen der man kan velge mellom ett- og tofelts rundkjøring. Ett-felts er valgt som standard.

			Alt av statiske komponenter (riktig veioppmerking, skilting) må være på plass i bildet ut ifra hvilken type rundkjøring det er.
+	For rundkjøring: Mulighet til å velge sykkelfelt, busslomme eller gangfelt.	Medium	I verktøylinjen nederst på skjermen er det en knapp (Drop Down-meny) med mulighet til å velge enten sykkelfelt, busslomme eller gangfelt. Dette blir da tegnet inn i rundkjøringen (ut ifra valgt rundkjøring) med riktig veioppmerking og skilting.
	For rundkjøring: Mulighet til å velge flere kombinasjoner av sykkelfelt, busslomme og gangfelt (for eksempel busslomme og sykkelfelt).	Lav	I verktøymenyen nederst på skjermen er det knapper (eventuelt "radio-buttons") for busslomme, sykkelfelt og gangfelt. Velger man en av dem, blir dette da tegnet inn i rundkjøringen med riktig veioppmerking og skilting. Velger man en til, blir dette også tegnet inn i tillegg til den som er valgt fra før.
		Andre situasjoner	
+	Velge andre type situasjoner enn veikryss og rundkjøring, og tegne med "frihånd" over dette	Lav	Velger "Vei" fra startmenyen. Kommer direkte til en side der man kan velge mellom andre typer situasjoner å tegne over. Dette er: • Avkjøringsfelt motorvei • Påkjøringsfelt motorvei • Landevei • Smal veg En verktøylinje nederst på skjermen der man kan velge mellom de ulike situasjonene. Alt av statiske komponenter (riktig veioppmerking, skilting) må være på plass i bildet ut ifra hvilken type situasjon det er.
		Felles for alle tegnefunksjoner (veikryss, rundkjøring og vei)	
+	For tegnefunksjoner: Mulighet til å tegne med frihånd	Høy	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *penn*. Man trykker først på knappen, og deretter tegner på skjermen.
+	For tegnefunksjoner: Mulighet til å viske ut det man har tegnet	Høy	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *viskelær*. Man trykker først på knappen, og deretter "visker" bort ønsket tegning fra skjermen ved å dra fingeren på skjermen.
+	For tegnefunksjoner: Mulighet til å angre det man har tegnet	Høy	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *angre*. Man trykker først på knappen, og deretter blir det som sist ble tegnet

			fjernet fra skjermen, dette inkluderer også om drabare elementer lagt til (ikke endring av posisjon).
+	For tegnefunksjoner: Mulighet til å fjerne alt som er tegnet med ett trykk	Høy	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *fjern alt*. Man trykker først på knappen, og deretter blir alt som er tegnet i denne sesjonen fjernet fra skjermen, dette inkluderer også drabare elementer som er lagt til.
+	For tegnefunksjoner: Mulighet til å velge farge på penneverktøyet	Medium	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *farge*. Man trykker først på knappen, og deretter velger man ønsket farge ut ifra en del forhåndsdefinerte farger for eksempel: • Svart (valgt som standard) • Grønn • Rød • Blå • Gul
+	For tegnefunksjoner: Mulighet til å velge størrelse på penneverktøyet	Lav	En verktøylinje øverst på skjermen for alle tegnefunksjoner. Her skal det ligge en knapp med ikon for *pennestørrelse*. Man trykker først på knappen, og deretter velger man ønsket størrelse med en "glider".
+	For tegnefunksjoner: Mulighet til å "dra inn" komponenter fra det dynamiske veimiljøet (biler, sykler, busker, fotgjengere osv.)	Lav	En komponentmeny på toppen av skjermen under headermenyen for alle tegnefunksjoner. Denne kan åpnes og skjules ved å bruke knappen helt til høyre i tegnemenyen. I komponentmenyen skal det ligge ikoner man kan "dra inn" i bildet, for eksempel: • Bil • Buss • Sykkel • Fotgjenger • Busk • Tre Samt forskjellige muligheter for tegning slik som: Gangvei, busslomme, sykkelvei etc.
+	For tegnefunksjoner: Mulighet til å fjerne komponenter (bil, buss og sykkel) som er "dratt inn" i bildet.	Lav	En "papirkurv"-knapp nederst på tegneområdet som blir uthevet og endrer farge når man drar den drabare komponenten over den. Hvert element må bli dratt dit individuelt. I tillegg blir alle drabare elementer slettet når man velger "papirkurv"-ikoner i tegnemenyen.

TABELL 13 - PRIORITERT LISTE OVER ILLUSTRASJONSFUNKSJONER

2.1.2. ANDRE FUNKSJONER

Andre funksjoner i appen, som er mindre viktige og ikke har sammenheng med illustrasjonsbiten av applikasjonen. Dette går på teoridelen av kjøreopplæring, som å vise fram ulike trafikkskilt, læreplanmål, forskrifter osv.

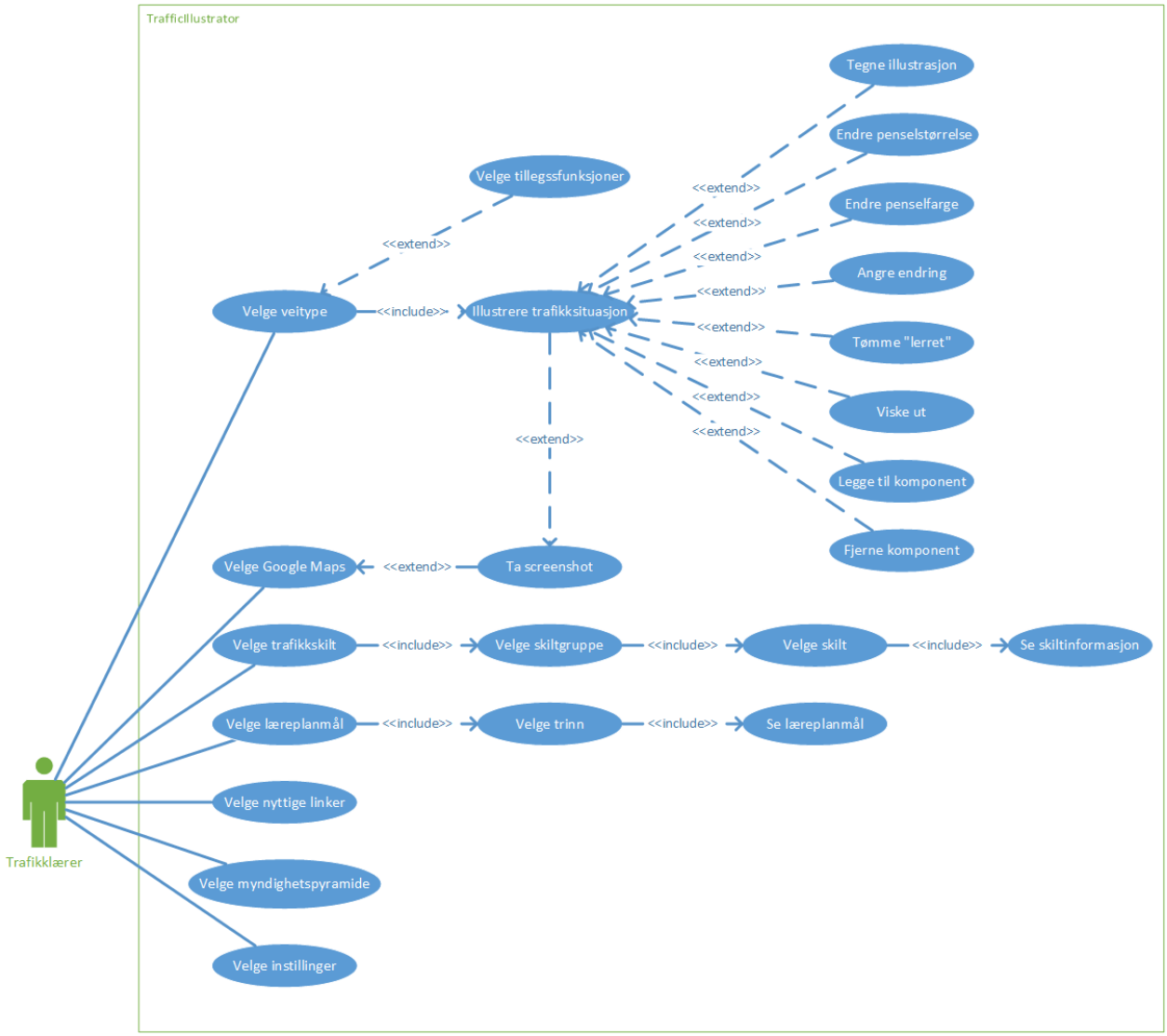
	KRAV	PRIORITET	FORESLÅTT LØSNING
	Trafikkskilt		
+	En oversikt over ulike trafikkskilt, fordelt på de ulike skiltgruppene.	Medium	En link *skiltgrupper* på startsiden. Da kommer man til en side med en meny til venstre på skjermen, med de ulike skiltgruppene. For hver skiltgruppe får man opp et galleri med alle skiltene som tilhører gruppen. Trykker man på et skilt blir det forstørret slik at man ser det bedre. Trykker man på skiltet en gang til, blir det forminsket igjen.
+	For hvert skilt: En kort beskrivelse av hva skiltet betyr.	Lav	Etter man har forstørret et skilt (trykket på det i galleriet) kan man trykke på et *informasjons*-ikon på selve skiltet. Da vil det komme en tekst foran skiltet (som en litt transparent modal). For å komme ut fra den tekstlige beskrivelsen, kan man enten trykke på krysset (X-ikonet) på modalen eller trykke utenfor modalen.
	Andre funksjoner (teori rundt trafikkopplæring)		
+	Oversikt over læreplanmålene for trafikkopplæringen	Lav	En link *læreplanmål* på startsiden. Man kommer da til en side med en meny til venstre på skjermen, der man kan velge mellom de ulike trinnene (trinn 1, trinn 2, trinn 3, trinn 4). Velger man for eksempel *trinn 1* kommer læreplanmålene opp som "bulletpoints".
+	Oversikt over myndighetspyramiden		En link *myndighetspyramiden* på startsiden. Man kommer da til en ganske enkel side som kun inneholder et bilde som viser myndighetspyramiden.
	Nyttige lenker		

+	Linker til viktige forskrifter for trafikk og trafikkopplæring	Lav	Linker på startsiden. Trykker man på disse blir det åpnet en ny tab i nettleseren. Følgende lenker skal være tilgjengelig: • Trafikkopplæringsforskriften • Trafikkregler • Veitrafikkloven • Skiltforskriften (i tilfelle skiltene i appen ikke er oppdatert)
	Google Maps		
+	Mulighet til StreetView for å vise eleven en spesifikk rundkjøring/kryss o.l. hen har kjørt i	Lav	Link på startsiden for *Google Maps*. Kan da gå inn og vise at "nå er vi akkurat i den rundkjøringen". Kan deretter "fryse bildet" og tegne oppå dette.

TABELL 14 - PRIORITERT LISTE OVER ANDRE, MINDRE VIKTIGE FUNKSJONER

2.2. USE CASE UML DIAGRAM

Under er et Use Case UML-diagram over appens bruksområder og sammenhengen mellom de ulike funksjonene. Dette er kun et overordnet bilde som viser den tenkte funksjonaliteten, ikke nødvendigvis hvordan den faktiske applikasjonens funksjoner kommer til å henge sammen.



FIGUR 3 - USE CASE-DIAGRAM OVER APPLIKASJONENS BRUKSOMRÅDER

3. DETALJERTE KRAV

Denne delen inneholder detaljerte beskrivelser av brukerkravene til systemet, og det er ut ifra disse beskrivelsene vi skal utvikle systemet.

De detaljerte kravene, eller beskrivelsene, vil også være utgangspunkt for å kunne vurdere om vi har lyktes med å oppfylle brukernes krav til applikasjonen.

3.1. USE CASE TEKSTLIG BESKRIVELSE AV HVERT USE CASE

En Use Case er en tekstlig beskrivelse av forskjellige brukersituasjoner i applikasjonen, altså hva brukeren skal kunne gjøre og hvordan applikasjonen skal respondere. Hver Use Case er relatert til en spesifikk del av applikasjonen.

UC 1: ILLUSTRERE TRAFIKKSITUASJON, VEIKRYSS

Navn:	**UC 1: Illustrere trafikksituasjon, veikryss**
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert
Postbetingelser	Trafikksituasjonen er ferdig illustrert
Hovedflyt	1. Trafikklærer velger veikryss fra hovedmenyen 2. Trafikklærer benytter standard type veikryss (høyreregel) 3. Trafikklærer benytter standard utforming (X-kryss) 4. Trafikklærer benytter standard veioppmerking 5. Trafikklærer drar bilvektor ut i veikonstruksjonen 6. Trafikklærer tegner retningen bilen kjører 7. Trafikklærer trykker på Hjem-knappen 8. Trafikklærer er tilbake i hovedmeny
Sideløp	2.1 Trafikklærer velger en annen type veikryss (forkjørskryss eller lyskryss) 3.1 Trafikklærer velger en annen utforming (T- eller Y-kryss) 4.1 Trafikklærer velger en annen type veioppmerking (gangfelt, sykkelfelt eller busslomme) 5.1 Trafikklærer tegner kjøretøy og retning
Unntak	

TABELL 15 - UC1: ILLUSTRERE TRAFIKKSITUASJON, VEIKRYSS

UC 2: ILLUSTRERE TRAFIKKSITUASJON, RUNDKJØRING

Navn:	UC 2: Illustrere trafikksituasjon, rundkjøring
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert
Postbetingelser	Trafikksituasjonen er ferdig illustrert
Hovedflyt	1. Trafikklærer velger rundkjøring fra hovedmenyen 2. Trafikklærer benytter standard type rundkjøring (ett-felts) 3. Trafikklærer benytter standard veioppmerking 4. Trafikklærer drar bilvektor ut i veikonstruksjonen 5. Trafikklærer tegner retningen bilen kjører 6. Trafikklærer trykker på Hjem-knappen 7. Trafikklærer er tilbake i hovedmeny
Sideløp	2.1 Trafikklærer velger en annen type veikryss (to-felts) 3.1 Trafikklærer velger en annen type veioppmerking (gangfelt, sykkelfelt eller busslomme) 4.1 Trafikklærer tegner kjøretøy og retning
Unntak	

TABELL 16 - UC 2: ILLUSTRERE TRAFIKKSITUASJON, RUNDKJØRING

UC 3: ILLUSTRERE TRAFIKKSITUASJON, VEI

Navn:	UC 3: Illustrere trafikksituasjon, vei
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert
Postbetingelser	Trafikksituasjonen er ferdig illustrert
Hovedflyt	1. Trafikklærer velger "Vei" fra hovedmenyen 2. Trafikklærer benytter standard type vei-situasjon (landevei) 3. Trafikklærer benytter standard veioppmerking 4. Trafikklærer drar bilvektor ut i veikonstruksjonen 5. Trafikklærer tegner retningen bilen kjører 6. Trafikklærer trykker på Hjem-knappen

	7. Trafikklærer er tilbake i hovedmeny
Sideløp	2.1 Trafikklærer velger en annen type veisituasjon (avkjøringsrampe, påkjøringsrampe) 3.1 Trafikklærer velger en annen type veioppmerking (gangfelt, sykkelfelt eller busslomme) 4.1 Trafikklærer tegner kjøretøy og retning
Unntak	

TABELL 17 - UC 3: ILLUSTRERE TRAFIKKSITUASJON, VEI

UC 4: VISE TRAFIKKSKILT OG INFORMASJON OM SKILT

Navn:	**UC 4: Vise trafikkskilt og informasjon om skilt**
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert
Postbetingelser	Viser trafikkskilt og informasjon om skiltet
Hovedflyt	1. Trafikklærer velger "Trafikkskilt" fra hovedmenyen 2. Applikasjonen viser vindu med forskjellige skilt, gruppert etter type 3. Trafikklærer trykker på ønsket skilt 4. Applikasjonen viser skiltet i stor størrelse 5. Trafikklærer trykker på informasjons-ikonet på skiltet 6. Applikasjonen viser informasjon om skiltet 7. Trafikklærer går tilbake til hovedmeny
Sideløp	3.1. Trafikklærer trykker på hurtigtast for skiltgruppe 3.1.1. Trafikklærer velger skilt 3.1.2. Applikasjonen viser skiltinformasjon 3.1.3. Trafikklærer går tilbake til hovedmenyen 5.1. Trafikklærer trykker på et annet skilt 5.1.1. Applikasjonen viser skiltet i stor størrelse 5.1.2 Trafikklærer går tilbake til hovedmeny

TABELL 18 - UC4: VISE TRAFIKKSKILT OG INFORMASJON OM SKILT

UC 5: VISE POSISJON PÅ GOOGLE MAPS OG TEGNE OVER KARTET

Navn:	UC 5: Vise posisjon på Google Maps og tegne over kartet
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert, brukeren har godtatt deling av posisjonsdata, applikasjonen har tilgang til Internett
Postbetingelser	Trafikksituasjonen er ferdig illustrert
Hovedflyt	1. Trafikklærer velger "Google Maps" fra hovedmenyen 2. Applikasjonen viser satellittkart over der brukeren befinner seg 3. Trafikklærer trykker på knapp for å "fryse bildet" 4. Applikasjonen viser tegnemenyen 5. Trafikklærer tegner over kartet 6. Trafikklærer går tilbake til hovedmeny
Sideløp	3.1. Trafikklærer zoomer inn på kartet til ønsket utsnitt 3.1.1. Trafikklærer trykker på knapp for å "fryse bildet" 3.1.2. Applikasjonen viser tegnemenyen 3.1.3. Trafikklærer tegner over kartet 3.1.4. Trafikklærer går tilbake til hovedmenyen 3.2. Trafikklærer velger en annen kartposisjon enn sin egen 3.2.1. Trafikklærer trykker på knapp for å "fryse bildet" 3.2.2. Applikasjonen viser tegnemenyen 3.2.3. Trafikklærer tegner over kartet 3.2.4. Trafikklærer går tilbake til hovedmenyen 5.1. Trafikklærer går tilbake til hovedmenyen
Unntak	

TABELL 19 - UC5: VISE POSISJON PÅ GOOGLE MAPS OG TEGNE OVER KARTET

UC 6: VISE LÆREPLANMÅL FOR KLASSE B

Navn:	UC 6: Vise læreplanmål for klasse B
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert

Postbetingelser	Viser læreplanmålene
Hovedflyt	1. Trafikklærer velger "Læreplanmål" fra hovedmenyen 2. Applikasjonen viser de fire ulike trinnene 3. Trafikklærer trykker på ønsket trinn 4. Applikasjonen viser alle læreplanmål for valgt trinn 5. Trafikklærer går tilbake til hovedmeny
Sideløp	5.1. Trafikklærer trykker på et annet trinn 5.1.1. Applikasjonen viser alle læreplanmål for valgt trinn 5.1.2. Trafikklærer går tilbake til hovedmenyen
Unntak	

TABELL 20 - UC6: VISE LÆREPLANMÅLENE

UC 7: VISE NYTTIGE LENKER TIL VIKTIGE TRAFIKKOPPLÆRINGSFORSKRIFTER

Navn:	**UC 7: Vise nyttige lenker til viktige trafikkopplæringforskrifter**
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert, Appen er koblet til nett
Postbetingelser	Åpner opp nettleser til valgt forskrift
Hovedflyt	1. Bruker velger fra hovedmenyen «nyttige lenker» 2. Bruker får opp en skjerm med lenker til forskjellige forskrifter og eventuelt andre nyttige nettlenker. 3. Bruker trykker på en lenke 4. Lenken åpnes opp i standard nettleser og appen legges i bakgrunnen
Sideløp	Ingen sideløp her
Unntak	

TABELL 21 - UC7: VISE LENKER TIL FORSKRIFTER

UC 8: VISE MYNDIGHETSPYRAMIDEN

Navn:	**UC 8: Vise myndighetspyramiden**
Aktør(er)	Trafikklærer

Prebetingelse	Applikasjonen er installert
Postbetingelser	Se bilde over myndighetspyramiden
Hovedflyt	1. Bruker er på hovedsiden 2. Bruker trykker på siden til myndighetspyramiden 3. En ny side åpnes opp som viser et bilde av dette med litt forklaring hva den består av
Sideløp	Ingen sideløp her
Unntak	

TABELL 22 - UC8: VISE MYNDIGHETSPYRAMIDEN

UC 9: SETTE EGNE INNSTILLINGER FOR APPEN

Navn:	**UC 9: Sette egne innstillinger for appen**
Aktør(er)	Trafikklærer
Prebetingelse	Applikasjonen er installert
Postbetingelser	Se en side med alle tilgjengelige innstillinger
Hovedflyt	1. Bruker trykker enten på tannhjulet på startsiden eller ved å trykke på «innstillinger» via hamburgermenyen 2. Bruker får opp en skjerm med mulige innstillinger 3. Bruker kan endre på innstillinger som blir lagret direkte i AsyncStorage (lokalt på Android enheten) 4. Bruker kan trykke på «Åpne velger» 5. En modal vil da åpne seg og bruker kan velge opptil 20 bilder/ikoner som kan brukes som drabare elementer på tegneskjermen 6. Bruker velger å lagre, avmarkere alt eller bare krysse seg ut av modalen uten å lagre.
Sideløp	Ingen sideløp
Unntak	

TABELL 23 - UC9: SETTE EGNE INNSTILLINGER FOR APPEN

3.2. TILLEGGSSPESIFIKASJON AV KRAV (SUPPLEMENTARY SPECIFICATION)

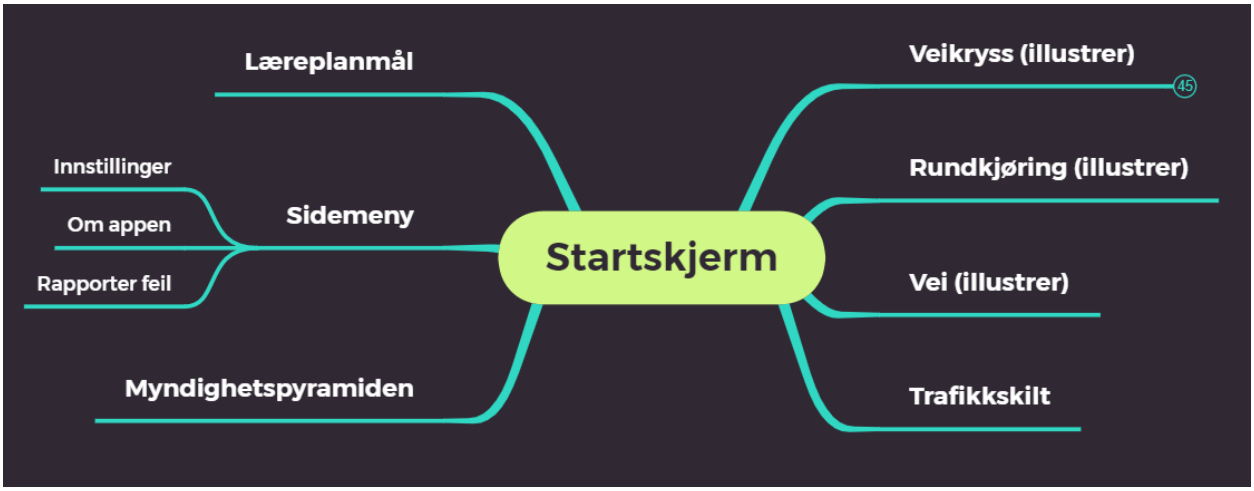
- Applikasjonen skal kunne kjøre på alle enheter som har Android operativsystem med minste operativsystem Android Q (10)

- Den skal være stabil (ikke krasje, fryse)
- Den skal være rask/høy responstid (bilder/trafikksituasjoner skal rendres kjapt)
- Den skal fungere uten Internett/mobilnett
- Muligheten til å trykke på nyttige lenker som åpner opp en nettleser forutsetter at nettbrettet er koblet på nett for at disse skal kunne åpne seg. Appen skal fortsatt fungere uansett
- Den skal benytte så liten lagringsplass som mulig, ved å redusere filstørrelsen på illustrasjoner og annen grafikk
- Det skal være brukervennlig (oppføre seg på samme måte som andre android-applikasjoner)
- Den skal oppfylle kravene til universell utforming

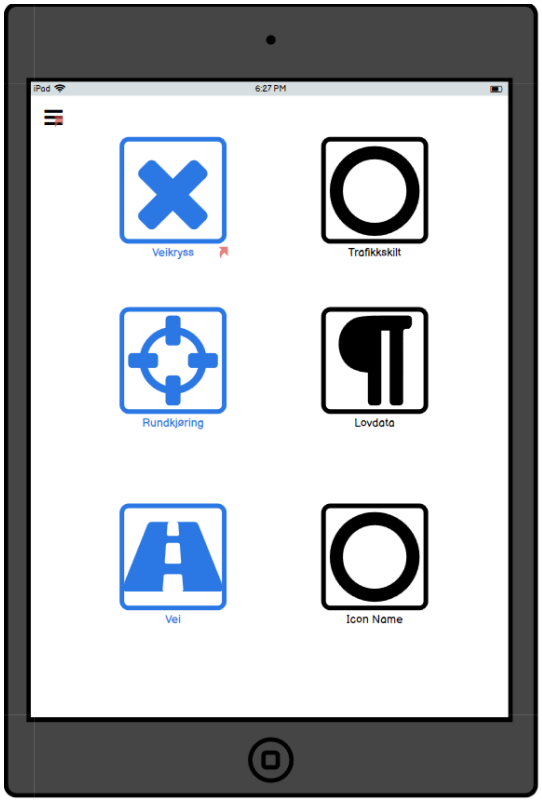
4. BESKRIVELSE AV BRUKERGRENSESNITT (WIREFRAMES)

I dette kapittelet beskrives hvordan tanken bak brukergrensesnittet er og se noen utkast av disse. Det blir vist illustrasjoner av applikasjonen, som er blitt laget som *wireframes* ved hjelp av et verktøy som heter Balsamiq. I denne delen vil det være bilder og en beskrivelse tilhørende hvert bilde. Her får man innsyn i det første utkastet av applikasjonsdesignet og hvordan en eventuell bruker vil interagere med brukergrensesnittet.

Funksjoner som skal være tilgjengelig fra startmenyen:

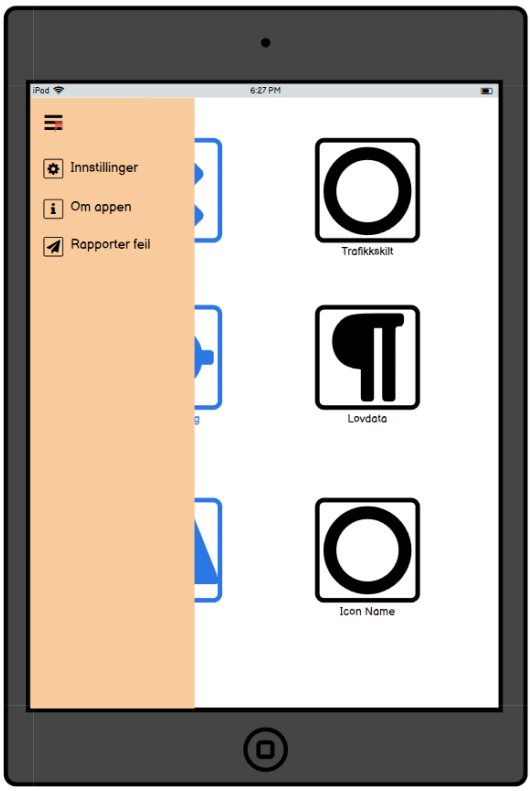


4.1. STARTSKJERM OG HOVEDMENY

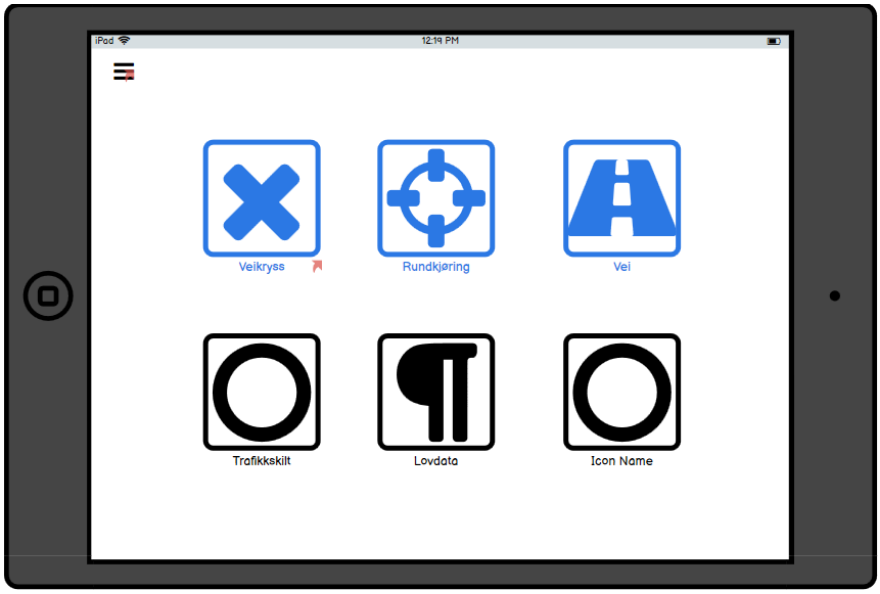


Startskjermen med alle appens funksjoner. Øverst til venstre i bildet er det en "hamburgermeny" man kan åpne ved å enten trykke på ikonet eller "dra" menyen ut fra venstresiden.

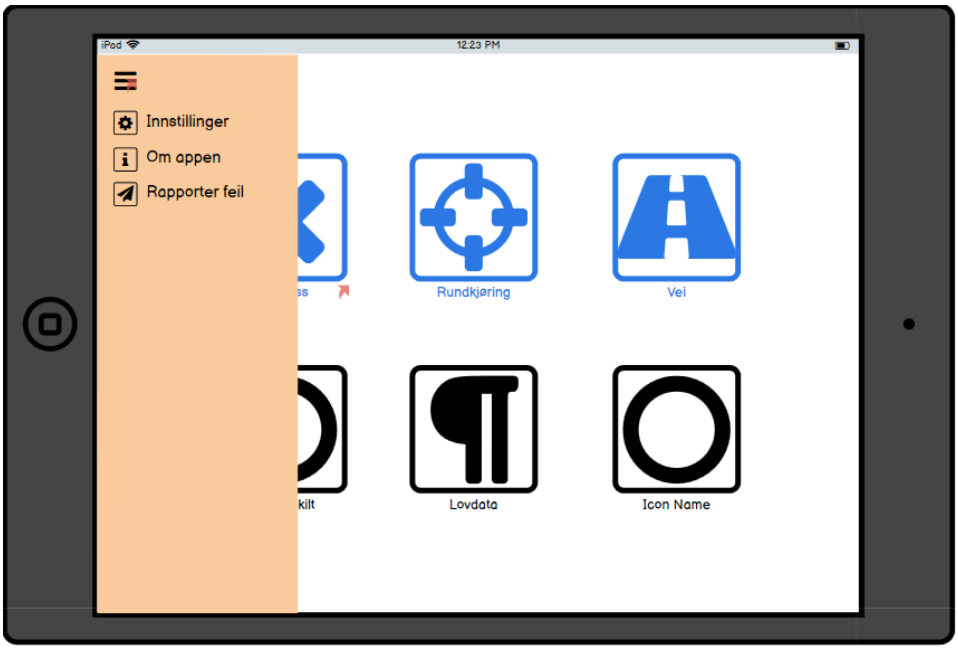
Ikonene/lenkene på startskjermen skal i stående format grupperes under hverandre, ut ifra hvilken funksjon de har. Linker til sider for tegning er markert med blå farge.



Startskjermen med sidemenyen åpnet. Her er tanken at brukeren skal kunne åpne menyen både gjennom en knapp oppe til venstre eller ved å 'sveipe' til høyre fra kanten av skjermen. Her vil det ligge linker som tar dem direkte til de sidene dem ønsker å bruke



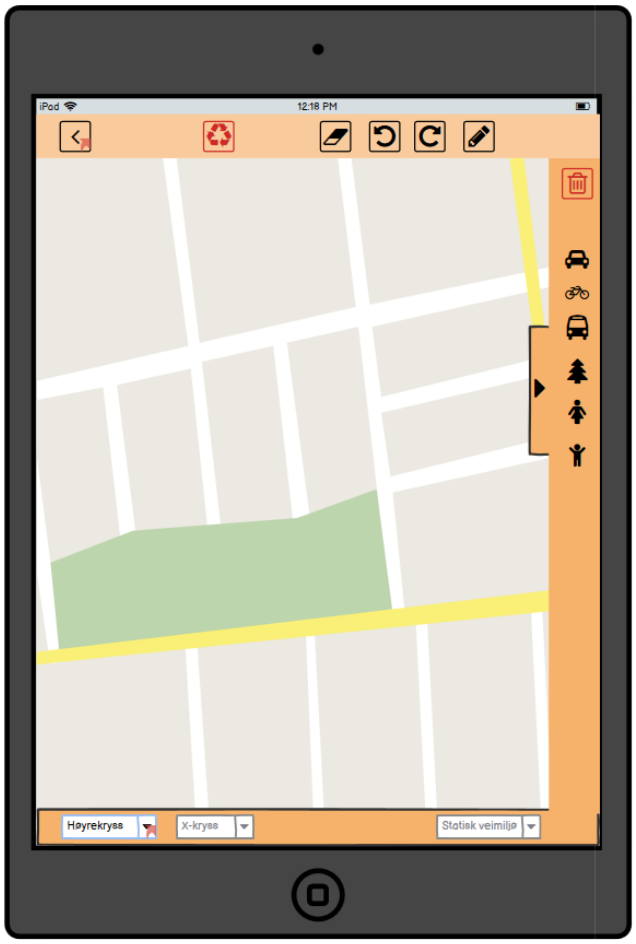
Startskjermen i liggende format. Her skal de linkene som i stående format var gruppert under hverandre, grupperes *etter* hverandre.



Samme som over, bare med sidemenyen åpen.

4.2. Siden for "Veikryss"

Åpnes fra hovedmenyen via linken **veikryss**.

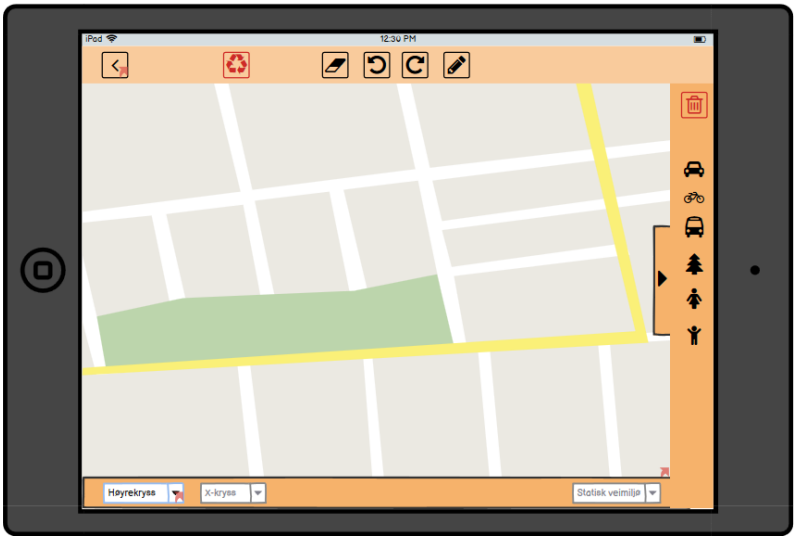


Øverst til venstre er det en knapp for å gå tilbake til startskjermen.

Tegneverktøylinje øverst på skjermen, med funksjoner for viskelær, angre-knapp, redo-knapp og pennefarge.

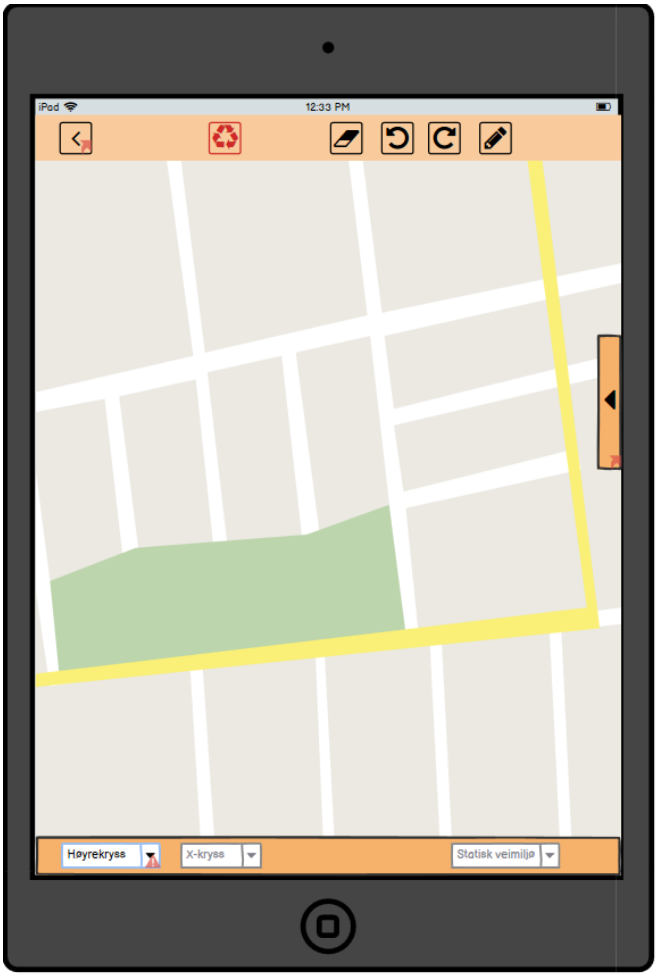
Komponentmeny til høyre. Her kan man trykke på en komponent og "dra den inn" i bildet. For å fjerne komponenten kan man dra den til papirkurv-ikonet (i rødt øverst i komponentmenyen). For å skjule komponentmenyen trykker man på pil-ikonet til venstre for menyen.

Nederst en verktøylinje for å velge type kryss, utforming på kryss. I tillegg en knapp for å velge elementer fra det statiske veimiljøet som sykkelfelt og busslomme. Alle disse blir tegnet inn riktig i bildet hvis man velger de.



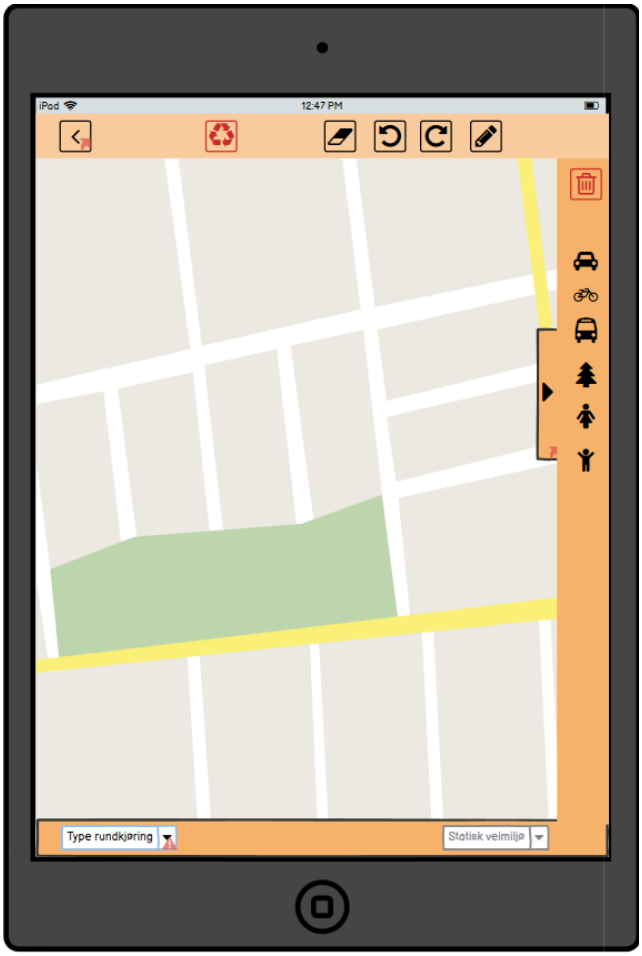
Samme som over, bare for liggende modus:

Med komponentmenyen til høyre skjult



Man får da en renere uttrykk med større tegneflate. Komponentmenyen kan enkelt åpnes igjen ved å trykke på pil-ikonet til høyre i skjermen.

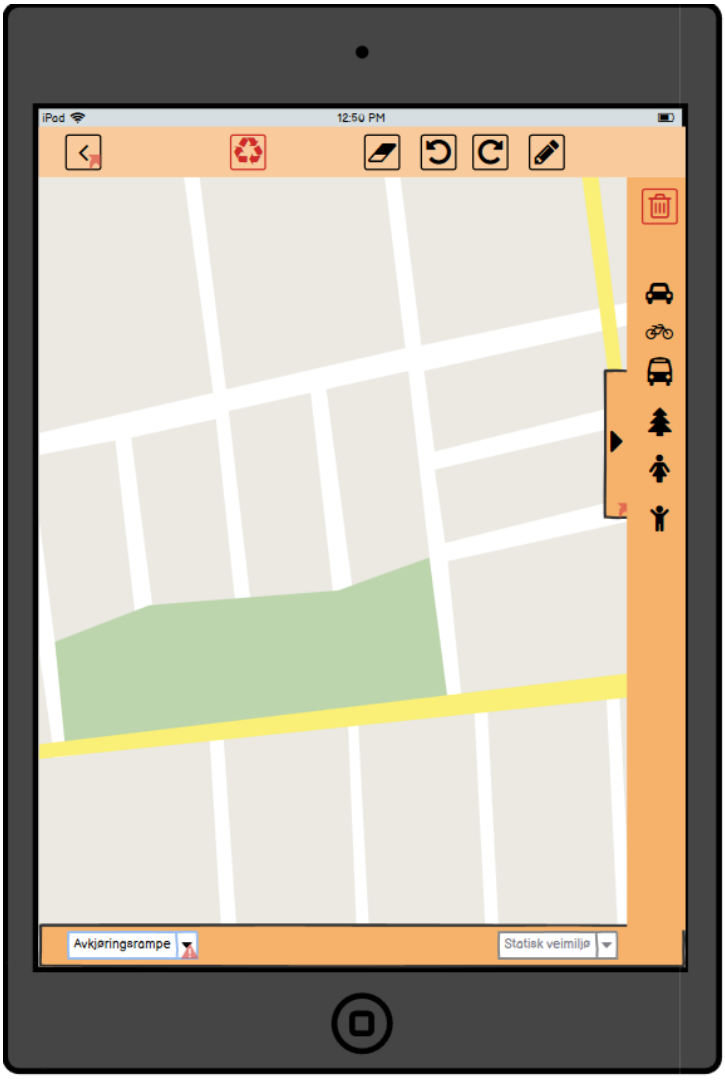
4.3. Siden for "Rundkjøring"



Samme utforming som for veikryss-siden, bortsett fra at verktøylinja nederst på skjermen inneholder andre valg spesifikt for rundkjøring i stedet for veikryss.

Det samme som for veikryss-siden gjelder også her: Man kan skjule komponentmenyen til høyre hvis man ønsker det. For liggende format gjelder også samme oppsett som for veikryss-siden.

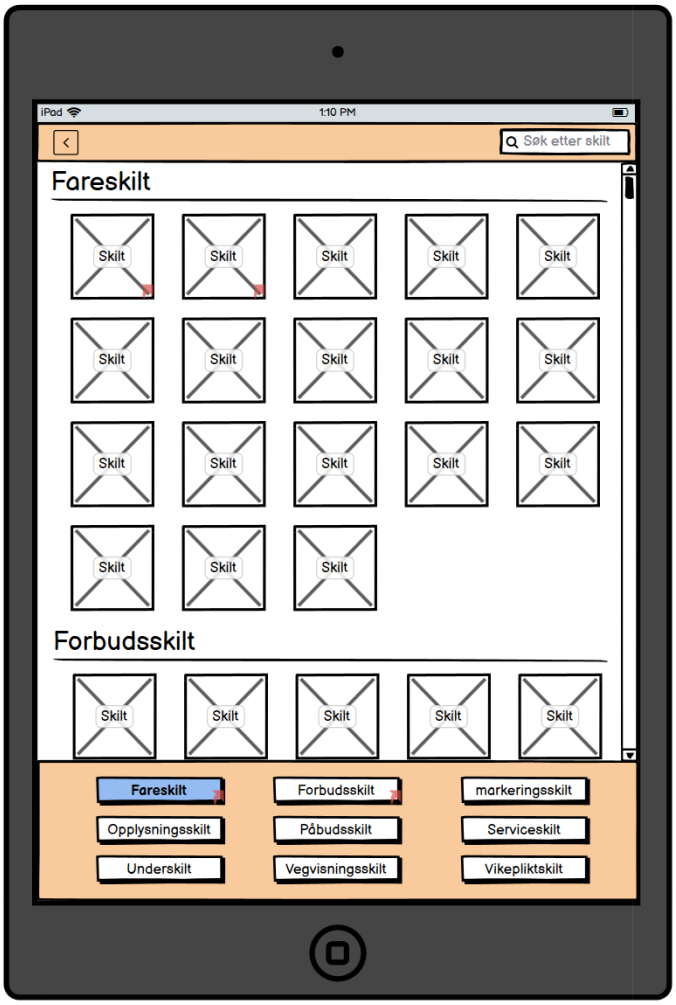
4.4. Siden for "Vei"



Samme oppsett på siden som for veikryss og rundkjøring. Den eneste forskjellen her er på verktøylinja nederst på skjermen, der man får opp valg for type vei (for eksempel påkjøringsrampe, avkjøringsrampe o.l. som ikke er hverken veikryss eller rundkjøring).

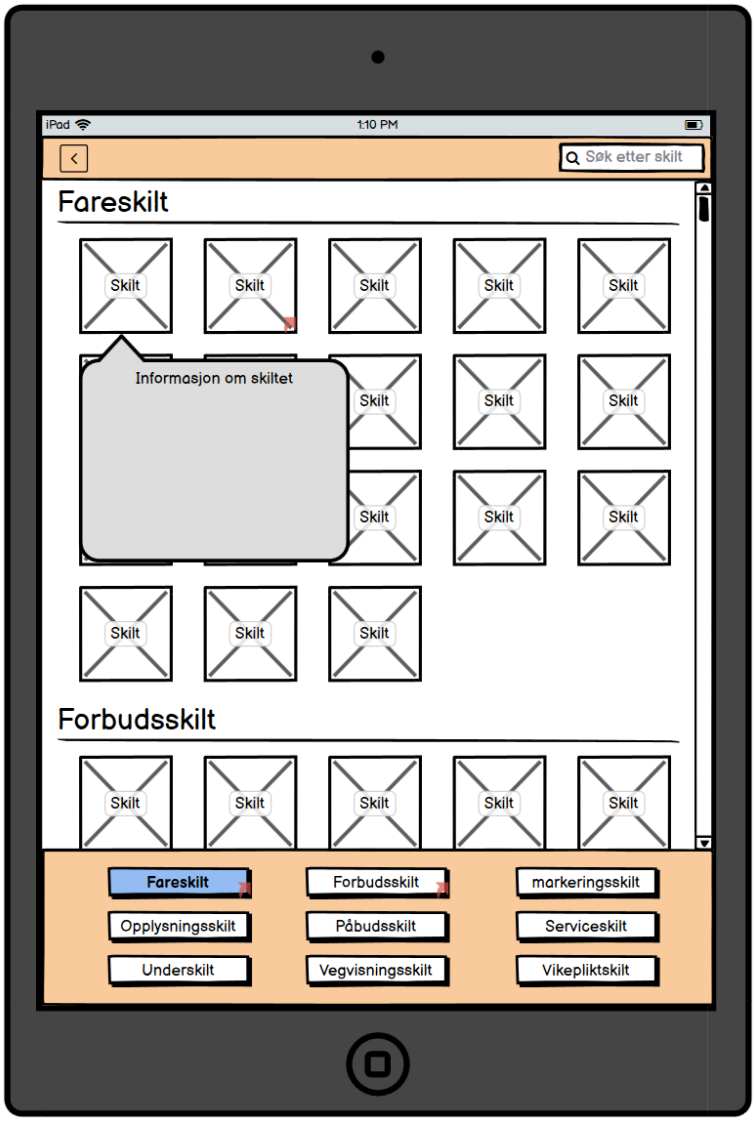
Det samme som for veikryss- og rundkjøring siden gjelder også her: Man kan skjule komponentmenyen til høyre hvis man ønsker det. For liggende format gjelder også samme oppsett som for veikryssiden og rundkjøringssiden.

4.4. Siden for "Trafikkskilt"



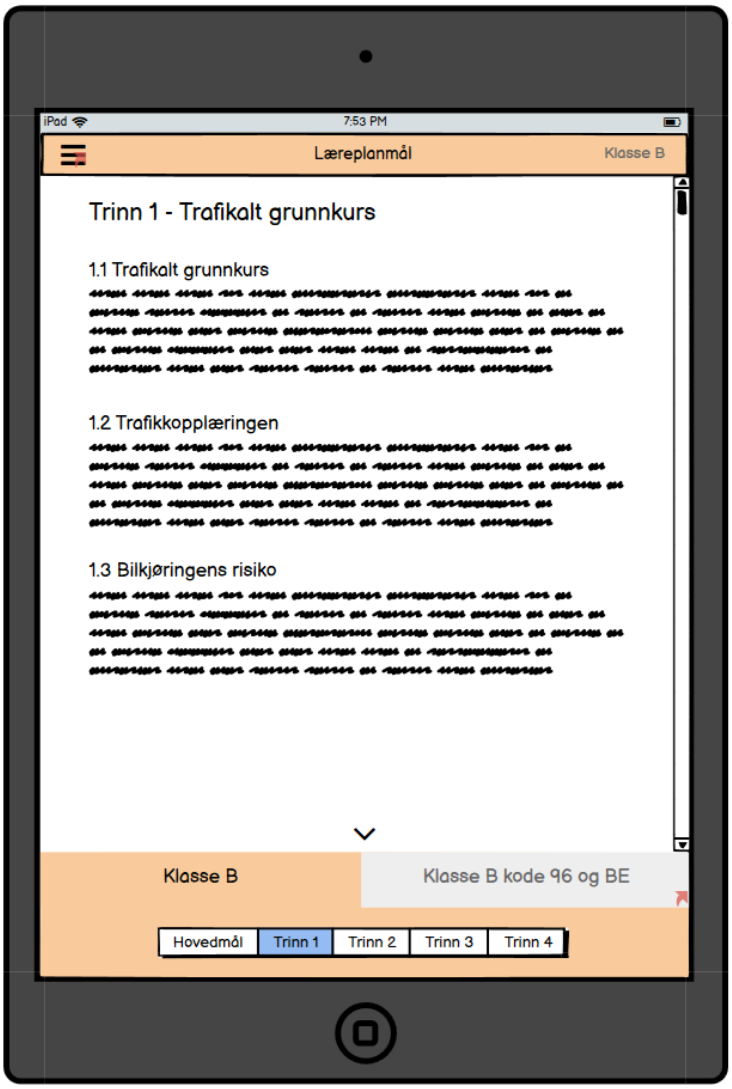
Ved å trykke på linken "Trafikkskilt" enten på startsiden eller i sidemenyen, kommer man til denne siden. Siden inneholder samtlige trafikkskilt, gruppert etter skiltgruppe.

En meny nederst på siden, med alle de ulike skiltgruppene. Trykker man for eksempel på "Forbudsskilt" i denne menyen, vil siden "scrolle" nedover eller oppover til denne skiltgruppen.



Ved å trykke på et skilt, får man opp en modal med informasjon om skiltet.

4.4. Siden for "Læreplanmål"



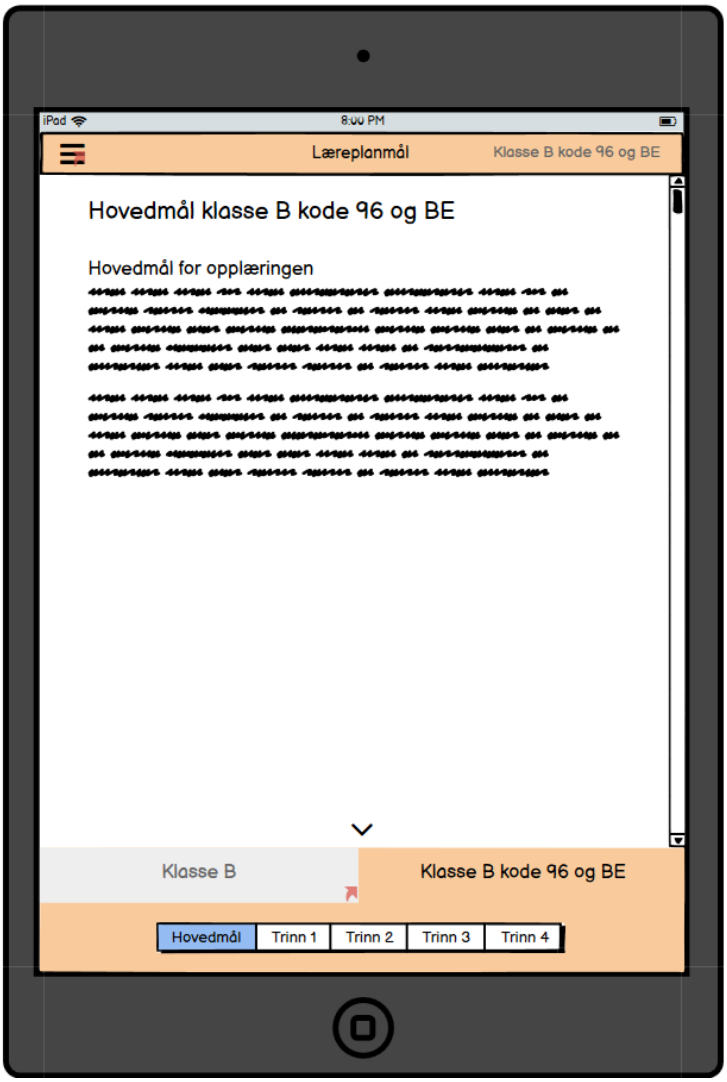
Ved å trykke på linken "Læreplanmål" enten på startsiden eller i sidemenyen, kommer man til denne siden. Siden inneholder læreplanmålene for opplæringen i klasse B, klasse B kode 96 og BE.

I menyen nederst på siden velger man først hvilken førerkortklasse man ønsker å se læreplanmålene til. Klasse B er valgt som standard, da dette er førerkortklassen som er vanligst.

Videre er læreplanmålene for de ulike førerkortklassene gruppert etter trinn i opplæringen i tillegg til et eget punkt som inneholder hovedmålene for opplæringen for hver klasse. Her er "hovedmål" valgt som standard.

I headeren står det hvilken førerkortklasse som er valgt, slik at det skal være enkelt for brukeren å holde oversikt over dette.

Siden vil inneholde en scrollbar, da enkelte trinn inneholder ganske mange mål, og man må da scrolle seg nedover på siden for å se alle.

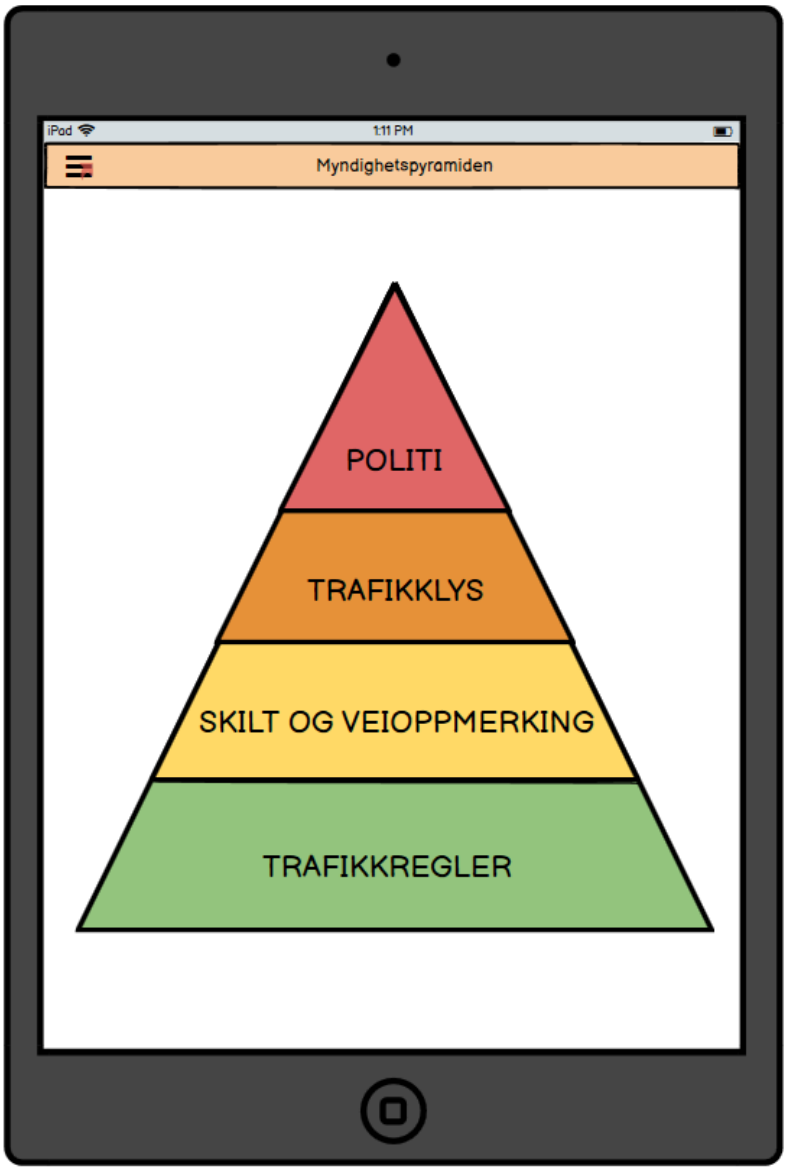


Siden for læreplanmål når "Klasse B kode 96 og BE" er valgt i menyen nederst på skjermen.

Teksten til høyre i headeren vil da endres til å reflektere dette, og fargene på linkene i menyen vil endres til å vise at det nå er denne klassen som er valgt.

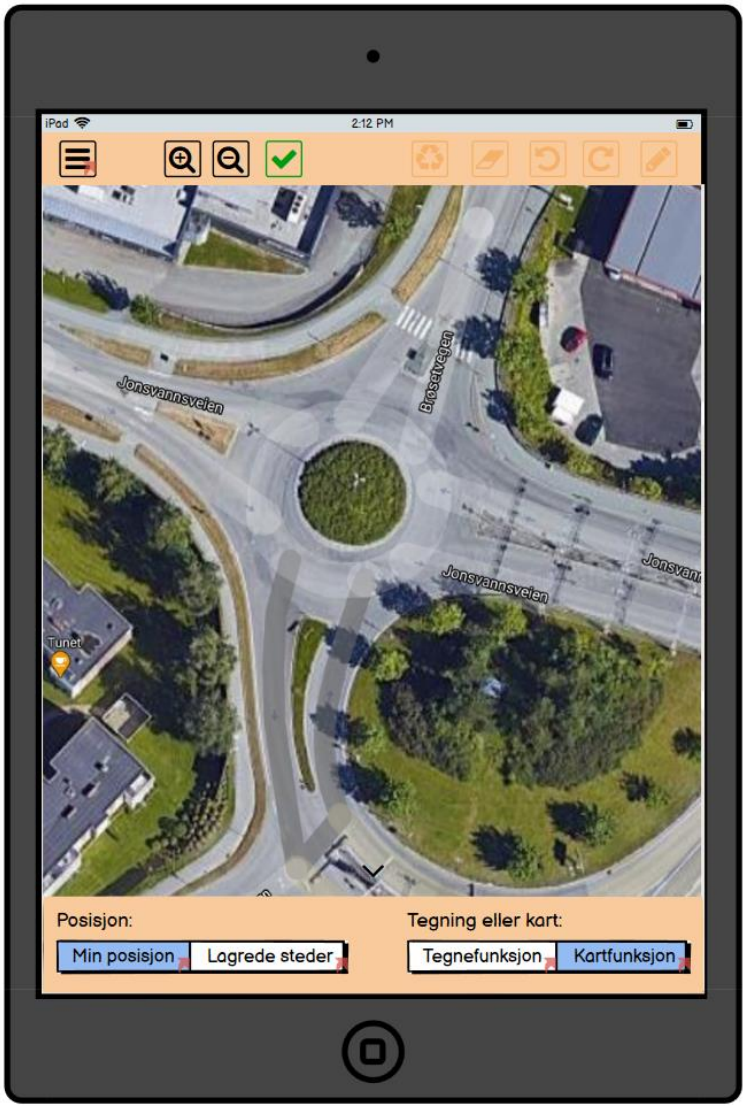
Det vil da som standard bli valgt "hovedmål" når man endrer førerkortklasse.

4.5. Siden for "Myndighetspyramiden"



Ved å trykke på linken "Myndighetspyramiden" enten på startsiden eller i sidemenyen, kommer man til denne siden. Denne siden er veldig enkel og inneholder kun et bilde som viser myndighetspyramiden.

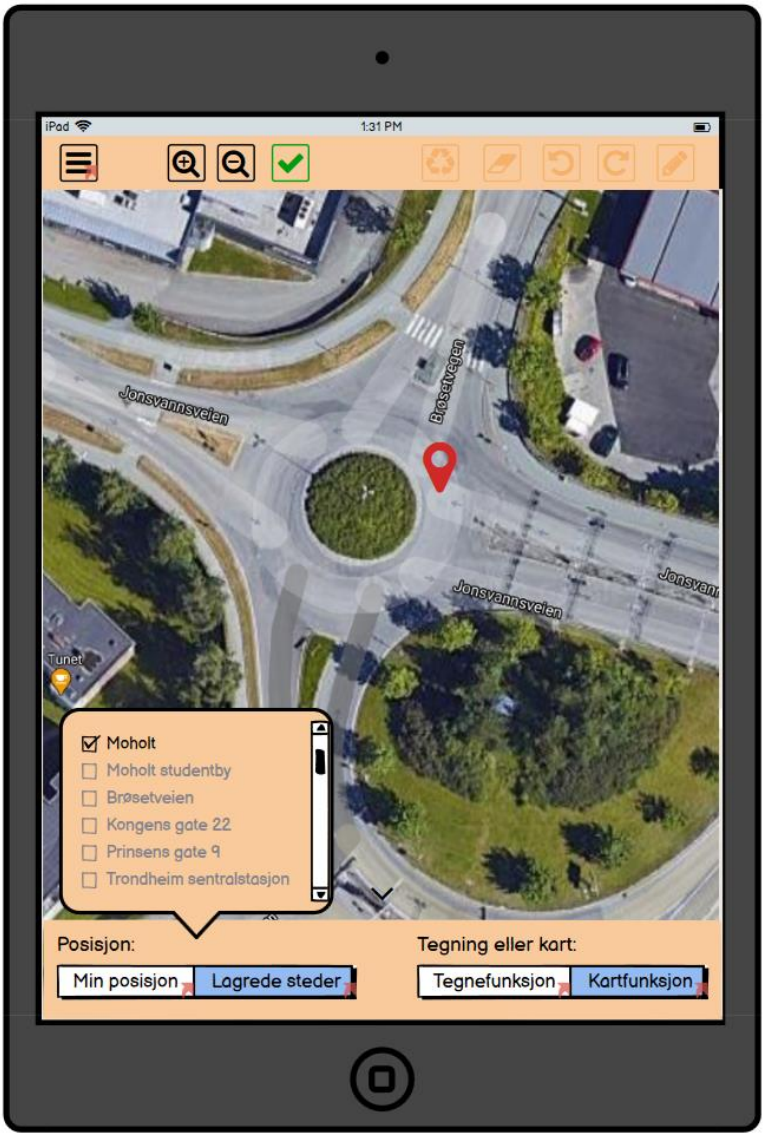
4.6. Siden for "Google Maps"



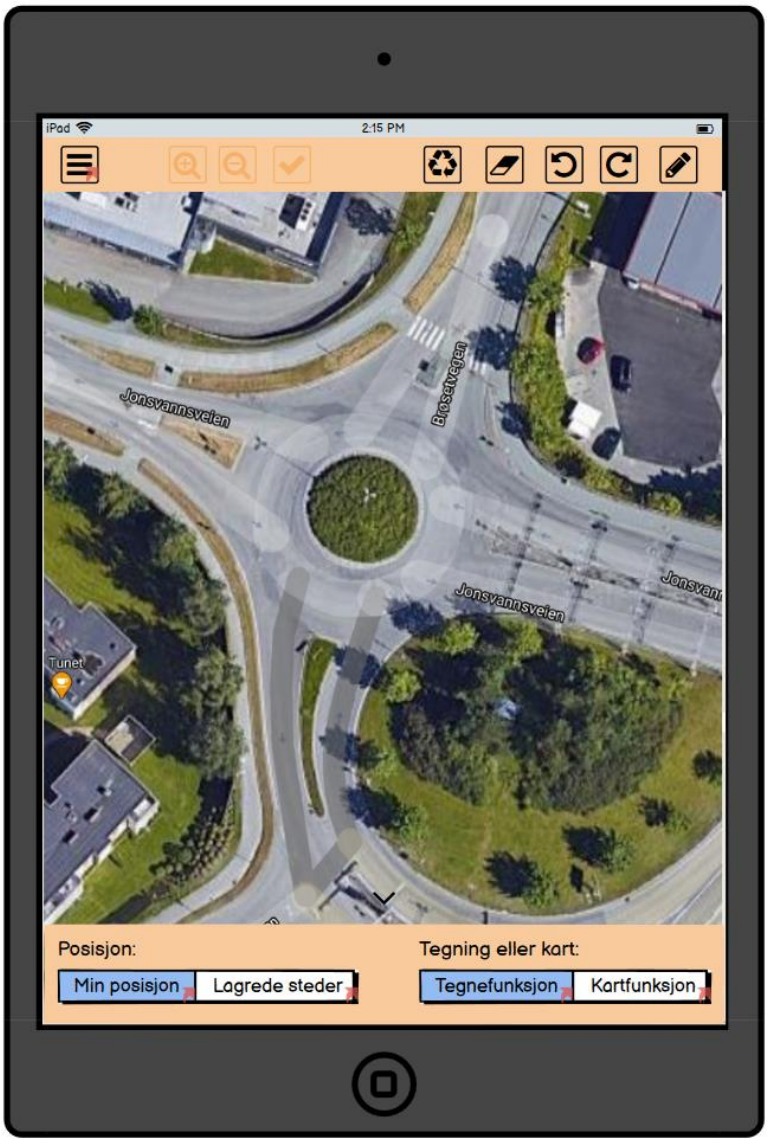
Ved å trykke på linken "Google Maps" enten på startsiden eller i sidemenyen, kommer man til denne siden.

Når man kommer inn på siden er kartfunksjonen valgt som standard. Det er ikoner i headeren for å zoome ut og inn på kartet (man kan også zoome med fingrene). Det er også et ikon i headeren for å lagre nåværende sted. Det vil da komme opp en kartmarkør som man drar til ønsket posisjon på bildet, og man skriver inn ønsket navn på det lagrede stedet.

I menyen nederst på siden kan man velge å gå til posisjonen man er på nå, eller velge en posisjon fra lagrede steder. Man kan også velge her om man ønsker å være på kartfunksjonen (som man er her), eller å fryse nåværende utsnitt og gå til tegnefunksjonen.



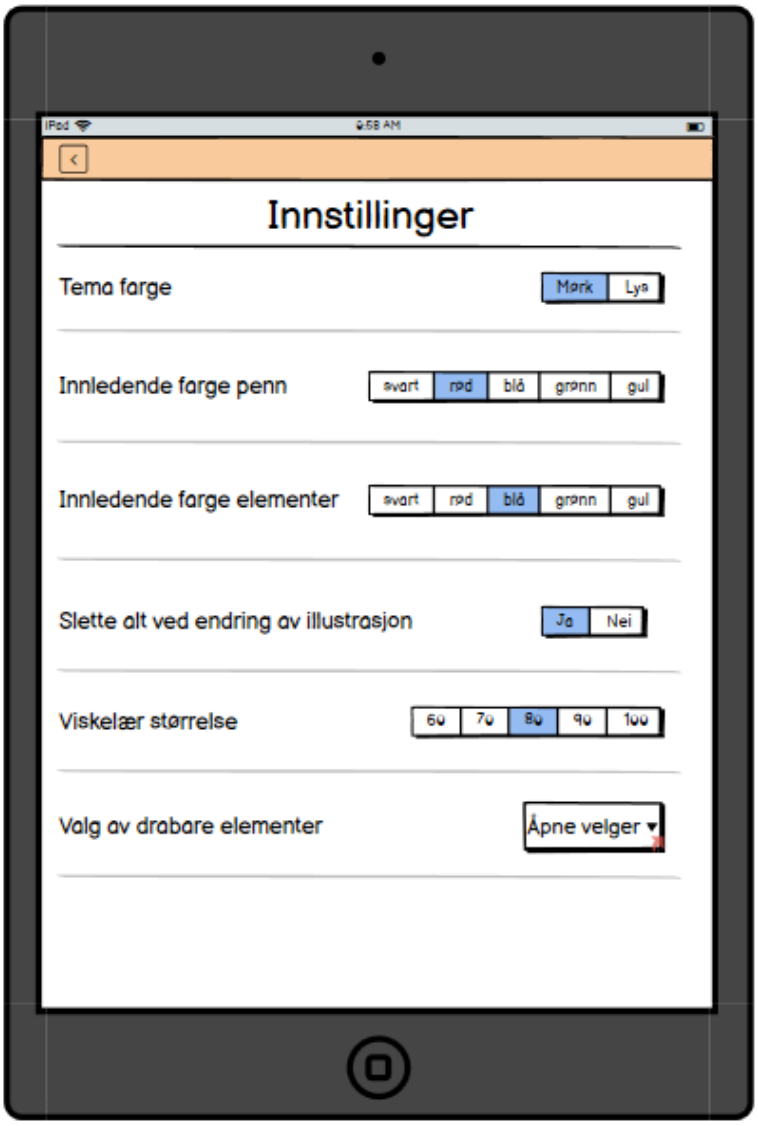
Her er hvordan det vil se ut hvis man trykker på "Lagrede steder". Man velger da en i listen, og kartutsnittet blir automatisk flyttet til valgt sted (forutsetter at man har lagret noen steder).



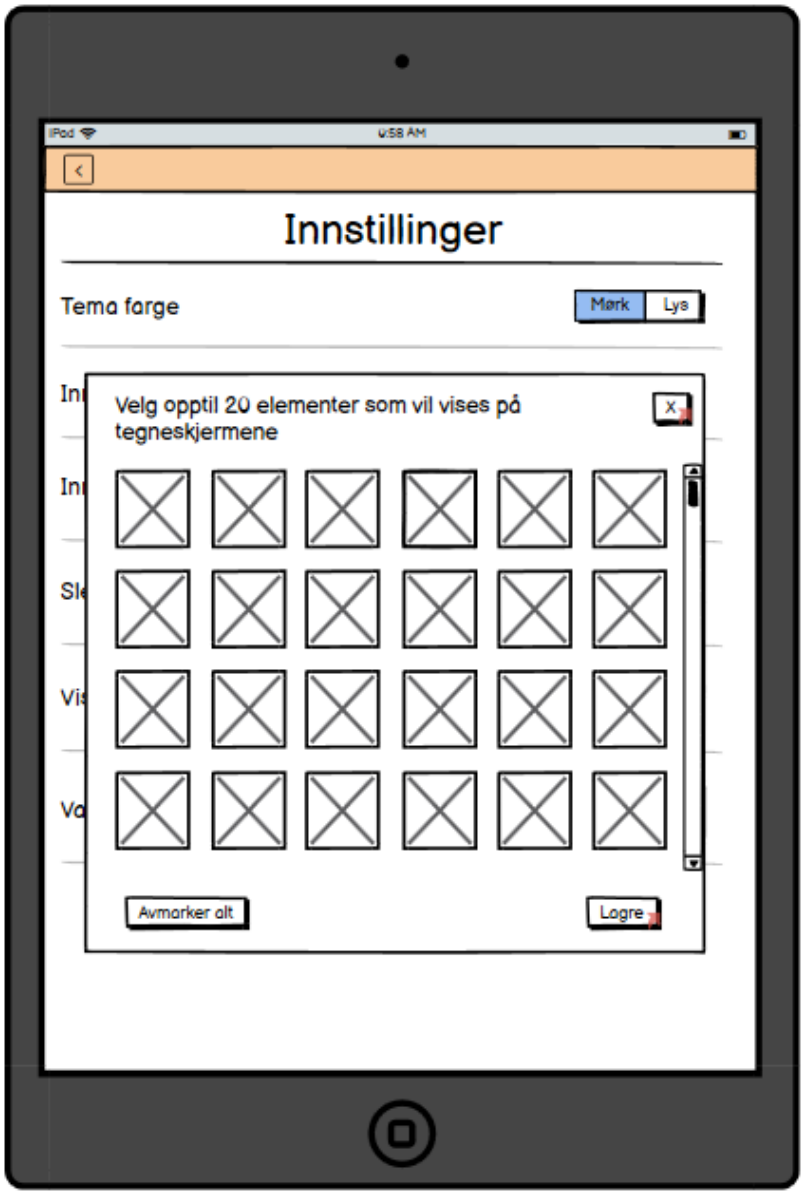
Her har man valgt "Kartfunksjon". Da vil kart-ikonene i headeren bli disablet, mens tegneikonene vil bli enablet. Kartutsnittet er her fryst og man kan tegne oppå bildet på samme måte som man kan på sidene for veikryss, rundkjøring osv.

4.7. Siden for "Innstillinger"

Denne siden kan finnes enten i hamburger menyen eller via startsiden som et lite tannhjul oppe i høyre hjørne.



På denne siden kan bruker velge forskjellige innstillinger som vil bli lagret lokalt og vil bli gjenbrukt hver gang brukeren åpner appen på nytt. Brukeren kan velge innledende farge på penneverktøy, drabare elementer, størrelse på viskelær, temafarge, om elementer/tegning skal bli slettet når illustrasjonen endres og til slutt hvilke elementer som skal vises på deres tegne skjerm.



Når bruker trykker på «åpne velger» så vil denne modal komme opp. Her kan bruker markere opptil 20 elementer som de bruker mest og som vil bli vist i den ene menyen på tegneskjermen. Bruker kan herfra avmerkere alt, lagre eller bare lukke for å ikke gjøre endringer.

5. REFERANSER

[1] C. Larman, "Chapter 6: Use Cases," in *Applying UML and Patterns - An introduction to object-oriented analysis and design and iterative development*, London, Pearson, 2004, pp. 61-100.

[2] S. W. Ambler, "Chapter 4 - UML Use-Case Diagrams," in *The Elements of UML*, Cambridge University Press, 2010, pp. 33-46.

DRIFTSDOKUMENTASJON

BACHELORPROSJEKT 087: UTVIKLING AV ANDROID-APPLIKASJON FOR TRAFIKKLÆRERE

Oliver Elias Nilssen, Joakim Heitmann Tronseth og Silje Tanemsmo

INNHOLDSFORTEGNELSE

1. INTRODUKSJON

Denne applikasjonen er resultatet av vårt bachelorprosjekt i Informatikk ved NTNU. Applikasjonen er skrevet for Android og er ment å brukes på nettbrett. Målgruppen for applikasjonen er trafikklærere i Norge, og tanken er at appen skal være et hjelpemiddel i trafikkopplæringen med særlig fokus på illustrering av ulike trafikksituasjoner. Vi har valgt å kalle applikasjonen for illusTrafikk, som er en sammenslåing av ordene *illustrasjon* og *trafikk*.

I dette dokumentet gir vi en overordnet beskrivelse av applikasjonen. Dette innebærer en oversikt over hvordan koden til applikasjonen er strukturert, hvilke avhengigheter applikasjonen har og hvordan de ulike komponentene i koden henger sammen. Hensikten med dette er å gjøre det lettere for andre å videreutvikle applikasjonen.

Som et tillegg til dette dokumentet har vi en detaljert dokumentasjon av alle komponentene i applikasjonen, med oversikt over alle props og metoder det er nødvendig å vite noe om. Denne dokumentasjonen er gjort tilgjengelig gjennom GitHub Pages på følgende url:

https://olivernilssen.github.io/trafikklearerapp/

Tilleggsdokumentasjonen er generert gjennom JSDoc. JSDoc er et API for å generere dokumentasjon av kode gjennom å legge inn kommentarer i selve koden [9]. På denne måten får man både dokumentert i selve koden og man får en mer strukturert oversikt over koden gjennom nettsiden som blir skapt.

1.1. OVERSIKT OVER INNHOLDET

Kapittel 2 – Utviklingsmiljø: Vi starter med å gå gjennom utviklingsmiljøet vi har benyttet i utviklingen av applikasjonen, og hvordan man skal gå fram for å sette opp og bruke miljøet.

Kapittel 3 – Google Play: Her beskriver vi hvordan vi har gått fram for å kunne legge ut applikasjonen på Google Play Store.

Kapittel 4 – Overordnet om applikasjonen: Her viser og forklarer vi filstrukturen og hvordan vi har navngitt ulike deler av koden. Deretter går vi gjennom avhengighetene til applikasjonen, som blant annet består av ulike tredjepartsbiblioteker vi har benyttet.

Kapittel 5 – Komponenter: I dette kapittelet beskriver vi hvordan de ulike komponentene i koden henger sammen. Dette er en overordnet framstilling som kun viser hvor i koden de ulike komponentene benyttes og en kort beskrivelse av hovedkomponentene og hvilken funksjon de har. Detaljerte beskrivelser av alle komponentene i koden er tilgjengelig gjennom vår JSDoc [1].

Kapittel 6 – Stilsetting av applikasjonen: Her beskriver vi hvordan vi har definert stilsetting på tvers av komponenter i applikasjonen. Vi har definert stilvariabler som farger, fontstørrelser og størrelse på knapper i egne filer som deretter benyttes av de fleste komponentene.

Kapittel 7 – Illustrasjoner og øvrig grafikk: I dette kapittelet går vi gjennom hvordan vi henter inn og bruker bilder og annen grafikk i applikasjonen.

[1] https://olivernilssen.github.io/trafikklearerapp/

Kapittel 8 – Python: En kort beskrivelse av hvordan vi brukte Python til å hente inn data/info fra Statens Vegvesen sine nettsider.

Kapittel 9 – Eksterne ressurser: Her beskriver vi hvilke ressurser vi har benyttet for å utvikle applikasjonen.

Kapittel 10 – Referanser: Oversikt over kildene vi har benyttet i rapporten.

JSDoc nettside: Detaljert dokumentasjon over alle komponentene applikasjonen består av.

1.2. FORKORTELSER OG DEFINISJONER

JSDoc	-	En API-dokumentasjonsgenerator for JavaScript. Man legger til dokumentasjonskommentarer direkte i kildekoden, og JSDoc-verktøyet skanner deretter kildekoden og genererer et HTML-dokumentasjonsnettsted [1].
Komponent	-	En komponent er et veldig grunnleggende element i React Native. Ved å dele en stor applikasjon inn i mange små komponenter blir koden enklere å lese og forstå.
JSON-fil	-	JSON (JavaScript Object Notation) er et tekstformat med lav overhead. Det er helt språkuavhengig, men bruker konvensjoner som er kjent fra andre programmeringsspråk. Disse egenskapene gjør JSON til et ideelt språk for datautveksling [2].
JS	-	JS (JavaScript) er en forkortelse for JavaScript-filer.

2. UTVIKLINGSMILJØ

Vi går her gjennom utviklingsmiljøet vi har benyttet i utviklingen av applikasjonen, og hvordan man skal gå fram for å sette opp og bruke miljøet.

Vi har benyttet følgende rammeverk og teknologier for å utvikle applikasjonen:

- React Native rammeverk
- Node.js – som inkluderer kommandolinjeverktøyene npm (Node Package Manager) for å installere tredjepartspakker og npx for å kjøre disse pakkene
- Android Studio: **Android Virtual Device (AVD)** og **Software Development Kit (SDK**)
- GitHub for versjonskontroll og samskriving av kode
- Visual Studio som IDE for koding

Man kan hente hele prosjektet ved å klone repositoriet fra GitHub. Da vil man få tilgang til kodebasen og for utenom node modulene, som installeres via kommandoen `>npm install` i prosjektets rotmappe (se kapittel 0 om filstrukturen). Dette vil installere alle tredjepartspakkene som kreves for å kjøre applikasjonen. Det neste steget er å installere alle SDK-pakkene fra Android Studio. Her bruker vi følgende pakker:

- SDK Platforms
  - Android 10 (Q) eller Android 11 (R)
- SDK Tools
  - Android SDK Build-Tools 31-rc3
  - NDK (Side-by-side)
  - Android SDK Platform-Tools
  - Google Play Services (ikke nødvendig).

Disse pakkene er installert for å kunne kjøre en Android Emulator, for å kunne bygge koden og lignende. Når disse er satt opp, så kan man starte en Android Emulator fra Android Studio AVD Manager ved å bruke en av plattformene som ble installert (10 eller 11).

2.1. SETTE OPP UTVIKLINGSMILJØET

Applikasjonen er opprettet med React Native CLI. Det krever at man har **Android Studio**, **Node** og **Java SE Development Kit (JDK)** [3]. For å sette opp miljøet har vi benyttet React Native sin offisielle dokumentasjon for å sette opp utviklingsmiljøet [2]. Her har vi fulgt guiden for `React Native CLI Quickstart` med `Target OS: Android.`

Guiden inkluderer hvordan man installerer og konfigurerer Node, Java SE Development Kit (JDK) og Android Studio. I tillegg til dette må man opprette en eller flere virtuelle Android-enheter i Android Studio som man kan benytte for å kjøre applikasjonen på under utviklingen. Man oppretter nye virtuelle Android-enheter via Android Studio, og man kan her velge mellom flere ulike modeller både for mobiler og nettbrett [4]. Det er valgfritt hvilken editor man ønsker å benytte, men vi har benyttet Visual Studio Code. Man kan også benytte Android Studio som editor.

[2] https://reactnative.dev/docs/environment-setup

2.1.1. NODE, JDK

React Native anbefaler å installere Node via *Chocolatey*, som er en pakkeadministrator for Windows [3]. Både Node og JDK kan installeres via Chocolatey.

INSTALLERING AV CHOCOLATEY PACKET MANAGER

Vi har her fulgt installasjonsguiden på Chocolatey sine sider [3].

1. Kjør powershell.exe som administrator

2. Sørg for å inspisere https://chocolatey.org/install.ps1 før installasjonen, da man bør verifisere alle script fra Internett man ikke er kjent med.

3. Sikre at *Get-ExecutionPolicy* ikke er satt til *Restricted*, da vil ikke scriptet kunne kjøres. De anbefaler å sette det til *AllSigned*.

   Kjør følgende kommando:

   ```
   >Get-ExecutionPolicy
   ```

   Hvis den returnerer `Restricted`, kjør følgende kommando:

   ```
   >Set-ExecutionPolicy AllSigned
   ```

4. Kjør deretter følgende kommando for å installere Chocolatey:

   ```
   >[System.Net.ServicePointManager]::SecurityProtocol =
   [System.Net.ServicePointManager]::SecurityProtocol -bor 3072;
   iex ((New-Object
   System.Net.WebClient).DownloadString('https://chocolatey.org/ins
   tall.ps1'))
   ```

INSTALLERING AV NODE OG JAVA SE DEVELOPMENT KIT (JDK)

1. Kjør powershell.exe som administrator

2. Kjør følgende kommando:

   ```
   >choco install -y nodejs.install openjdk8
   ```

Kommandoen installerer siste LTS-versjon av Node, som på nåværende tidspunkt er 14.15, og Java SE Development KIT versjon 8 (JDK 8).

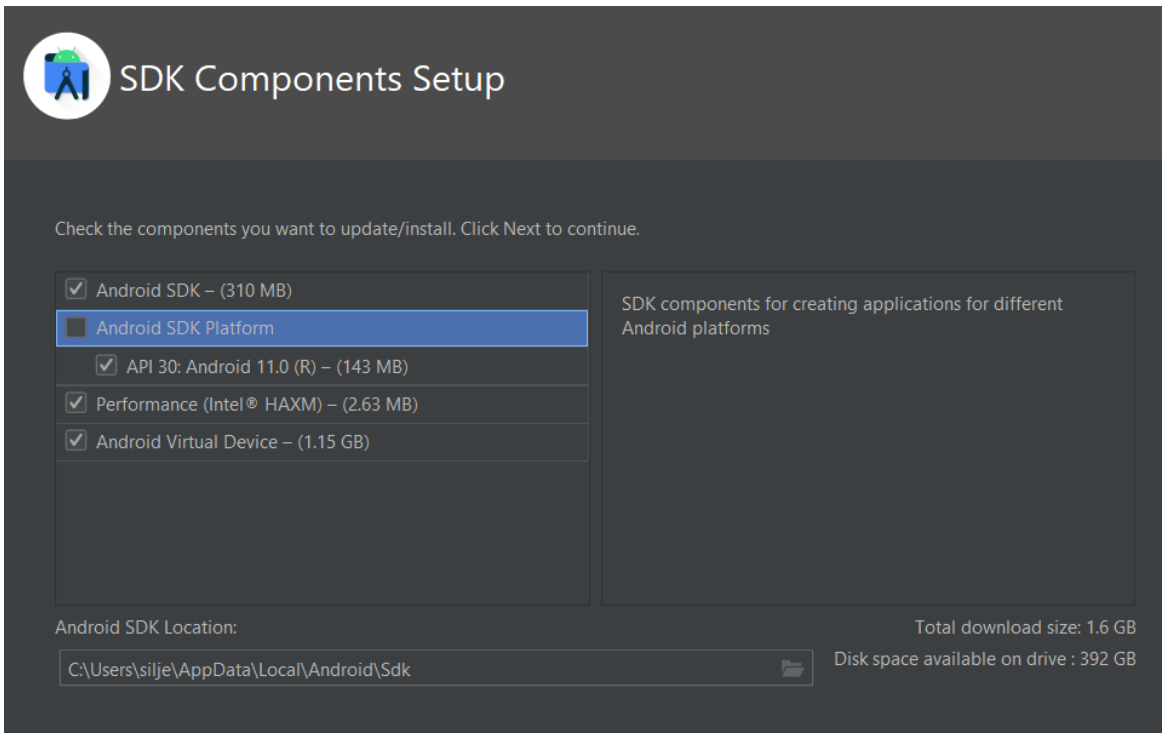
[3] https://chocolatey.org/install

2.1.2. ANDROID UTVIKLINGSMILJØ

Dette inkluderer installasjon av Android Studio, installasjon og konfigurering av Android SDK, og konfigurering av miljøvariabler.

INSTALLERING AV ANDROID STUDIO

Man laster ned Android Studio fra den offisielle siden til Android Developers [4]. I installasjonsveiviseren installerer man følgende komponenter:



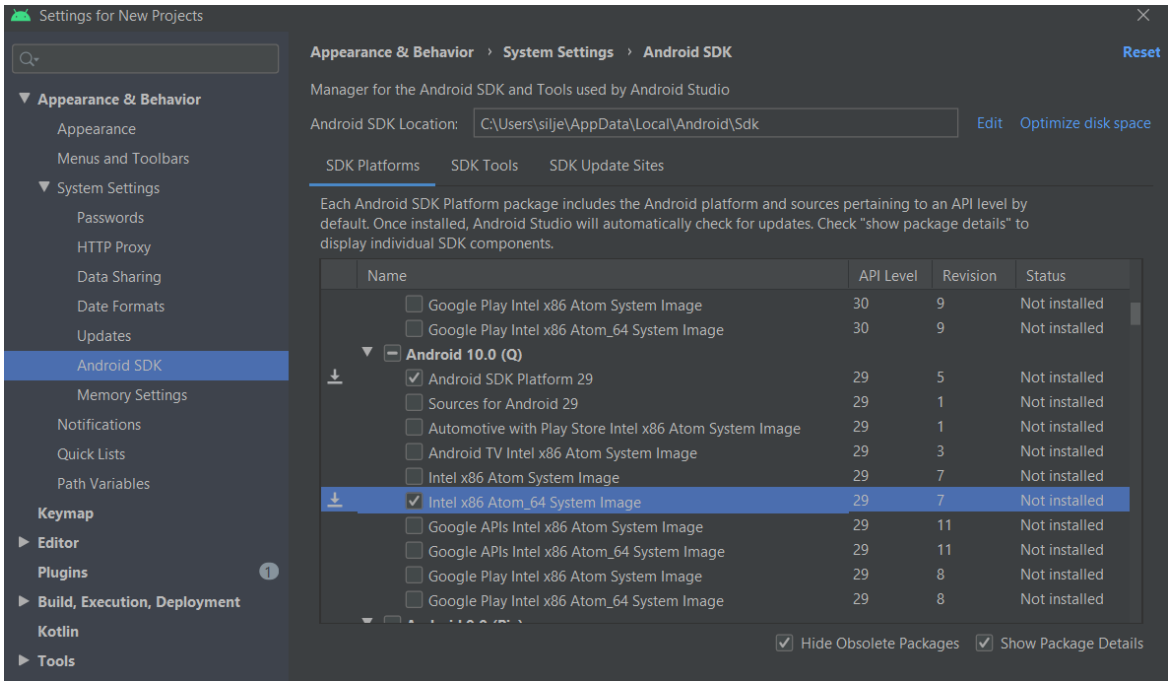
Performance (Intel ® HAXM) skal kun installeres hvis man ikke allerede benytter Hyper-V.

INSTALLERING AV PÅKREVDE ANDROID SDK

Android Studio installerer de nyeste Android SDK-er som standard. Men utvikling av en React Native applikasjon med native kode krever at man har Android 10 (Q) SDK [3]. Nyere versjon Android 11 fungerer også. Android SDK-er installeres via *SDK Manager* i Android Studio.

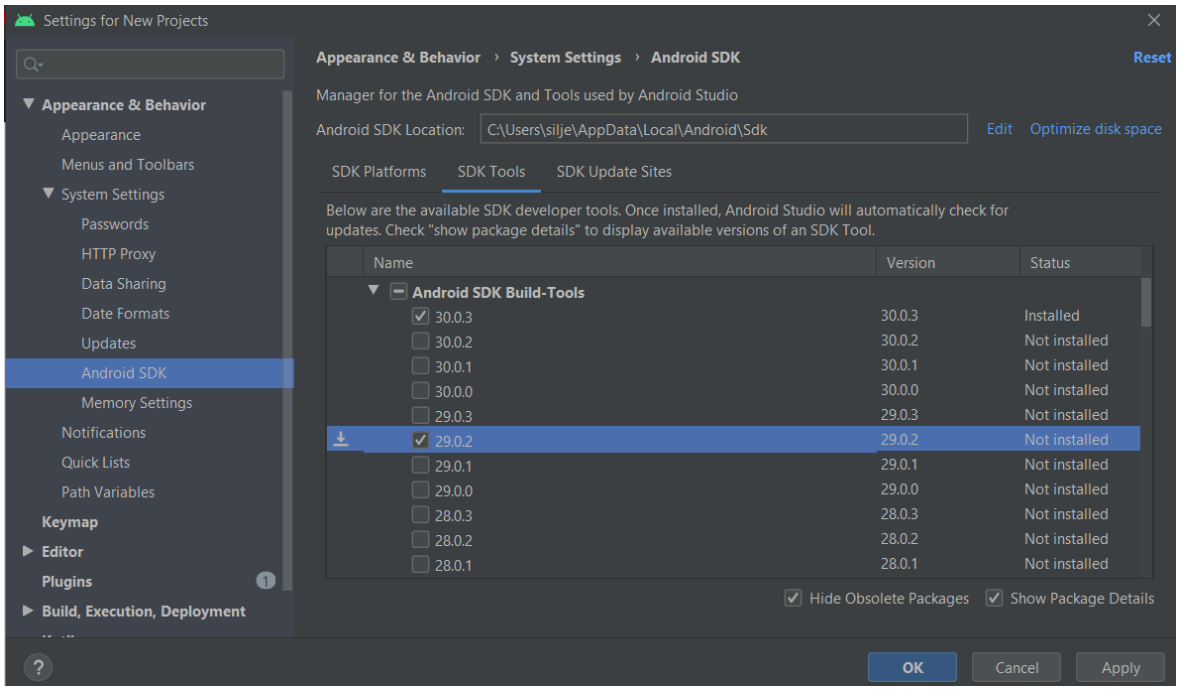
Følgende *SDK Platforms* må installeres:

[4] https://developer.android.com/studio



Dette er da **Android SDK Platform 29** og **Intel x86 Atom_64 System Image** under *Android 10*. Man kan også benytte *Android 11*, her huker man da av for de samme valgene som vist over.

Følgende *SDK Tools* må installeres:



Dette er da **29.0.2** (API-nivået tilhørende Android 10) under *Android SDK Build Tools.* Man kan også benytte Android 11, som har API-nivå 30.

KONFIGURERING AV MILJØVARIABLER

React Native-verktøyene krever at det settes opp noen miljøvariabler for å kunne bygge applikasjoner med native kode [3].

For å legge til ANDROID_HOME miljøvariabel:

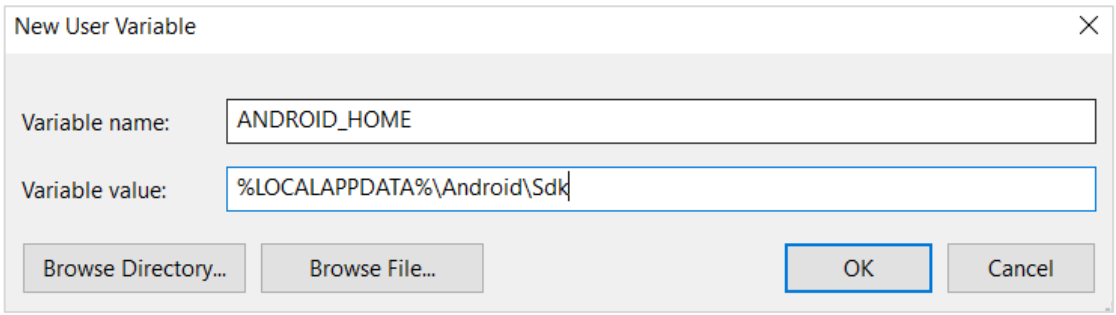
1. Åpne *Windows Control Panel*

2. Velg *User Accounts* og deretter klikk på *User Accounts* en gang til

3. Velg *Change my environment variables*

4. Velg *New...* for å lage en ny ANDROID_HOME brukervariabel som peker til stien til Android SDK.

   SDK-er blir som standard installert på lokasjonen *%LOCALAPPDATA%\Android\Sdk*. En kan finne lokasjonen til SDK-ene ved å gå til Android Studio under *Settings > Apperance & Behaviour > System Settings > Android SDK*.



5. Verifiser at variabelen har blitt lagt til ved å kjøre følgende powershell-kommando:

```
>Get-ChildItem -Path Env:\
```

For å legge til "platform-tools" til Path:

1. Åpne Windows Control Panel

2. Velg *User Accounts* og deretter klikk på *User Accounts* en gang til

3. Velg *Change my environment variables*

4. Velg **Path**-variabelen og *Edit*

5. Velg *New* og legg til stien til **platform-tools** til listen.
   Standard lokasjon for denne mappen er *%LOCALAPPDATA\Android\Sdk\platform-tools.*

2.1.3. REACT NATIVE KOMMANDOLINJEGRENSESNITT

React Native kommer med et innebygd kommandolinjegrensesnitt. Man får tilgang til grensesnittet ved hjelp av **npx**, som leveres med Node.js. Det anbefales å benytte npx, da man her slipper å installere og administrere en bestemt versjon av kommandolinjegrensesnittet globalt [3].

Med `>npx react-native <command>` vil den nåværende stabile versjonen av CLI bli lastet ned og utført når kommandoen kjøres [3].

2.2. KJØRE APPLIKASJONEN I DEBUG-MODUS

Under utviklingen er det svært nyttig å kjøre applikasjonen i en emulator, for å se hvordan endringer man gjør i koden påvirker applikasjonen. For å kjøre applikasjonen i utviklingsmiljøet:

- Sørg for at du har en Android emulator kjørende
- I en terminal, går til prosjektets rotmappe (se kapittel 0 om filstrukturen)
- Kjør følgende kommando, som skal starte opp et nytt terminal-vindu og også starte applikasjonen i emulatoren:

  ```
  >npx react-native run-android
  ```

- For å oppdatere den kjørende applikasjonen, rediger koden og deretter trykk **R+R** (R to ganger) i emulatoren for å oppdatere

2.3. ADMINISTRERING AV AVHENGIGHETER

For å kunne bruke tredjepartspakker kreves, som nevnt tidligere, **npm.** For å installere en ny pakke kjører man for eksempel `>npm install <package-name> --save` [5]. Dette vil gjøre det mulig for alle andre utviklere å få tilgang til pakken ved å bare kjøre en *npm install* kommando senere og pakken vil bli lagt inn som en avhengighet i filen *package.json* i utviklingsmappen.

Skal en pakke avinstalleres gjøres det på lignende vis: `>npm uninstall <package-name>` [5]. Det er igjen viktig at når andre brukere da skal jobbe fra sin respektive branch, at dette oppdateres ved å kjøre *npm install* slik at tredjepartspakkene blir oppdatert og det ikke oppstår konflikter i koden.

2.4. VERSJONSKONTROLL MED GIT

Gruppen benytter git via GitHub som versjonskontroll [6]. Dette vil si at når vi har gjort endringen i koden, så kjører vi en '*add-commit-push*' kommando for at koden skal bli tilgjengelig for andre utviklere.

`>git add .` - Forteller git at alle nye endringer skal bli gjort klart til å bli '*commited*'.

`>git commit –m «melding om hvilke endringer»` - Gir endringene en forklaring, her er det viktig at man enkelt og kort forklarer hvilke endringer som er gjort i koden siden sist.

`>git push` - Her vil endringene man har *'commited'* sendes til kodebasen som ligger i GitHub, men bare til den respektive branchen som er jobbet på. For at endringene skal bli tilgjengelige globalt, så må de også bli sendt til en *Branch* som heter ***development***.

Branchen ***development*** er hvor alle store endringer sendes og hvor gruppen arbeidet ut ifra. For å sende endringene til denne branchen går vi til GitHub og kjører en **PR (Pull-Request)** til ***development***. Alle brukere vil da få beskjed både i Slack og i GitHub om at nye endringer har blitt gjort offentlig. Nå kan alle brukere kjøre en enkel kommando fra sin Branch.

`>git pull origin development` - Vil hente inn alle endringer som er gjort i *development*. Dette gjøres systematisk før man begynner å kode eller endrer noe i sin egen Branch for å sikre at alle jobber fra samme kodebasen. På denne måten unngår man større konflikter som kan oppstå om man jobber med den samme koden.

Når det kommer til **PR**, skal dette i utgangspunktet godkjennes av en annen på gruppen som ikke sendte forespørselen. Dette er for å sjekke at kvaliteten på det som ønskes å migreres er bra nok og at det ikke har oppstått konflikter i kodebasen som gjør at migreringen skaper problemer. Vi har derimot kjørt en liten miks av at andre godkjenner og at vi bare godkjenner det selv, spesielt i de situasjonene der GitHub informerer om at det ikke oppstår konflikter ved å slå sammen koden.

Gruppen har også satt regler på hvordan branchene skal navngis. Vi går her ut ifra hvilken kode oppgavene har i Jira [7]. Et eksempel på en branch er *BADR-114_Sidemeny.* Her er *BADR* prosjektnavnet i Jira, *114* er koden Jira har gitt denne oppgaven og *Sidemeny* er hva denne oppgaven innebærer.

Når man er ferdig med en oppgave og den er blitt migrert med *development*, skal denne branchen slettes. På denne måten skiller vi ulike komponenter som opprettes, endres eller konfigureres fra hverandre og flere kan jobbe samtidig ut fra egne brancher på *development* branchen.

Helt til slutt når koden er god nok for en av prototypene, så skal dette sendes til branchen ***main***. Denne branchen skal bare inneholde produksjonskoden, altså kode som ikke inneholder feil, logging eller problemer og som vi kan med sikkerhet si at er trygg nok til å laste opp på **Google Play Store**. Det vil si at når vi arbeider og endrer på koden, skal det aldri sendes til *main* før dette er bestemt og planlagt av hele gruppen.

3. PUBLISERING PÅ GOOGLE PLAY STORE

Vi vil her gå gjennom hva som måtte gjøres for å klargjøre applikasjonen for publisering, dvs. hva som måtte gjøres for å gjøre at appen kunne legges ut på Google Play Store. Deretter går vi gjennom hvordan vi har benyttet Google Play Console, som er portalen man benytter for å legge ut applikasjoner. Til slutt går vi gjennom hvordan man oppdaterer appen og ruller ut nye versjoner av den på Google Play.

3.1. KLARGJØRING

For å kunne distribuere en applikasjon via Google Play Store, må den først signeres med en utgivelsesnøkkel som deretter må brukes til alle fremtidige oppdateringer [8]. Vi genererte en privat signert nøkkel med verktøyet *keytool*, som er inkludert i *Java SE Development Kit (JDK)*, med følgende kommando:

```
>keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-
key.keystore -alias illustrafikk-key -keyalg RSA -keysize 2048 -
validity 10000
```

Deretter lastet vi opp den genererte nøkkelen til mappen *android/app/* i prosjektets rotmappe. Videre la vi inn informasjon om nøkkelen, navn og valgt passord, i fila *android/app/gradle.properties* som inneholder variabler som benyttes når man bygger appen.

```
# Key and password
MYAPP_UPLOAD_STORE_FILE=my-upload-key.keystore
MYAPP_UPLOAD_KEY_ALIAS=illustrafikk-key
MYAPP_UPLOAD_STORE_PASSWORD=*********
MYAPP_UPLOAD_KEY_PASSWORD=********
```

Det siste som måtte gjøres før vi kunne bygge appen første gang var å legge inn signeringskonfigurasjonen i fila */android/app/build.gradle.* Denne fila definerer blant annet hvilke variabler fra *gradle.properties* som skal benyttes ved bygging av appen. Vi måtte her legge inn at ved bygging av appen skulle den signeres med den nøkkelen vi la inn.

```
signingConfigs {
        release {
            storeFile file("${MYAPP_UPLOAD_STORE_FILE}")
            storePassword "${MYAPP_UPLOAD_STORE_PASSWORD}"
            keyAlias "${MYAPP_UPLOAD_KEY_ALIAS}"
            keyPassword "${MYAPP_UPLOAD_KEY_PASSWORD}"
        }
    }
    buildTypes {
        release {
```

```
            signingConfig signingConfigs.release
            minifyEnabled enableProguardInReleaseBuilds
            proguardFiles getDefaultProguardFile("proguard-android.txt"),
               "proguard-rules.pro"
        }
    }
```

3.2. Generere AAB (Android App Bundle)

En AAB er en fil som inneholder all koden som trengs for å kjøre applikasjonen. Det er også Android sitt foretrukne format for apper som lastes opp til Google Play Store.

For å generere en AAB kjører vi følgende kommando i mappa *android/* i prosjektets rotmappe:

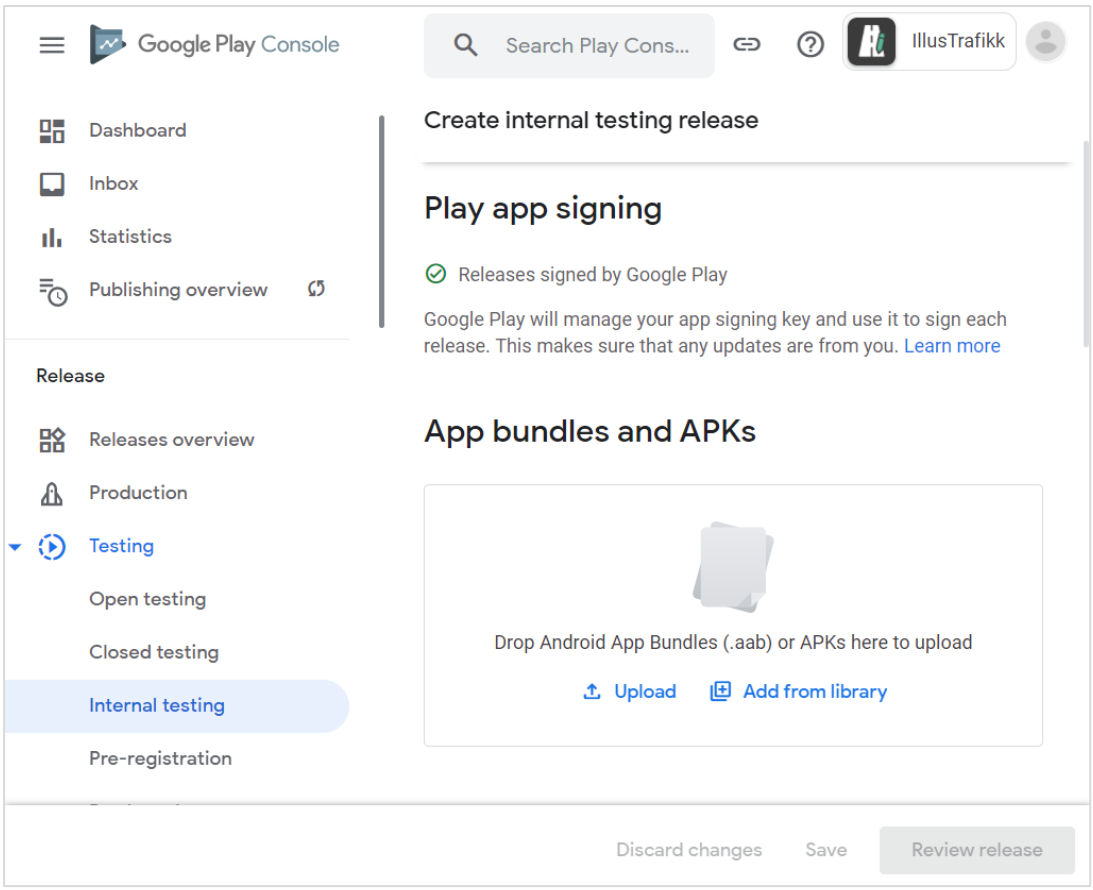
```
>gradlew bundleRelease
```

Den genererte AAB finnes under *android/app/build/outputs/bundle/release/app-release.aab*, og er klar til å lastes opp til Google Play.

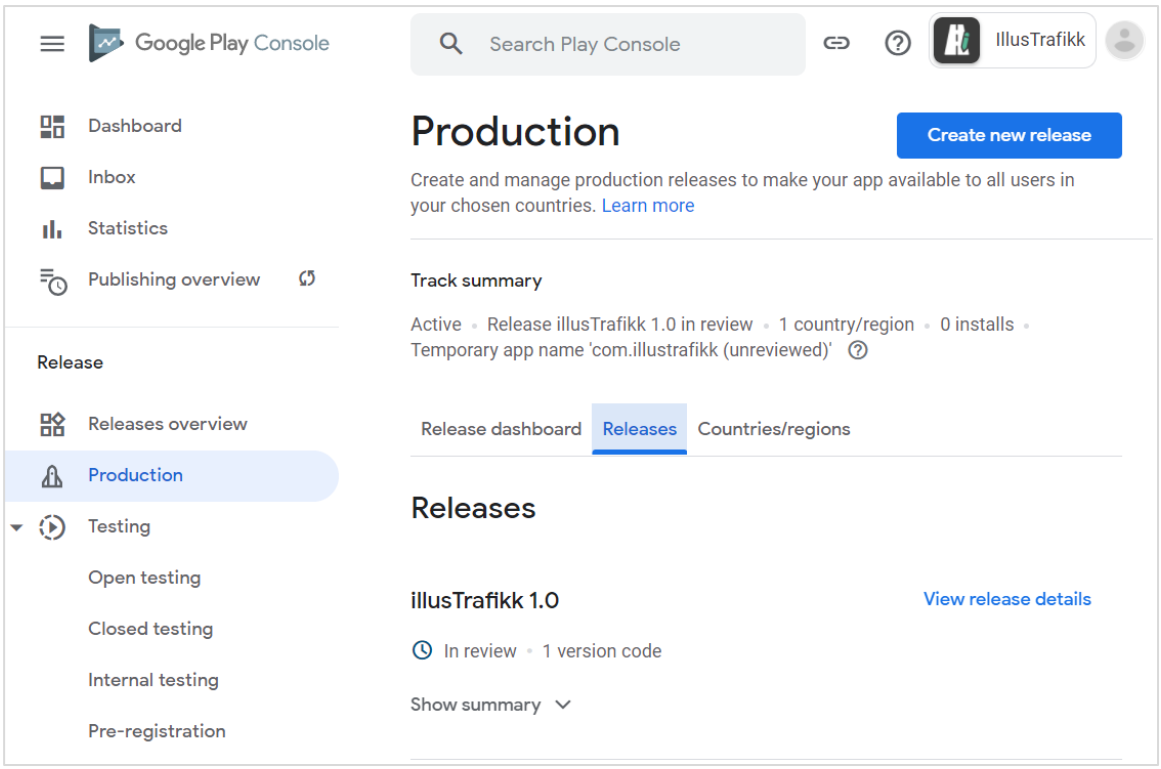
3.3. Google Play Console

Vi har opprettet en egen konto for dette prosjektet, for å få tilgang til Google Play Console. Dette er portalen man benytter for å legge ut applikasjoner i Google Play Store. Kontoen er opprettet med e-postadressen ***illustrafikk@gmail.com***. Portalen er veldig oversiktlig og selvforklarende, men vi vil likevel forklare kort hva vi har gjort her for å legge ut applikasjonen.

Omtrent midtveis i prosjektet la vi ut testversjoner av applikasjonen i Google Play, under *Internal Testing* som bildet under viser. Med en slik utgivelse legger man inn en liste med personer som skal få tilgang til applikasjonen, og de kan da laste ned appen via en lukket side på Google Play Store som kun personer som har tilgang kan se. På denne måten kan man få testet appen med flere personer, uten å måtte gjøre den tilgjengelig for alle. For å legge ut appen laster man helt enkelt opp AAB-en (se kapittel 0 Generere AAB), som bildet under viser.



For å legge ut applikasjonen i produksjon, dvs. gjøre den tilgjengelig for nedlastning via Google Play Store for alle, oppretter man en ny *release* under *Production*. Som bildet under viser, har vi har opprettet en *release* i produksjon. Under fanen *Release dashboard* kan man følge med på antall nedlastninger og avinstalleringer av appen, anmeldelser, og stabiliteten til appen. Sistnevnte går ut på at det hentes inn data om appen krasjer, hvor ofte dette skjer og hvor mange enheter det skjer på.



3.4. Legge ut nye versjoner

For å legge ut nye versjoner av appen, genererer man først en ny AAB (se kapittel 3.2 Generere AAB). Deretter oppretter man en ny *release* via Google Play Console under produksjon, hvis man ønsker å oppdatere appen for alle. Ønsker man kun å legge ut en ny versjon for testing, oppretter man en ny *release* under testing.

Det som er viktig å tenke på når man laster opp nye versjoner (ny AAB) i Google Play Console, er at hver AAB man laster opp må ha en unik versjonskode. For å gjøre utrulling av nye versjoner enklere i framtiden, har vi konfigurert slik at versjonskoden automatisk inkrementeres hver gang man kjører kommandoen for å bygge appen. Dette har vi gjort ved å definere versjonskoden i en separat fil, *version.properties*, som inkrementeres med følgende kode i fila *build.gradle* som kjøres hver gang man bygger appen:

```
def propFile = file('version.properties')
    def Properties versionProp = new Properties()
    if(propFile.exists())
        versionProp.load(new FileInputStream(propFile))
    def verCode = (versionProp['VERSION_CODE'] ?: "0").toInteger() + 1
```

Det eneste som dermed må gjøres for å legge ut en ny versjon i Google Play, er å generere en ny AAB og legge ut denne som en ny release.

4. OVERORDNET OM APPLIKASJONEN

Får vi går inn på detaljnivå om de ulike komponentene applikasjonen består av, vil vi først gi en oversikt over hvilke filer appen består av, hvordan de er strukturert og hvorfor. Vi forklarer også hvorfor vi har valgt å bruke funksjonsbaserte komponenter over klassebaserte. Deretter gir vi en innføring i hvordan vi har valgt å navnsette filer, komponenter, variabler og funksjoner. Til slutt gir vi en oversikt over hvilke avhengigheter applikasjonen og utviklingsmiljøet har.

4.1. FILSTRUKTUREN

Her går vi gjennom filstrukturen til applikasjonen, slik at det skal være enklere å finne frem til ulike komponenter i applikasjonen. Vi har valgt å dele inn applikasjonen i flere, mindre komponenter og deretter sette sammen disse i en hovedkomponent. På denne måten blir koden enklere å lese, og det vil være enklere å vedlikeholde koden.

```
trafikklearerapp/
├── .idea
├── TrafikkApp/
├── .gitignore
├── LICENSE
└── README.md
```

FILSTRUKTUR FOR GIT ROTMAPPE

git rotmappen er den som ligger i versjonskontrollsystemet git (GitHub). Denne inneholder blant annet README-filen i tillegg til selve mappen for applikasjonen – *TrafikkApp*.

```
TrafikkApp/
├─ __tests__/
├─ android/
├─ assets/
├─ components/
├─ docs/
├─ ios/
├─ node_modules/
├─ screens/
├─ styles/
├─ App.js
├─ app.json
├─ AppContext.js
├─ index.js
├─ jsdoc-config.json
├─ components/
└─ package.json
```

FILSTRUKTUR FOR APPLIKASJONENS ROTMAPPE

Rotmappen for applikasjonen, TrafikkApp, inneholder en del mapper og filer som blir opprettet automatisk når man oppretter et nytt React Native-prosjekt med react-native-cli. I tillegg har vi valgt å opprette noen egne mapper her; *components*, *screens* og *styles*. Vi har valgt å strukturere filene i prosjektet etter hvilken funksjon de har. Dette har vi gjort for å gjøre det enklere å finne fram i koden og for å øke skalerbarheten til prosjektet.

Mappene vi har benyttet aktivt, dvs. der vi har lagt inn grafikk-filer og .js-filer, er *assets*, *components*, *screens* og *styles*.

assets – Her ligger alle grafikk-filer som benyttes i applikasjonen. Dette inkluderer blant annet ikoner, illustrasjoner av veikryss og rundkjøringer, og trafikkskilt.

components – Denne mappen inneholder alle komponenter som benyttes i applikasjonen. Vi har som sagt valgt å strukturere filene etter hvilken funksjon de har i appen, så mappen *components* har igjen undermapper ut ifra hvilken funksjon komponentene har.

screens – Her ligger alle "sidene" i appen. Dette er sidene man navigerer til når man trykker på linkene på startskjermen eller i navigasjonsmenyen.

styles – Denne mappen inneholder alle stiler som benyttes for flere komponenter. Dette inkluderer farger, fontstørrelser, ikonstørrelser osv.

I tillegg benyttes fila *App.js* for å hente inn appens start-komponent som igjen henter inn de ulike sidene i appen. *AppContext.js* benyttes for å kunne lagre brukerdata, dvs. endringer brukeren gjør i applikasjonen som man ønsker skal være gjeldende også når man lukker appen og starter den igjen.

```
assets/
├── Elements/
├── images/
├── intersections/
├── roadSigns/
├── roundabouts/
├── sign_descriptions/
└── vei/
```

FILSTRUKTUR FOR UNDERMAPPEN *ASSETS*

Denne mappen er igjen delt opp i undermapper ut ifra hvilken funksjon bildet/illustrasjonen har.

Elements – Mappen inneholder bildene eller ikonene som benyttes for drabare elementer for tegning. Den inneholder også ikoner som benyttes for kartfunksjonen.

images – Her ligger alle generelle bilder i applikasjonen. Dette er grafikk som benyttes kun for stilsetting og som ikke har noen annen funksjon.

intersections – Inneholder alle illustrasjoner over veikryss, gruppert etter forkjørskryss, høyrekryss og lyskryss. Disse illustrasjonene benyttes som bakgrunnsbilde for tegning på veikryss-siden i applikasjonen.

roadSigns – Inneholder alle trafikkskilt, gruppert etter skiltgruppe.

roundabouts – Inneholder alle illustrasjoner over rundkjøringer, fordelt på om det er rundkjøring med 1 eller 2 felt. Disse illustrasjonene benyttes som bakgrunnsbilde for tegning på rundkjøring-siden i applikasjonen.

signDescriptions – Inneholder beskrivelse av alle de forskjellige trafikkskiltene, gruppert i JS-filer med JSON format etter type trafikkskilt.

vei – Inneholder alle illustrasjoner over "vei", som er fartsøknings- og reduksjonsfelt (E6), landevei, smal veg o.l. Disse illustrasjonene benyttes som bakgrunnsbilde for tegning på henholdsvis "fartsøknings- og reduksjonsfelt"-siden og landevei-siden i applikasjonen.

FILSTRUKTUR FOR UNDERMAPPEN *COMPONENTS*

```
components/
├── aboutComponents/
├── authorityPyramidComponents/
├── curriculumComponents/
├── draggableComponents/
├── helpers/
├── mapComponents/
├── navigationComponents/
├── reusableComponents/
├── roadSignComponents/
├── settingsComponents/
├── sketchComponents/
├── sketchHeaderComponents/
├── startScreenComponents/
```

Mappen er delt inn etter hvilken funksjon komponentene har. Til sammen inneholder mappen ganske mange ulike komponenter fordelt på ulike filer. Vi vil derfor her kun liste opp mappene og ikke filene som ligger i de ulike mappen.

aboutComponents/ inneholder alle komponenter som brukes på siden for AboutScreen (siden for "om appen").

authorityPyramidComponents/ inneholder alle komponenter som brukes på siden for AuthorityPyramidScreen (siden for myndighetspyramiden).

curriculumComponents/ inneholder alle komponentene som brukes kun på siden CurriculumObjectivesScreen (siden for læreplanmål).

draggableComponents/ inneholder alle komponentene som benyttes for å implementere drabare elementer på tegnesidene. Dette er bilder eller ikoner man kan sette inn og flytte rundt på, på bakgrunnsbildet på tegnesidene.

helpers/ inneholder hjelpefunksjoner som benyttes av ulike komponenter i appen. Dette er for eksempel funksjoner og komponenter for lagring av brukerdata, innhenting av tillatelser for posisjonsdata o.l.

mapComponents/ inneholder alle komponenter relatert til den integrerte Google Maps-funksjonen i appen, som benyttes på MapScreen (siden for kart) og MapSketchScreen (siden for å tegne på kartet).

navigationComponents/ inneholder komponenter for navigering mellom sidene. Dette inkluderer selve navigeringen, sidemenyen for navigering og definisjon av hvilke sider man kan navigere til.

```
styles/
├── buttons.js
├── colors.js
├── icons.js
├── index.js
├── themeVariables.js
└── typography.js
```

reusableComponents/ inneholder alle komponenter som er gjenbrukbare og benyttes av flere komponenter i appen. Dette er blant annet Header for alle sidene, og en bunnmeny som benyttes på flere sider.

roadSignComponents/ inneholder komponenter som benyttes på siden for trafikkskilt.

settingsComponents inneholder alle komponenter som benyttes på siden SettingsScreen (siden for innstillinger).

sketchComponents/ inneholder alle komponentene relatert til tegnefunksjonen i appen, som benyttes på alle tegnesidene. Dette er sidene for veikryss, rundkjøring, landevei, fartsøknings- og reduksjonsfelt, og siden for å tegne på kartet. Her ligger også hovedkomponenten for tegneområdet – SketchArea.

sketchHeaderComponents/ inneholder komponentene som benyttes for tegnemenyen på tegnesidene.

startScreenComponents/ inneholder alle komponentene som benyttes på siden for StartScreen (startsiden) i appen.

FILSTRUKTUR FOR UNDERMAPPEN *SCREENS*

Mappen inneholder alle sidene i applikasjonen. Det er disse komponentene som hentes inn og benyttes av navigasjons-komponentene.

```
screens/
├── AboutScreen.js
├── AuthorityPyramidScreen.js
├── CountryRoadScreen.js
├── CurriculumObjectivesScreen.js
├── HighwayScreen.js
├── index.js
├── IntersectionScreen.js
├── MapScreen.js
├── MapSketchScreen.js
├── RoadSignScreen.js
├── RoundaboutScreen.js
├── SettingsScreen.js
└── StartScreen.js
```

De ulike screens-komponentene henter inn ulike komponenter ut ifra hvilken funksjon siden har.

For eksempel henter sidene for veikryss (IntersectionScreen), rundkjøring (RoundaboutScreen), fartsøknings- og reduksjonsfelt (HighwayScreen), landevei (CountryRoadScreen) og kart (MapSketchScreen) inn komponenten SketchArea, da de alle er sider for tegning.

FILSTRUKTUR FOR UNDERMAPPEN *STYLES*

Mappen inneholder alle stiler som benyttes på tvers av komponentene i applikasjonen.

Variabler for de ulike stilene, for eksempel fontstørrelser, høyde og vidde på knapper, defineres i *themeVariables*. *themeVariables* benyttes så i stilsettingen av knapper, ikoner og tekst.

Dette har vi gjort for å enkelt kunne endre blant annet fontstørrelser, ikonstørrelser og knappestørrelser for hele applikasjonen ved å kun endre variablene i *themeVariables*.

Fargene som benyttes i applikasjonen defineres i fila *colors*.

4.2. Funksjonsbaserte komponenter

I React Native er det to hovedtyper komponenter applikasjonen kan bestå av. Dette er **funksjonsbaserte** komponenter og **klassebaserte** komponenter. Vi har benyttet funksjonsbaserte komponenter i applikasjonen.

Klassebaserte komponenter er JavaScript ES2015-klasser som utvides fra en basisklasse fra React, kalt *Component*. Dette gir klassen (*App* i dette eksempelet) tilgang til React livssyklus-metoder som render, i tillegg til state- og props-funksjonalitet fra parent-klassen [9].

```
class App extends Component {
    render () {
        return (
            <Text>Klassebasert komponent</Text>
        )
    }
}
```

Funksjonelle komponenter derimot administrerer ikke i utgangspunktet sine egne *states* og har ikke tilgang til livssyklus-metodene som tilbys av React Native. De er derfor enklere, da de faktisk er JavaScript-funksjoner.

```
const StartPage = () => {
    return (
        <h1>Funksjonsbasert komponent</h1>
    );
}
```

Vi benytter funksjonsbaserte komponenter fordi dette gir mindre kode, det gjør koden enklere å lese og forstå, og man får bedre ytelse [10]. I tillegg gjør React Hooks, som ble innført i React versjon 16.8, at man kan benytte blant annet *states* uten å måtte skrive en klasse [11]. Den offisielle

dokumentasjonen til React sier også at det er god praksis og anbefalt å benytte Hooks i nye komponenter man skriver [12].

En artikkel fant at ved å kun konvertere en klassebasert komponent til en funksjonsbasert komponent, økte ytelsen med 6 %. Med enda mer optimalisering kunne man øke ytelsen med hele 45 % i forhold til en klassebasert komponent [13].

Funksjonsbaserte komponenter gjør det også enklere å gjenbruke logikk, ved at man for eksempel kan lage en egendefinert Hook og gjenbruke denne i de komponentene man ønsker. I en klasse kan man ikke gjøre dette, her må man skrive om hele komponenten.

4.3. Navngiving av filer, komponenter, variabler og funksjoner

For å gjøre det mest mulig oversiktlig og enklest for de som skal lese koden, er det viktig å være konsistent på hvordan man navngir filer, komponenter, variabler og funksjoner. Felles for all navngiving, bortsett fra grafikk-filer, er at vi har benyttet engelsk. Dette har vi gjort fordi det er dette som er vanligst i programmering.

Vi benytter ulike metoder for navngivning ut ifra om det er en fil, en komponent, en variabel eller en funksjon. Vi skiller også mellom navn på grafikkfiler, komponent-filer og variabel-filer. Grunnen til at vi gjør dette er for å lettere kunne se hva ting gjør, kun ut ifra måten de er navngitt på.

Grunnen til at vi benytter norsk filnavn for grafikk-filer, er at mange av disse filene brukes av enkelte komponenter for å generere innhold dynamisk ut ifra navnet på filen.

De ulike metodene vi har benyttet er populære metoder for å kombinere flere ord i en streng [14]. Tabellen under viser en oversikt over hvilken metode de ulike elementene er navngitt etter.

TYPE	NAVNGIVINGSMETODE	EKSEMPEL
Mapper	camelCase	sketchHeader/
Filnavn komponenter	PascalCase	SketchArea.js
Filnavn variabler	camelCase	themeVariables.js
Filnavn grafikk	kebab-case	veikryss-hoyre-X.png
Komponenter	PascalCase (likt filnavnet)	SketchArea
Variabler	camelCase	roadDesign
Theme-variabler	SNAKE_CASE	FONT_SIZE_LARGE
Funksjoner	camelCase	radioButtonChange

Tabell 24 – Navngiving av filer, komponenter, variabler og funksjoner

Generelt har vi hatt fokus på å sette forståelige og beskrivende navn på filer, komponenter, variabler osv. Ved å gjøre det, reduserer man behovet for å kommentere hva koden gjør.

4.4. AVHENGIGHETER

Oversikt over alle avhengigheter for applikasjonen og hva de brukes til. Dette er viktig å vite når det utgis nye versjoner av React og React Native, og man skal oppdatere applikasjonen til å benytte disse. Da bør man på forhånd sjekke at avhengighetene også støtter ny versjon.

Nåværende versjoner av React Native og React (dette er også de siste versjonene som er gitt ut på nåværende tidspunkt):

```
"react": "16.13.1",

"react-native": "0.63.4",
```

Alle avhengigheter støtter nåværende versjoner av React Native og React. Hvilke versjoner som benyttes for applikasjonen kan finnes i fila *package.json* som ligger i prosjektets rotmappe.

NAVN	BESKRIVELSE	BRUKES AV KOMPONENT/PAKKE
@react-native-async-storage/async-storage	For lokal caching av data.	SettingsView
@react-native-community/masked-view	Er en avhengighet til *@react-navigation.*	@react-navigation/native
@react-native-community/checkbox	Checkbox for å huke av for valg.	AlertModal
@react-navigation/drawer	Drawer eller "skuff" som man drar frem fra venstre side av skjermen. For å navigere mellom sider (screens).	Navigator
@react-navigation/native	Navigasjon mellom ulike sider.	@react-navigation/drawer
@terrylinla/react-native-sketch-canvas	Komponent for å tegne ved berøring.	SketchArea
react-native-easy-gestures	For å dra, skalere og rotere en komponent.	Draggable

react-native-gesture-handler	API for å utnytte plattformens berørings- og bevegelsessystem, for å bygge best mulig berøringsbaserte opplevelser i React Native.	PopoutItems @react-navigation/native
react-native-geolocation-service	API for å ta i bruk enhetenes lokasjon på kartsiden.	MapArea
react-native-intent-launcher	Pakke for å kunne kalle den native funksjonen «startActivity» i React Native for å kunne gjøre noe som bare kan gjøres med android native kode. I vårt tilfelle innstillingene til vår applikasjon på selve enheten.	AlertPermissionModal
react-native-maps	Komponenter for å ta i bruk Google Maps for React Native.	MapScreen MapCallout MapArea
react-native-popup-menu	Utvidbar popup-menykomponent for React Native.	ColorButton SketchColorMenu DeleteButtonPopover
react-native-reanimated	*React Native Animated*-bibliotek reimplementert. Gir mye større fleksibilitet enn *React Native Animated*, spesielt når det kommer til bevegelsesbaserte interaksjoner.	@react-navigation/native
react-native-responsive-component	Tilbyr en måte for å effektivt rendre komponenter basert på enhetens skjermstørrelse og/eller orientering.	themeVariables
react-native-safe-area-context	Tilbyr en fleksibel mate å håndtere "safe area" på. Er en avhengighet til *@react-navigation.*	MainView @react-navigation/native
react-native-screens	For navigering mellom sider. Er en avhengighet til *@react-navigation.*	@react-navigation/native

react-native-vector-icons	For å vise ikoner i applikasjonen. Vi benytter ikoner fra biblioteket *Fontawesome5*.	Alle komponenter som benytter ikoner.
react-navigation	For navigering mellom sider. Er en avhengighet til *@react-navigation.*	@react-navigation/native
react-native-splash-screen	For å vise og skjule "splash screen". Dette er en skjerm som kommer opp før appen har lastet inn.	App

TABELL 25 - AVHENGIGHETER TIL APPLIKASJONEN

DEVDEPENDENCIES

Dette er pakker som brukes under utviklingen av applikasjonen, og for å bygge applikasjonen. Disse pakkene er altså kun nødvendige under selve utviklingen av applikasjonen.

NAVN	BRUKSOMRÅDE
@babel/core	Babel er en JavaScript-kompilator. Den benyttes hovedsakelig til å konvertere ECMAScript 2015+-kode til en bakoverkompatibel versjon av JavaScript, i nåværende og eldre miljøer.
@babel/runtime	Et bibliotek som inneholder modulære runtime-hjelpere for babel, og en versjon av regenerator-runtime.
eslint	Verktøy for å rapportere mønstre i ECMAScript/JavaScript-kode. Eslint ser både etter problematiske mønstre (fanger opp mulige feil) og stilsjekking.
@react-native-community/eslint-config	Brukes for å konfigurere Eslint.
babel-jest	For å lage, kjøre og strukturere tester med babel.
jetifier	Verktøy som automatisk migrerer avhengighetene til AndroidX ved runtime.
metro-react-native-babel-preset	Forhåndsinnstillinger for Babel for React Native-applikasjoner. React Native benytter denne som standard ved transformering av appens kildekode.
prettier	For formatering av kode. Med prettier får man en konsistent stil på koden, ved at koden blir formatert ut ifra regler som er satt for maksimal linjelengde, indentering o.l.
react-test-renderer	Pakken som gir en React-renderer som kan brukes til å gjengi React-komponenter til rene JavaScript-objekter, uten å avhenge av DOM eller et eget mobilmiljø.
jsdoc	Bibliotek for å generere dokumentasjon av kode, ut ifra kommentarer i koden. Man får da generert HTML-filer over dokumentasjonen, med strukturert oversikt over alle komponenter, metoder o.l.
better-docs	Tilleggsbibliotek for bedre jsdoc-dokumentasjon, med vekt på stilen på html-siden som blir generert.

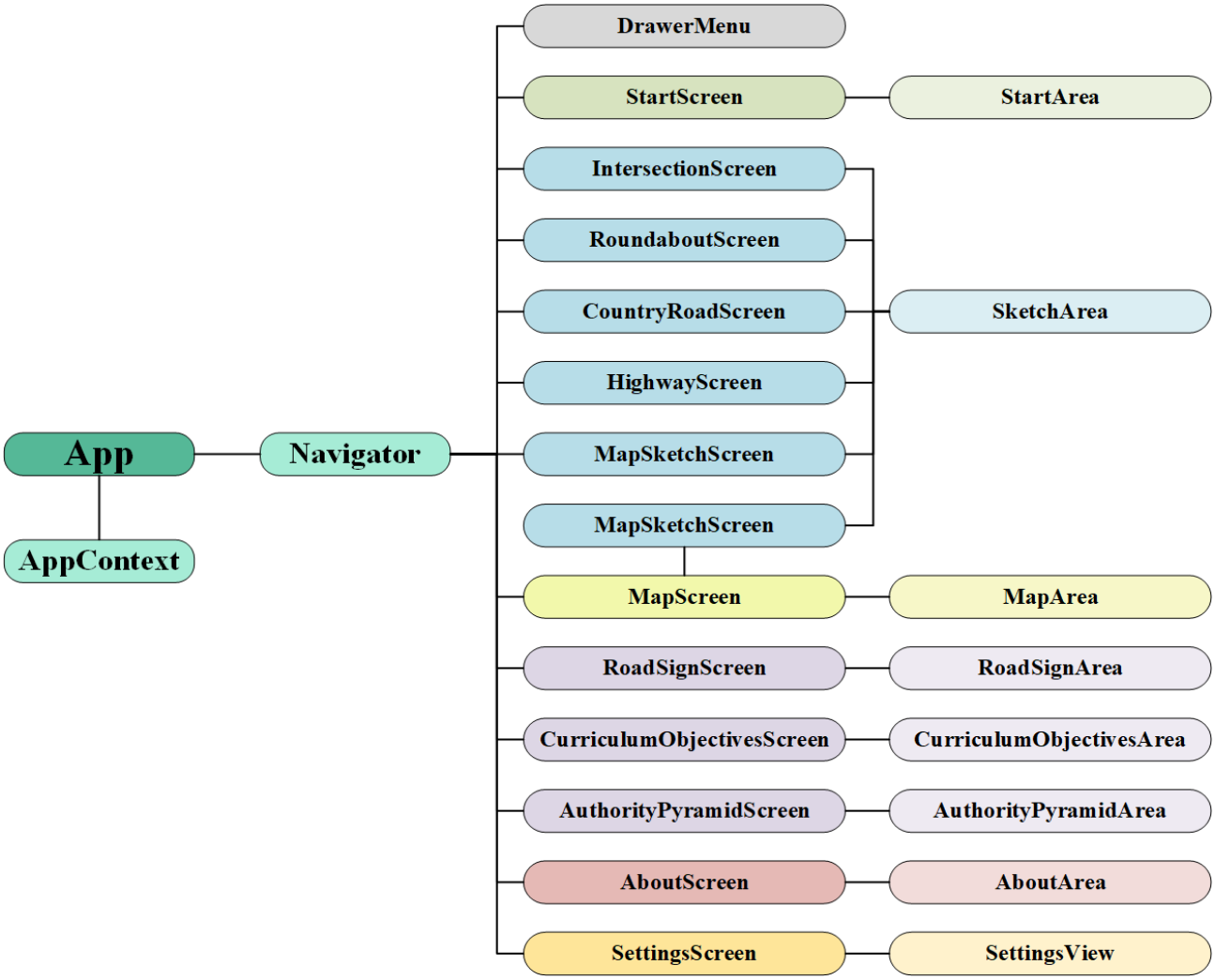
TABELL 26 - AVHENGIGHETER TIL UTVIKLINGSMILJØET

5. KOMPONENTER

En overordnet innføring i de ulike komponentene applikasjonen består av, og sammenhengen mellom de ulike komponentene. Måten vi har skrevet koden på, er at vi har splittet opp koden i flere ulike komponenter. På denne måten vil det være enklere å gjenbruke kode og det vil være enklere å vedlikeholde koden. Mer detaljert om de ulike komponentene, hvilke parametere, funksjoner og metoder de benytter, kan ses i JSDoc-dokumentasjonen [5].

5.1. OVERORDNEDE KOMPONENTER

Under er et diagram som forklarer relasjonen mellom de forskjellige komponentene som utgjør applikasjonen.



FIGUR 4 - APPENS OVERORDNEDE KOMPONENTER

Starter man fra venstre i diagrammet og går mot høyre har man App, dette er selve applikasjonen og alle komponentene samles her. Fra den har man to komponenter: AppContext og Navigator, hvor

[5] https://olivernilssen.github.io/trafikklearerapp/

AppContext gjør det mulig å dele variabler på et globalt nivå, og Navigator tar seg av rutingen og navigasjonen mellom de forskjellige sidene i applikasjonen.

Deretter kommer de forskjellige sidene i applikasjonen, disse er fargekodet etter hvilken funksjon de har. For eksempel så har alle illustrasjonssidene en blåfarge for å vise at disse har en lik funksjon, det samme gjelder sidene relatert til teori som har fått en lilla farge. Sidene vil bli forklart i dette kapittelet.

Helt til høyre i diagrammet er underkomponentene, disse gås gjennom i neste kapittel (0 Underkomponenter).

5.1.1. DRAWERMENU

DrawerMenu er tilgjengelig på alle sidene via et hamburgerikon i headeren. Når man trykker på ikonet vil en meny komme inn fra venstre side, og derfra kan man navigere til alle de andre sidene i applikasjonen. Sidene er del opp i kategorier etter hvilke funksjoner de har for en mer ryddig og oversiktlig navigering.

5.1.2. STARTSCREEN

StartScreen er den første siden man kommer til når man starter applikasjonen. Fra denne siden har man tilgang til alle de andre sidene i appen via linker på skjermen. Man har også tilgang til sidemenyen via et hamburgerikon i headeren. Det er også et tannhjulikon til høyre i headeren, som tar en direkte til innstillinger-siden.

5.1.3. ILLUSTRASJONSSIDER

Alle illustrasjonssider benytter seg av komponenten SketchArea. SketchArea består av alle komponentene som utgjør illustrasjonsfunksjonen i applikasjonen. Sidene for veikryss og rundkjøringer har en ekstra funksjon i menyen som gjør det mulig å velge om man vil ha gangfelt, sykkelfelt eller ingen av delene. Standarden er satt til ingen av delene.

INTERSECTIONSCREEN

Side for illustrering av trafikksituasjoner relatert til veikryss. I en bunnmeny har man muligheten til å velge mellom utformingen av krysset, altså om det er X-, T- eller Y-kryss, og om det er høyrekryss, forkjørskryss eller lyskryss.

ROUNDABOUTSCREEN

Side for illustrering av trafikksituasjoner relatert til rundkjøringer. I bunnmenyen har man mulighet til å velge mellom to forskjellige utforminger, altså om rundkjøringen er et-felts eller to-felts.

COUNTRYROADSCREEN

Side for illustrering av trafikksituasjoner relatert til landevei og man vil kunne velge mellom to forskjellige utforminger i bunnmenyen, en for landevei og en for smal veg.

HIGHWAYSCREEN

Side for illustrering av trafikksituasjoner relatert til hovedveier og man vil kunne velge mellom to forskjellige utforminger, en for fartsøkningsfelt og en for reduksjonsfelt.

MAPSKETCHSCREEN

Side for illustrering av trafikksituasjoner ved bruk av skjermdump fra Google Maps. Denne siden er kun tilgjengelig gjennom kartsiden, og har ingen bunnmeny.

5.1.4. MAPSCREEN

I MapScreen har man muligheten til å benytte seg av Google Maps for å kunne navigere og ta en skjermdump av et utsnitt av kartet. Skjermdumpen kan man deretter benytte til å illustrere på, velger man dette blir man sendt til *MapSketchScreen* som er en av illustrasjonssidene.

5.1.5. ROADSIGNSCREEN

Siden viser en sveip-bar liste med bilder av trafikkskilt. Skiltene er gruppert etter skiltgruppe, og det er i alt 9 forskjellige skiltgrupper man kan velge mellom. Man kan skifte mellom skilttypene ved å klikke på knappene for de forskjellige skilttypene i en bunnmeny. Hvert bilde i listen er en knapp, og når man trykker på knappen vil en modal med en større versjon av skiltet dukke opp. Trykker man på bildet igjen, eller på info-knappen, vil navn og beskrivelse av skiltet dukke opp. Man kan lukke modalen ved å trykke på X-knappen, eller ved å trykke utenfor modalen.

5.1.6. CURRICULUMOBJECTIVESSCREEN

Siden inneholder læreplanmål for Klasse B og for Klasse B kode 96 og BE, gjennom bunnmenyen vil man kunne først velge hvilke av de to klassene man ønsker å se og deretter kan man velge om man vil se hovedmål, eller målene til de forskjellige trinnene. Målene vises så i en sveip-bar liste med kort, hvor hvert kort inneholder et mål.

5.1.7. AUTHORITYPYRAMIDSCREEN

Viser et stort bilde av myndighetspyramiden.

5.1.8. ABOUTAPPSCREEN

Viser informasjon om applikasjonen og bakgrunnen for at den ble laget, samt navnet på utviklerne og kontaktinformasjon.

5.1.9. SETTINGSSCREEN

Lar brukeren endre diverse innstillinger i applikasjonen, som så blir gjeldende.

- *Viskelærstørrelse*: Lar brukeren velge hvilken størrelse det er på viskelæret, kan velge mellom 50 til 100. Inkrementerer med 10.

- *Slette alt ved illustrasjonsbytte*: Lar brukeren bestemme om man ønsker at alt av illustrasjon skal slettes eller ei når man skifter mellom de forskjellige utformingene på illustrasjonssidene.

- *Innledende farge på penn*: Lar brukeren bestemme hvilken farge pennen starter med, standard er rød.

- *Innledende farge på drabare elementer*: Lar brukeren bestemme innledende farge på de drabare elementene, standard er svart.

- *Drabare element-velger*: Lar brukeren selv bestemme hvilke drabare elementer som skal vises i menyen for drabare elementer. Kan max velge 15.

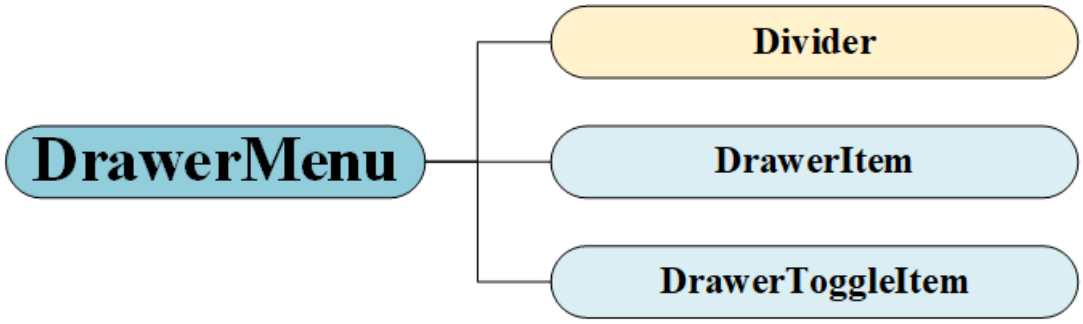
5.2. UNDERKOMPONENTER

Area-komponentene brukes for å samle sammen og strukturere de forskjellige underkomponentene som blir brukt. Area-komponentene blir så importert og brukt på de forskjellige sidene. For en mer detaljert beskrivelse av de ulike komponentene, hvilke parametere, funksjoner og metoder de benytter, se JSDoc-dokumentasjonen [6].

Komponentene i diagrammene under har fargekode som følge av type komponent; blå for hovedkomponent, lysegul for gjenbrukbare komponenter, lyseblå for underkomponenter som er laget kun for hovedkomponenten, og til slutt grå for komponenter som inneholder data.

5.2.1. DRAWERMENU

Ettersom DrawerMenu ikke er et eget screen har det ikke en egen area-komponent. DrawerMenu inneholder selv alle komponentene som utgjør menyen.

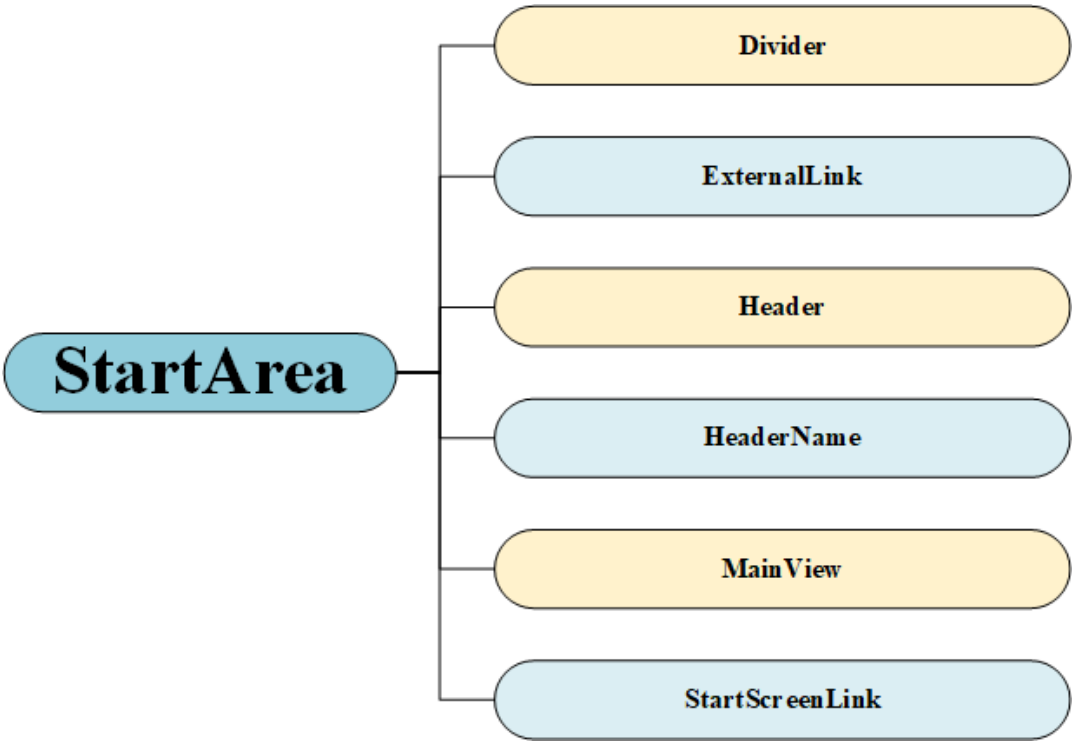


FIGUR 5 - KOMPONENTER SOM BENYTTES I DRAWERMENU

[6] https://olivernilssen.github.io/trafikklearerapp/

5.2.2. STARTAREA

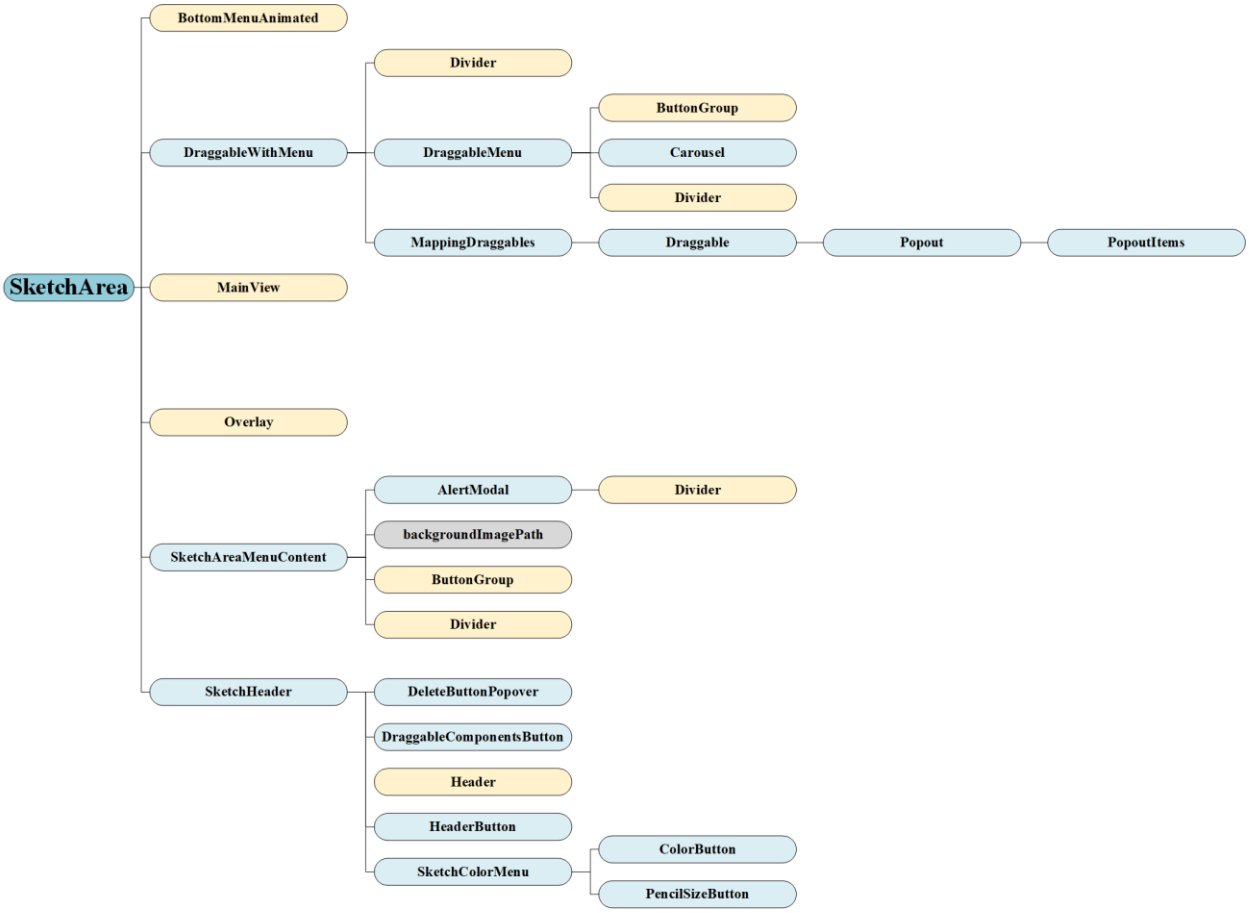
Inneholder alle komponentene som benyttes i StartScreen.



FIGUR 6 - KOMPONENTER SOM BENYTTES I STARTAREA

5.2.3. SKETCHAREA

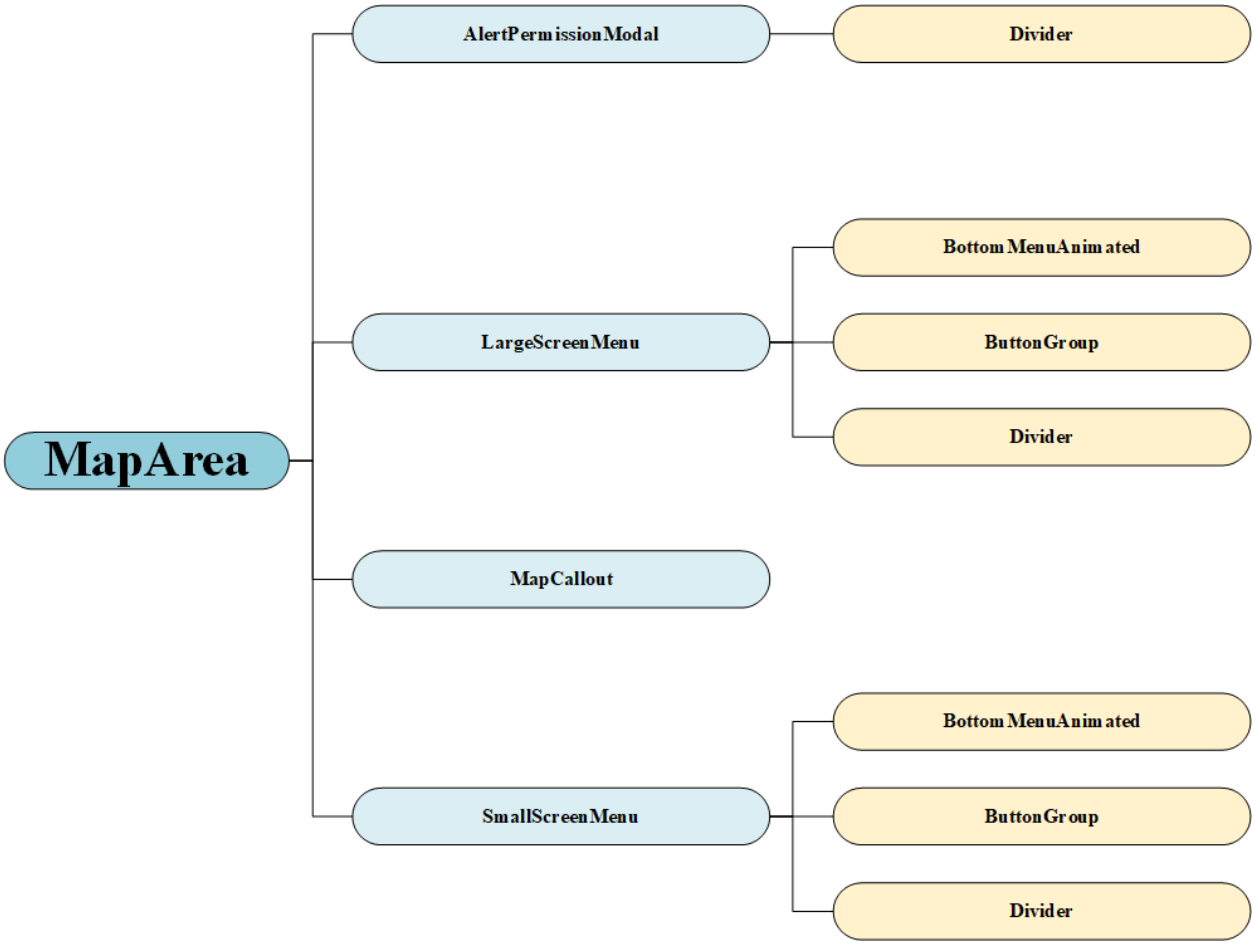
Dette er en veldig stor og omfattende komponent og inneholder alle komponentene som benyttes på sidene relatert til illustrering av trafikksituasjoner. Disse er ***IntersectionScreen***, ***RoundaboutScreen***, ***CountryRoadScreen***, ***HighwayScreen*** og ***MapSketchScreen***.



FIGUR 7 - KOMPONENTER SOM BENYTTES I SKETCHAREA

5.2.4. MAPAREA

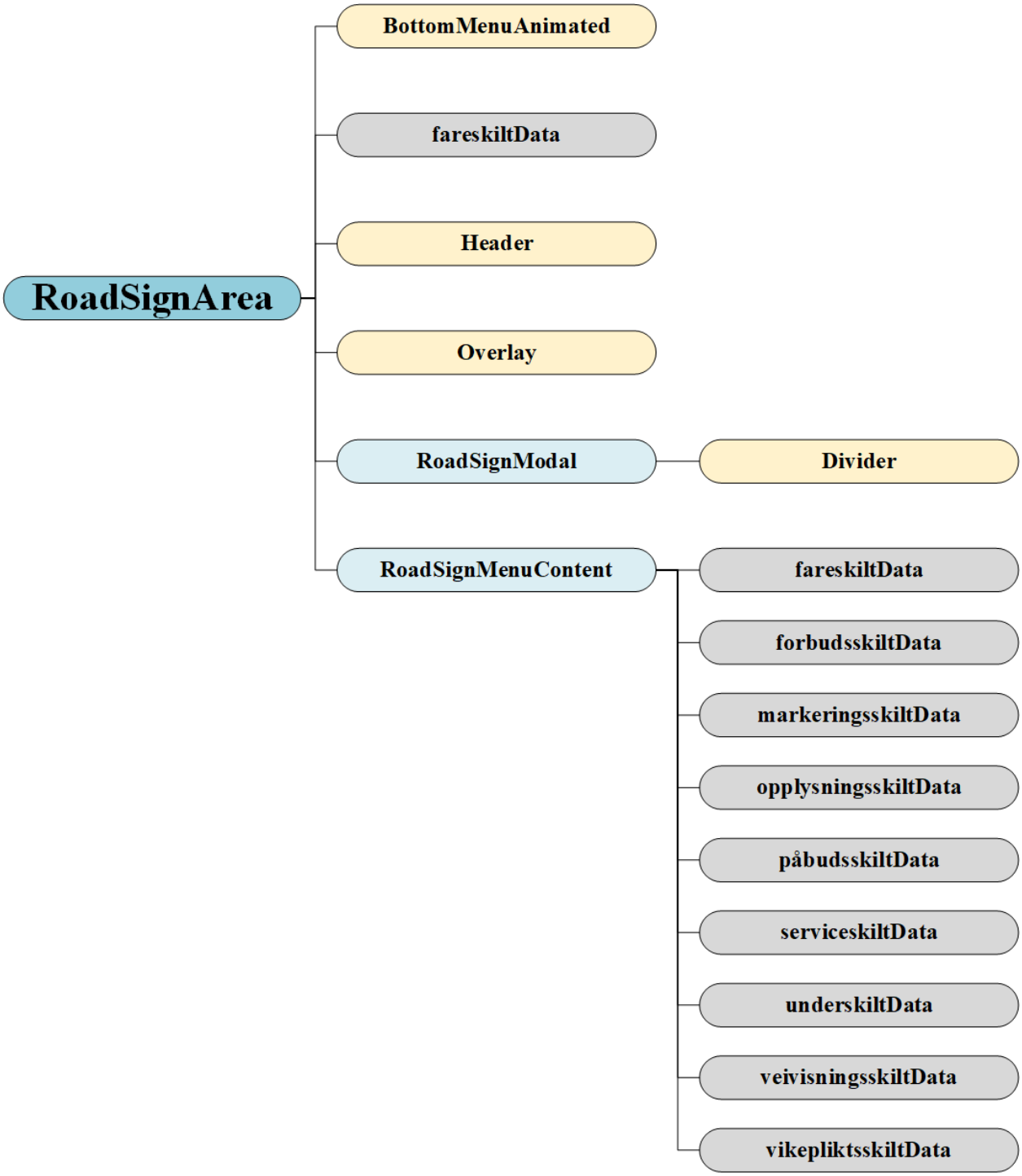
Inneholder alle komponentene som benyttes i MapScreen



FIGUR 8 - KOMPONENTER SOM BENYTTES I MAPAREA

5.2.5. ROADSIGNAREA

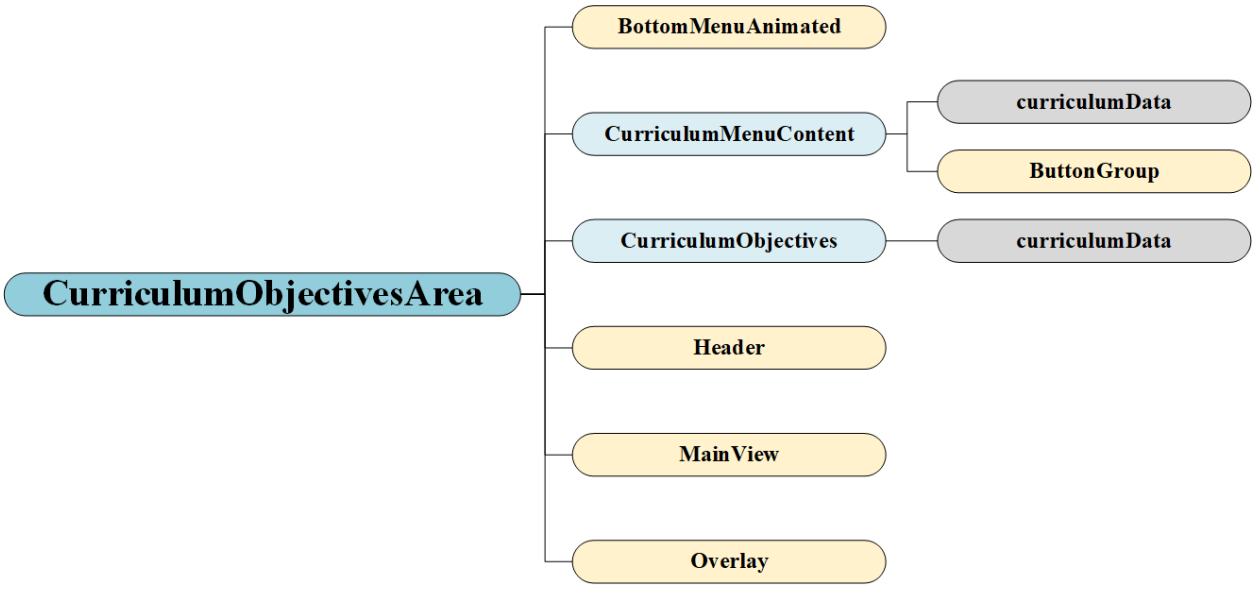
Inneholder alle komponentene som benyttes i RoadSignScreen.



FIGUR 9 - KOMPONENTER SOM BENYTTES I ROADSIGNAREA

5.2.6. CURRICULUMOBJECTIVESAREA

Inneholder alle komponentene som benyttes i CurriculumObjectivesScreen.



FIGUR 10 - KOMPONENTER SOM BENYTTES I CURRICULUMOBJECTIVESAREA

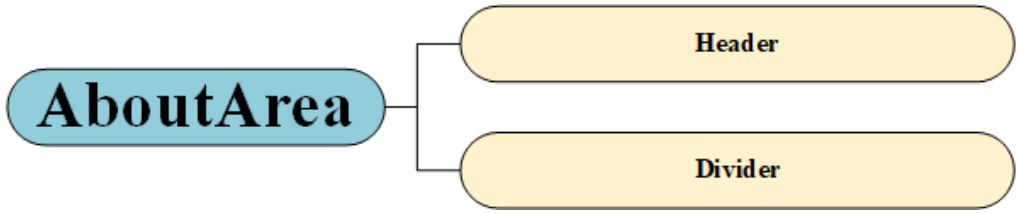
5.2.7. AUTHORITYPYRAMIDAREA

Inneholder alle komponentene som benyttes i AuthorityPyramidScreen.

FIGUR 11 - KOMPONENTER SOM BENYTTES I AUTHORITYPYRAMIDAREA

5.2.8. ABOUTAREA

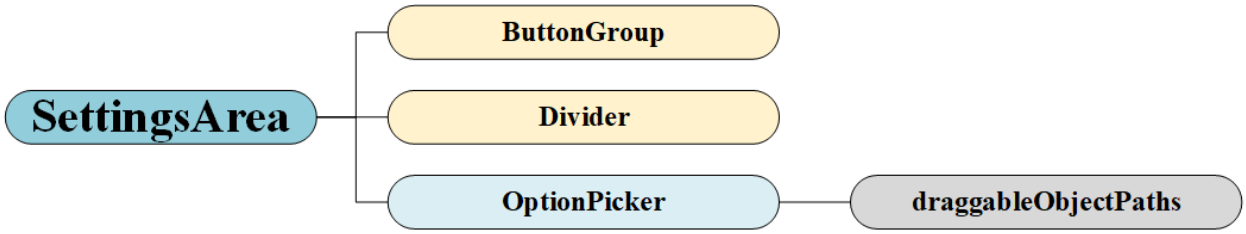
Inneholder alle komponentene som benyttes i AboutScreen.



FIGUR 12 - KOMPONENTER SOM BENYTTES I ABOUTAREA

5.2.9. SETTINGSAREA

Inneholder alle komponentene som benyttes i SettingsScreen.

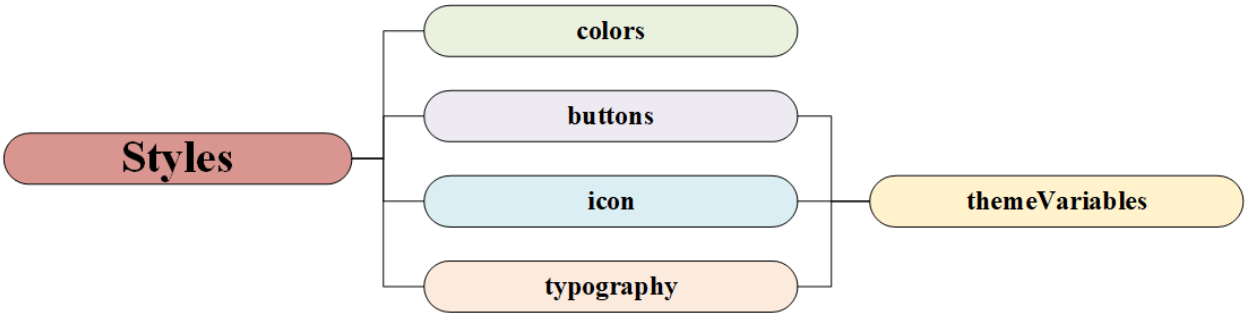


FIGUR 13 - KOMPONENTER SOM BENYTTES I SETTINGSVIEW

6. STILSETTING AV APPLIKASJONEN

De aller fleste komponentene (de som benytter spesifikk stilsetting og/eller farger) henter farger og spesifikk stilsetting (for eksempel for fontstørrelse og størrelse på knapper) fra predefinerte stil-komponenter som ligger under mappen *styles/* i prosjektets rotmappe (se kapittel 0 om filstrukturen). Hovedtankene bak en slik struktur [15]:

1. Ved å plassere Styles i rotmappen er den enkel å finne og man slipper å bruke unødvendig lange relative filstier.
2. Ved å først definere enkle variabler for stiler, kan man igjen bruke disse til å bygge mer kompliserte. Dette vil gjøre det mer lesbart og forståelig.
3. Ved å lage en index-fil hvor man importerer og eksporterer de ulike stil-komponentene, gjør man det mye enklere å importere stilene man trenger og man får en penere syntaks.
4. Ved å definere spesifikke stilsettinger i selve komponentene, har man bedre kontroll på ulike stilvalg man har gjort i de ulike komponentene. Dette sørger også for at man ikke gjør endringer i stilvalget for en komponent når man endrer noe i en annen.



FIGUR 14 - KOMPONENTER BRUKT TIL STILSETTING

6.1. COLORS

Inneholder alle fargene som benyttes i applikasjonen. Disse blir definert med navn, og deretter blir de linket opp mot forskjellige variabler som så eksporteres til de forskjellige komponentene i applikasjonen. Variablene har beskrivende navn relatert til hvor de skal benyttes, for eksempel *startScreenBg* som benyttes som bakgrunnsfarge på *StartScreen.*

6.2. BUTTONS

Henter inn variabler fra *themeVariables* og bruker disse for å definere størrelse og form på knapper. Disse blir så eksportert slik at de kan benyttes for å stilsette knapper i applikasjonen.

6.3. ICONS

Henter inn variabler fra *themeVariables* og bruker disse til å definere størrelsene på ikonene. Deretter blir de eksportert.

6.4. TYPOGRAPHY

Henter inn fontstørrelser fra *themeVariables* og bruker disse for å definere de forskjellige tekststilene som benyttes i applikasjonen.

6.5. THEMEVARIABLES

Inneholder variabler for font-, ikon og knappestørrelser som så benyttes i de andre stilkomponentene.

7. ILLUSTRASJONER OG ØVRIG GRAFIKK

Når det kommer til illustrasjoner så gjelder det illustreringer som er blitt produsert av prosjektgruppen. Disse filene er lagret i et spesifikt format som må formateres til å kunne brukes som 'bilder'. Øvrig grafikk er derimot alle bilder, tegninger og ikoner som er blitt brukt i applikasjonen. Disse har vi enten laget selv, hentet via tredjepartspakker eller lastet ned med CC0 (Creative Commons) som betyr at vi har lov og rettigheter til å bruke og manipulere bildene gratis. Dette kapittelet tar for seg hvilke verktøy gruppen har brukt under prosjektet og filstrukturen som er anvendt for å hente inn grafikk.

7.1. GRAFIKKVERKTØY

Illustrasjonene som er brukt i applikasjonen er laget ved hjelp av inkScape [16]. InkScape er et vektorgrafikkverktøy for å lage vektorbilder, altså skalerbare bilder. Her har alle trafikkbildene blitt produsert i SVG-format, for så å bli eksportert til PNG-format. PNG er brukt da det sparer plass på enhetene og tillater gjennomsiktig bakgrunn. Gruppen har prøvd å forholde seg til bilder, ikoner og illustrasjoner som tar så liten plass som mulig, men som samtidig er skalerbare nok til at det ikke oppstår bilder med dårlig oppløsning.

Ikoner brukt er for det meste hentet fra en tredjepartspakke som heter FontAwesome [17]. FontAwesome er lett å installere og tar veldig liten plass på Android-enheter. Her har man mange ikoner å velge mellom, som både er skalerbare og man kan velge mellom forskjellige versjoner (eks. solid, light, regular osv.) alt etter hva man trenger. Disse ikonene kan man også sette farge på etter behov og er en fantastisk ressurs for å gi et litt rikere brukergrensesnitt. For å bruke dem i koden, kan de hentes inn ved å importere de i filen du arbeider med som en komponent og bruke nettsiden til FontAwesome for å finne navnene på de ikonene man vil bruke. For at disse skal fungere på Android-enheter må de legges til som en applikasjonsressurs under */android/app/src/main/assets* i utviklingsmiljøet. Dette gjør at de blir lett tilgjengelig på enheten som skal vise de.

7.2. FILSTRUKTUR FOR INNHENTING AV GRAFIKK

Når alle illustrasjoner og grafikkfiler er lagt til i undermappen */assets/* i utviklingsmappen, kan vi begynne å importere disse inn på brukergrensesnittet. I JavaScript gjøres dette gjennom en funksjon som heter '*require(filsti)*'. Det eneste problemet med denne funksjonen er at den ikke godtar variabler annet enn direkte *strings* som tilsvarer den korrekte stien til filen. Vi valgte derfor å lage egne JavaScript-filer som holder denne informasjonen i JSON-format. Ved å gjøre dette kan man enkelt importere filen inn i den komponenten man skal bruke den for så å rekursivt gå gjennom den. I disse grafikkfilene har vi da en nøkkel, som er navnet på filen eller navnet på en undergruppe med filer, samt en *source* som da bruker *require* med den riktige stien til det bildet vi bruker.

7.2.1. ILLUSTRASJONSBILDER

Den mest brukte JavaScript-filen med bildestier er filen '*backgroundImagePath.js*'. Under vil man se et utsnitt av denne filen og hvordan den er strukturert.

```
Veikryss: {
```

```
Høyrekryss: {
    X: {
        Vanlig: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-X.png'),
        Gangfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-X-gangfelt.png'),
        Sykkelfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-X-sykkelfelt.png'),
    },
    T: {
        Vanlig: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-T.png'),
        Gangfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-T-gangfelt.png'),
        Sykkelfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-T-sykkelfelt.png'),
    },
    Y: {
        Vanlig: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-Y.png'),
        Gangfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-Y-gangfelt.png'),
        Sykkelfelt: require('../../assets/intersections/hoyrekryss/veikryss-hoyre-Y-sykkelfelt.png'),
    },
},
```

Filen fortsetter nedover for alle de forskjellige typer vei-illustreringer vi anvender i applikasjonen vår. Ved å benytte denne type filstruktur, kan vi enkelt hente ut de riktige bildene til de forskjellige skjermene på en rekursiv og systematisk måte. Ved å programmatisk gå gjennom for eksempel alt som ligger under nøkkelen *Veikryss*, vil vi da få nøklene på alle de forskjellige typene veikryss som er tilgjengelig (høyrekryss, lyskryss og forkjørskryss) og alle underkategorier av disse typene igjen (X-, T- og Y-utforming).

Den ene illustrasjonsiden som ikke tar i bruk denne type filstruktur er Kart-tegneskjermen. Her blir bildet lagret i det midlertidige minnet på enheten. Når bildet blir lagret (i PNG-format), får vi også vite filstien. Filstien blir lagret internt som en tekst-streng på enheten gjennom *AsyncStorage* pakken. Når brukeren vil tegne på et bilde av kartet, så blir altså den siste skjermdumpen som var tatt, tatt i bruk og illustrert på tegneskjermen ved å bruke den lagrede filstien.

7.2.2. TRAFIKKSKILT

Vi anvender også samme type fil for å hente ut alle bildene som ligger lagret under skiltsiden, men her er strukturen litt annerledes. Her nøster vi ikke elementene.

Spesifikt for skiltsiden så måtte gruppen ta i bruk Python for å hente ut alle beskrivelsene for de forskjellige skiltene fra Statens Vegvesen sine nettsider [18]. Gruppen laget et script i Python (se kapittel 8 om Python) som systematisk hentet inn HTML-informasjonen på nettsiden over alle de forskjellige typer trafikkskiltene, for så å lagre relevant info i en *JSON*-fil, sammen med filstien til bildene. Selve skiltene var tilgjengelig for nedlastning i EPS-format, som vi da konverterte til PNG ved bruk av InkScape. Vi benyttet også InkScape for å skalere ned og gjøre de mer anvendelige i applikasjonen.

```
'100_1': {
    navn: 'Farlig sving',
```

```
    beskrivelse: 'til høyre',
    thumbnail: require('../roadSigns/Fareskilt/thumbnails/100_1.png'),
    source: require('../roadSigns/Fareskilt/100_1.png'),
},
'100_2': {
    navn: 'Farlig sving',
    beskrivelse: 'til venstre',
    thumbnail: require('../roadSigns/Fareskilt/thumbnails/100_2.png'),
    source: require('../roadSigns/Fareskilt/100_2.png'),
},
'102_1': {
    navn: 'Farlige svinger',
    beskrivelse: 'den første til høyre',
    thumbnail: require('../roadSigns/Fareskilt/thumbnails/102_1.png'),
    source: require('../roadSigns/Fareskilt/102_1.png'),
},
```

Over kan man se hvordan disse bildene og deres beskrivelse er blitt organisert i filen. Her har hvert bilde en kode som representerer bildet. Når vi hentet ut informasjonen fra nettsiden deres brukte vi dette formatet for å enkelt kunne strukturere bildene og beskrivelsene slik at de tilhørte hverandre. Hver kode har et skiltnavn, en beskrivelse, en filsti til en mindre versjon av bildet og en filsti til en større versjon av bildet.

Det ble laget en slik fil for hver skiltgruppe (fareskilt, forbudsskilt etc.) for å enkelt skille mellom både stien til bildene og for å forenkle det på skiltsiden hvor informasjonen skulle hentes ut. Etter at skriptet var ferdigkjørt, måtte JSON-filene endres til JS-filer for å kunne importeres, samt for å kunne bruke *require*-funksjonen for å hente inn bildene.

7.2.3. DRABARE ELEMENTER

Filene for de drabare elementene holder derimot en litt enklere struktur. Som for de andre filene, så bruker vi også her et nøkkel-verdi-system for å hente inn bilder og data. Her er nøkkelen navnet på bildet. Navnet på bildet er ikke noe som faktisk blir brukt i applikasjonen, men heller for å gi hvert element en unik gjenkjenningsverdi. Hver nøkkel inneholder to undernøkler; *source* og *hasTint. Source* forteller oss hvor bildet ligger og bruker *require*-funksjonen. *hasTint* sier noe om bildet har farger på seg fra før eller ikke. Sistnevnte er nødvendig fordi noen av disse elementene allerede har farger som gjør at bildet blir ugjenkjennelig om man endrer fargen på det til noe annet.

```
Bus: {
    source: require('../../assets/Elements/bus.png'),
    hasTint: false,
},
Bicycle: {
    source: require('../../assets/Elements/bicycle-man.png'),
    hasTint: false,
},
Car: {
    source: require('../../assets/Elements/car-top-view.png'),
```

```
    hasTint: false,
},
```

Gruppen har valgt å bruke to filer med denne strukturen. Den ene er for å sette noen få valgte drabare elementer ved førstegangskjøring av applikasjonen, den andre inneholder alle de forskjellige elementene som er tilgjengelige for brukeren via innstillingene.

8. PYTHON

Gruppen tok i bruk en metode som på engelsk heter '*web scraping*', herav skraping, for å hente informasjon fra websider. Skraping er en metode som programmatisk henter inn HTML-data fra en forespurt nettside. Ved hjelp av tredjepartspakken *BeatuifulSoup,* som analyserer HTML og XML data til en trestruktur, kan man enkelt lese denne type datastruktur. Python gjør det veldig enkelt å sende GET-forespørsler til nettsider som sammen med *BeatuifulSoup* gjør at skraping er en enkel metode for å hente inn relevant data.

Skiltbeskrivelsene til alle skiltene er ikke tilgjengelige via et API-et hos Statens Vegvesen, og derfor valgte gruppen å bruke denne metoden for å slippe kopier/lim inn metoden. Det første som måtte gjøres var å utføre en *inspect* på nettsiden som inneholdt beskrivelsene. Dette gjøres for å se strukturen til websiden. Det kom raskt frem at hvert bilde, sammen med skilt-kode og beskrivelse, lå under en HTML-klasse som het *dokumentpakke*.

FIGUR 15 – HTML-ELEMENT

Det neste var å komme frem til en logikk, da noen *dokumentpakke*-elementer bare inneholdt et bilde og en kode, mens andre kun inneholdt en beskrivelse osv. Skriptet kjørte flere sjekker (*if-statements*) for å lagre de riktige beskrivelsene og navnene til skiltene under riktig skilt-kode. Hver kode ble lagret som en nøkkel med koden som navn som inneholdt 4 andre nøkler; *name, description, source* og *thumbnail*. De to første ble skrapet fra nettsiden, mens de to siste ble hentet fra filstrukturen hvor skriptet kjørte, ved hjelp av skilt-koden for å hente inn de riktige bildene. All denne infoen ble til slutt lagret i en ny JSON-fil (opprettet under kjøring av skriptet).

Sluttresultatet var flere JSON-filer, en for hver skiltgruppe. Bildene ble som nevnt i kapittel 0 om Trafikkskilt, lastet ned via Statens Vegvesen sine API-sider i EPS-format og deretter konvertert til PNG. Alle bildene var på forhånd navngitt i henhold til hvilken kode de hadde, som gjorde at Python-scriptet enkelt kunne legge til filstien til hver skilt-kode. Helt til slutt for å kunne bruke disse filene som importerte strukturer i JavaScript, måtte de gjøre om fra JSON-filer til JS-filer med en JSON-struktur eller *JS Object*. Det var kun noen få endringer som måtte til for å få filene eksportert og klare for importering.

Scriptet ***Scrapper.py*** kan kjøres for å hente inn alle beskrivelsene på nytt om ønskelig, og ligger tilgjengelig under */assets/sign_descriptions/Scrapper.py* i prosjektets rotmappe (se kapittel 0 om filstrukturen). Denne filen kjøres ved å ha Python installert på maskinen, for så å gå til hvor filen ligger.

Herfra kjører man følgende kommando:

```
>python Scrapper.py
```

9. EKSTERNE RESSURSER

Som forberedelse til å utvikle en Android-applikasjon i React Native, har vi gått gjennom deler av et online-kurs i nettopp dette. Kurset vi benyttet er tilgjengelig på Udemy, som er en plattform der det tilbys en mengde ulike kurs på mange ulike områder. Kurset vi benyttet heter "React Native: The Practical Guide" og er laget av Maximilian Schwarzmüller [19]. Schwarsmüller er en erfaren utvikler og en populær kursinstruktør på Udemy.

For utviklingen av applikasjonen har vi hovedsakelig benyttet den offisielle dokumentasjonen til React Native [20]. For de tredjepartspakkene vi har benyttet, som det er en oversikt over i kapittel 0 Avhengigheter, har vi i tillegg benyttet den offisielle dokumentasjonen tilhørende disse pakkene. For enkelte komponenter og deler av applikasjonen har vi benyttet tilleggsressurser.

For å sette opp en "Splash Screen" for applikasjonen har vi benyttet en guide fra Dev Community, som er en plattform bestående av over 600 000 ulike utviklere verden over [21]. En "Splash Screen" er en skjerm bestående av en logo eller et bilde, som vises før selve applikasjonskoden blir lastet inn. Uten en slik skjerm vil brukeren se en svart eller hvit skjerm før applikasjonen lastes inn, noe vi ønsket å unngå.

For å sikre at illustrasjonene vi laget av de ulike veikryssene, rundkjøringene o.l. ble riktige, har vi benyttet ulike håndbøker fra Statens Vegvesen. Siden applikasjonen er laget for at trafikklærere skal bruke denne i trafikkopplæringen, er det veldig viktig at de illustrasjonene de skal tegne over stemmer med virkeligheten og de reglene og prinsippene de lærer bort til sine elever. For å sørge for riktig utforming av de ulike veikryssene, rundkjøringene o.l., benyttet vi Håndbok N100 – Veg- og gateutforming [22]. For å sørge for riktig veioppmerking av de ulike trafikksituasjonene benyttet vi Håndbok N302 - Vegoppmerking [23].

I tillegg har vi benyttet Statens Vegvesens Håndbok V851 – Læreplan for førerkortklasse B, B kode 96 og BE [24]. Denne håndboken inneholder læreplanmålene for disse førerkortklassene (de vanligste førerkortklassene), og denne informasjonen benytter vi i applikasjonen på siden for nettopp læreplanmål.

10. REFERANSER

[1] "Use JSDoc: Getting Started with JSDoc 3," JSDoc, 2017. [Online]. Available: https://jsdoc.app/about-getting-started.html. [Accessed 15 April 2021].

[2] json.org, "JSON," [Online]. Available: https://www.json.org/json-en.html. [Accessed 1 Mai 2021].

[3] React Native, "Setting up the development environment," [Online]. Available: https://reactnative.dev/docs/environment-setup. [Accessed 11 Mars 2021].

[4] Google Developers, "Create and Manage Virtual Devices | Android Developers," Google, [Online]. Available: https://developer.android.com/studio/run/managing-avds.html?authuser=1. [Accessed 9 Januar 2021].

[5] npm Docs, "CLI commands | npm Docs," [Online]. Available: https://docs.npmjs.com/cli/v7/commands. [Accessed 20 Februar 2021].

[6] Git, "Git - git Documentation," [Online]. Available: https://git-scm.com/docs/git. [Accessed 15 Mars 2021].

[7] Atlassian, "Jira | Issue & Project Tracking Software | Atlassian," 2021. [Online]. Available: https://www.atlassian.com/software/jira. [Accessed 1 Februar 2021].

[8] React Native, "Publishing to Google Play Store," 2021. [Online]. Available: https://reactnative.dev/docs/signed-apk-android. [Accessed 10 Mai 2021].

[9] freeCodeCamp, "Functional vs Class Components in React Native," 17 Januar 2020. [Online]. Available: https://www.freecodecamp.org/news/functional-vs-class-components-react-native/. [Accessed 9 Mars 2021].

[10] M. Hamedani, "React Functional or Class Components: Everything you need to know," 10 September 2018. [Online]. Available: https://programmingwithmosh.com/react/react-functional-components/. [Accessed 9 Mars 2021].

[11] React, "Introducing Hooks - React," [Online]. Available: https://reactjs.org/docs/hooks-intro.html. [Accessed 9 Mars 2021].

[12] React, "Hooks FAQ," [Online]. Available: https://reactjs.org/docs/hooks-faq.html#should-i-use-hooks-classes-or-a-mix-of-both. [Accessed 9 Mars 2021].

[13] P. Lehoux, "45 % Faster React Functional Components, Now," 3 Mai 2017. [Online]. Available: https://medium.com/missive-app/45-faster-react-functional-components-now-3509a668e69f. [Accessed 9 Mars 2021].

[14] P. Divine, "Case Styles: Camel, Pascal, Snake, and Kebab Case," 3 November 2018. [Online]. Available: https://betterprogramming.pub/string-case-styles-camel-pascal-snake-and-kebab-case-981407998841. [Accessed 2 Mai 2021].

[15] J. Schoeman and V. Larsson, "React Native Styling: Structure for Style Organization," thoughtbot, inc, 18 mars 2019. [Online]. Available: https://thoughtbot.com/blog/structure-for-styling-in-react-native. [Accessed 21 mars 2021].

[16] Inkscape, "Draw Freely | Inkscape," [Online]. Available: https://inkscape.org/. [Accessed 10 Mars 2021].

[17] Fonticons, Inc, "Font Awesome," FontIcons, [Online]. Available: https://fontawesome.com/. [Accessed 20 Februar 2021].

[18] Statens Vegvesen, "Trafikkskilt | Statens Vegvesen," 16 Mars 2021. [Online]. Available: https://www.vegvesen.no/trafikkinformasjon/langs-veien/trafikkskilt/fareskilt. [Accessed 3 Mai 2021].

[19] M. Schwarzmüller, "The Practical Guide to React Native," Udemy, 2021. [Online]. Available: https://www.udemy.com/share/101WauB0IbcVlSRX4=/. [Accessed 1 Januar 2021].

[20] Facebook Inc, "Introduction - React Native," 2021. [Online]. Available: https://reactnative.dev/docs/getting-started. [Accessed 22 April 2021].

[21] C. M. Pandey, "How to add Splash Screen in a React Native Android app," 26 August 2020. [Online]. Available: https://dev.to/cmcodes/how-to-add-splash-screen-in-a-react-native-android-app-287l. [Accessed 12 Mars 2021].

[22] Statens Vegvesen, "N100 - Veg- og gateutforming," 2019. [Online]. Available: https://www.vegvesen.no/fag/publikasjoner/handboker/om-handbokene/vegnormalene/n100. [Accessed 20 Januar 2021].

[23] Statens Vegvesen, "N302 - Vegoppmerking," 2015. [Online]. Available: https://www.vegvesen.no/fag/publikasjoner/handboker/om-handbokene/vegnormalene/n302. [Accessed 20 Januar 2021].

[24] Statens Vegvesen, "V851 - Læreplan for førerkortklasse B, B kode 96 og BE," 2016. [Online]. Available: https://www.vegvesen.no/fag/trafikk/opplaeringsbransjen/forskrifter-og-laereplaner. [Accessed 15 Mars 2021].

[25] R. Gaba and R. Atul, "Theme Variables | React Made Native Easy," [Online]. Available: https://www.reactnative.guide/8-styling/8.1-theme-variables.html. [Accessed 21 mars 2021].

SLUTTRAPPORT

BACHELORPROSJEKT 087: UTVIKLING AV ANDROID-APPLIKASJON FOR TRAFIKKLÆRERE

Oliver Elias Nilssen, Joakim Heitmann Tronseth og Silje Tanemsmo

FORORD

Vi er tre studenter ved studieprogrammet Informatikk med spesialisering i drift av datasystemer ved NTNU. Studiet har som hensikt å utdanne kompetente personer som har en dyp innsikt innen sikkerhet, nettverksdrift, skyløsninger og generelle IT-kunnskaper. Programmet er veldig praktisk tilrettelagt, og vi har tilegnet oss kunnskap som er veldig etterspurt i arbeidsmarkedet. Gjennom hele studiet har vi jobbet i grupper og taklet store prosjekter, dette har hjulpet oss gjennom utførelsen av dette bachelorprosjektet.

Hensikten med dette prosjektet har vært å sette seg inn i hvordan man gjennomfører et prosjekt fra start til slutt. Dette innebærer:

- Planlegging av prosjektet
- Utarbeiding av diverse dokumenter
- Å sette seg inn i ny utviklingsmetodikk
- Utvikling av selve applikasjonen
- Administrering og motivering av teamet
- Møter, møteinnkalling og møterapportering

Vi har selv ønsket å lære mer om hvordan man utvikler en applikasjon og om koding generelt, ettersom vi tidligere i studiet har hatt prosjekter hvor vi har utviklet både et spill og en skrivebordsapplikasjon. Vi ønsket også å jobbe mer *smidig* og valgte derfor en metodikk som innebar at vi jobbet i såkalte *sprints*, hvor hver sprint var på ca. 2 uker. Denne måten å jobbe på er blitt mer og mer relevant i arbeidslivet og benyttes for å kunne sikre en kontinuerlig forbedring av produkter og tjenester.

Vi vil spesielt si takk til Hege Tilset, som hadde den originale ideen om en applikasjon for illustrering av trafikksituasjoner, og for hennes bidrag som testperson for applikasjonen. Stein Meisingseth, for hans veiledning og hjelp ikke bare gjennom dette prosjektet, men også gjennom hele studiet. Jostein Lund, for å si seg villig til å være testperson og for hans hjelp og veiledning gjennom hele studiet. Georg Kippernes, for hans støtte og tilbakemeldinger som kontaktperson hos Sopra Steria. Marlene Berg Sundsøy (interaksjonsarkitekt hos Sopra Steria), for hennes tilbakemeldinger i forhold til brukerinteraksjon og design.

INNHOLDSFORTEGNELSE

1. OPPGAVEBESKRIVELSE

Denne prosjektoppgaven har gått ut på å utvikle en applikasjon for Android, som skal fungere som et digitalt hjelpemiddel for trafikklærere i trafikkopplæringen. Selv om studieprogrammet fokuserer på drift av datasystemer, har vi gjennom studiet hatt flere emner relatert til programmering og utvikling, noe som var medvirkerne til at vi valgte denne oppgaven som avsluttende bachelorprosjekt.

Ideen til prosjektet fikk vi fra en trafikklærer som savnet å ha et verktøy for å blant annet kunne illustrere ulike trafikksituasjoner for sine elever. Et slikt verktøy må ha en del basisillustrasjoner, som for eksempel ulike former for veikryss og rundkjøringer, som trafikklæreren deretter kan tegne oppå. Hensikten er å blant annet vise hva som er korrekt plassering i veibanen, når man skal gi tegn, og hvem som har vikeplikt i ulike situasjoner. Det eksisterer ikke et slikt verktøy for enheter som kjører Android, noe som fører til at de fleste trafikklærere i dag benytter penn og papir for å illustrere og vise ulike trafikksituasjoner for elevene sine. Dette blir lite presist, da man ikke har tid til å tegne opp for eksempel veikryss med riktige avstander, korrekt veioppmerking, skilting osv. Med denne oppgaven ønsket vi å utvikle et verktøy som både letter arbeidshverdagen til trafikklærere og øker kvaliteten på trafikkopplæringen.

For utvikling av applikasjonen har vi benyttet rammeverket React Native, som er et mye brukt rammeverk for å utvikle applikasjoner til Android, iOS og også web. Oppgaven innebar at vi måtte sette oss inn i alle elementer som inngår i utvikling av mobile applikasjoner med React Native, og utforming av brukergrensesnitt. Oppgaven innebar også at vi måtte sette oss inn i riktig utforming av veikryss, rundkjøringer o.l., slik at basisillustrasjonene i applikasjonen hadde korrekt veiutforming, veimerking og skilting.

Vår oppdragsgiver for denne oppgaven er Sopra Steria, som er Norges ledende konsulentselskap innen digitalisering. Vi definerte og utformet oppgaven selv, men Sopra Steria v/ vår kontaktperson Georg Kippernes i Sopra Steria Trondheim har veiledet oss gjennom hele prosjektperioden, og bidratt til at vi sitter igjen med et resultat vi er fornøyd med.

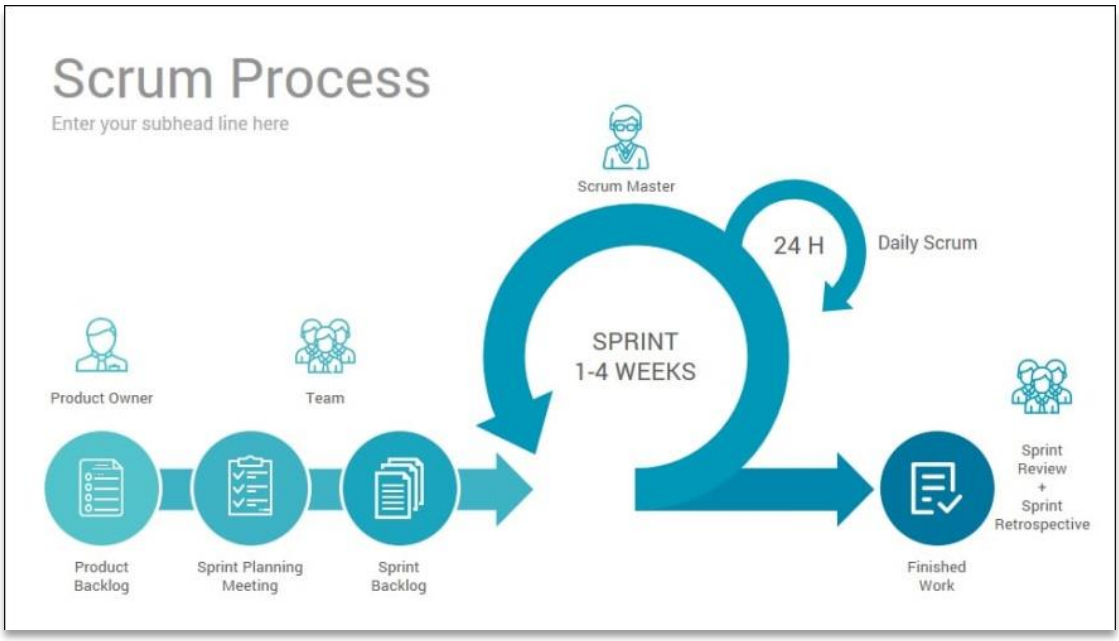
2. HVORDAN OPPGAVEN BLE LØST

I dette kapittelet går vi gjennom hvordan vi rent praktisk løste oppgaven. Hvilke metoder, standarder, litteratur og programvare vi benyttet for å løse oppgaven. Vi går også gjennom hvordan arbeidet ble fordelt innad i gruppen og mellom de ulike aktivitetene i prosjektet, samt hva vi har utarbeidet av dokumentasjon.

2.1. METODER OG STANDARDER

I kravdokumentet har vi tatt i bruk metoden UML (Unified Modelling Language) for å kunne skape en oversiktlig representasjon av alle funksjonene i applikasjonen, og hvordan de henger sammen. Ved å først tegne opp alle funksjonene og hvordan de henger sammen, ble det mye enklere å utforme de forskjellige Use Case-diagrammene.

Vi har tatt i bruk Scrum-metodikk for å kunne jobbe iterativt med utviklingen av applikasjonen. Tanken bak er at man deler opp utviklingen i sprints der hver sprint har en varighet på 1-4 uker. Hver sprint starter ved at man først oppdaterer og legger inn nye oppgaver i en backlog, og deretter fordeler oppgavene ut til de forskjellige medlemmene i teamet. Det holdes korte daglige scrum-møter, også kalt stand-up's, med en varighet på omtrent 15 minutter. Der skal hvert medlem fortelle hva de har gjort, hva de skal gjøre i dag og eventuelt hva de har problemer med og om de trenger hjelp. På slutten av hver sprint, gjøres det en sprint review og en sprint retrospekt. Her skal teamet fortelle hva som gikk bra, hva som gikk dårlig og hvordan de kan forbedre seg. Deretter starter hele prosessen på nytt igjen.



FIGUR 16 - SCRUM-PROSESSEN

For å bygge selve applikasjonen har vi tatt i bruk programmeringsspråket *JavaScript* og rammeverket React Native. Hovedgrunnen til at vi valgte React Native er at vi alle har erfaring fra JavaScript og React fra tidligere i studiet, og at rammeverket ikke er bundet til et OS. Dette betyr at det er mulig å

kjøre applikasjonen både på Android og iOS ved å gjøre noen små endringer i koden. Dette er ikke noe vi har fokusert på nå, men er en ting vi har tenkt på hvis vi velger å fortsette utviklingen av applikasjonen.

For å sørge for at navnene som benyttes i koden er lett å forstå har vi operert med forskjellige navngivingsmetoder basert på hvilken type element det er.

TYPE	NAVNGIVINGSMETODE	EKSEMPEL
Mapper	camelCase	sketchHeader/
Filnavn komponenter	PascalCase	SketchArea.js
Filnavn variabler	camelCase	themeVariables.js
Filnavn grafikk	kebab-case	veikryss-hoyre-X.png
Komponenter	PascalCase (likt filnavnet)	SketchArea
Variabler	camelCase	roadDesign
Theme-variabler	SNAKE_CASE	FONT_SIZE_LARGE
Funksjoner	camelCase	radioButtonChange

TABELL 27 - NAVNGIVNINGSMETODER I KODEN

De forskjellige oppgavene i backloggen har et eget prefiks basert på hvilken type oppgave det er, dette for å gjøre det enklere å vite hvilken type hovedoppgave man har ført timer på. Vi har benyttet følgende prefixer:

- *D – Dokumentasjon (forstudierapport)*
- *A – Prosjektadministrasjon*
- *DE – Design av applikasjonen*
- *R – Research*
- *KD – Kravdokument*
- *K – Koding (Alpha og MVP)*
- *M – Møtevirksomhet*
- *TB – Testing og forbedring*
- *BT – Brukertesting*
- *DR – Driftsdokumentasjon*
- *SD – Sluttrapport*

Vi har også opprettet en standard for navngiving av de forskjellige grenene i Git, slik at man ser hvilken oppgave i Jira den tilhører. Oppgaver (issues) i Jira starter alltid med *BADR* og et nummer. For oppgaven *BADR-1* skal navnet på branchen starte med *BADR-1*.

2.2. BRUK AV LITTERATUR OG INTERNETT

I starten av prosjektet gjennomførte vi et nettkurs i React Native på Udemy, kalt *React Native: A practical guide* [1]. Udemy er en nettbasert undervisningsplattform. Dette gjorde vi for å sette oss inn i syntaks og nyere teori og standarder for rammeverket. Ellers har vi benyttet den offisielle React

Native-dokumentasjonen [2] og mye StackOverflow når vi har lett etter løsninger på de problemene vi har støtt på.

For å kunne utforme illustrasjonene som vi benytter som bakgrunn på de forskjellige illustrasjonssidene i applikasjonen, har vi hentet informasjon og inspirasjon fra forskjellige brosjyrer som ligger tilgjengelig på Statens Vegvesens hjemmeside. Vi har også hentet bilder, navn og beskrivelser av trafikkskiltene herfra.

I applikasjonen har vi tatt i bruk flere pakker og biblioteker, og vi har da brukt den offisielle dokumentasjonen for å forstå hvordan disse skal benyttes. Herunder vil vi spesielt nevne:

- React-navigation
- Terrylinla/React-native-sketch-canvas
- Jsdoc (Better-docs)
- React-native-popup-menu

Disse vil ikke bli beskrevet noe videre her, ettersom vi har dokumentert disse under 4.4. Avhengigheter i Driftsrapporten.

I begynnelsen brukte vi mange flere pakker og biblioteker, og vi brukte mye tid på å sette oss inn i dokumentasjonen. Ettersom dette kan føre til svakheter i applikasjonen fant vi ut at det var bedre å ha så få avhengigheter som mulig, vi begynte derfor å fase ut pakkene og lage våre egne komponenter med basis i de offisielle React Native komponentene. Hadde vi hatt mer kjennskap til React Native, ville vi ha gjort dette fra begynnelsen av.

2.3. Beskrivelse av standardprogramvare

Vi har brukt Android Studio for å få tilgang til verktøy nødvendig for å kjøre emulatorer eller debugge på egne enheter. For å få dette til å fungere må man installere JDK (Java Development Kit), som vil gjøre det mulig for maskinen å forstå og kjøre java-kode, samt Node. Node er en kryssplattform for server- og nettverksapplikasjoner, og har blitt benyttet til å kjøre applikasjonen vår på emulatorene. En annen viktig funksjon Node.js gir er muligheten til å installere forskjellige pakker og avhengigheter gjennom dens pakkebehandler «npm». Det er slik vi har lagt til de forskjellige pakkene vi har benyttet i vår applikasjon.

For å holde oversikt over backlog og tidsbruk har vi tatt i bruk Jira med en ekstra tilleggsapplikasjon for logging av tid kalt Clockify. Jira benytter seg av Kanban-metodikken for organiseringen av oppgaver. Først oppretter man et prosjekt, deretter en backlog for prosjektet. I backloggen kan man opprette oppgaver basert på om de er «story», «task» eller «bug», når man så starter sprinten må disse oppgavene flyttes inn i sprinten. Man kan tildele et medlem oppgaven, og den kan bli gitt en prioritet. Oppgavene kan ha statusen To Do, In Progress eller Done, basert på om oppgaven er påbegynt eller ei, eller om den er ferdigstilt. Det er også mulig å skrive inn tiden brukt på hver oppgave.

Alle i teamet har tatt del i kodingen, og for at dette skal kunne gå greit over flere enheter uten å få problemer med koden, har vi tatt i bruk Git via GitHub. Med Git kan man klone koden som ligger lagret i skyen og lagre den lokalt på datamaskinen. Deretter kan man opprette en «branch» eller gren, som er en kopi av den originale klonen. I denne kopien kan man så gjøre endringer på koden uten at det påvirker den øvrige kodebasen. Når man så er fornøyd med det man har gjort, og det ikke

oppstår noen konflikter, kan man slå sammen den nye koden med kodebasen. Alle med tilgang til kodebasen vil nå ha tilgang til endringene som har blitt gjort.

Vi har i hovedsak benyttet editoren Visual Studio Code for å programmere selve applikasjonen, men har også tatt i bruk Atom i enkelte tilfeller på grunn av et tilleggsprogram som gjør det mulig å jobbe i sanntid i samme kode.

For å tegne de forskjellige illustrasjonene som benyttes som bakgrunn på illustrasjonssidene i applikasjonen, har vi brukt InkScape, et open source illustrasjons- og bilderedigeringsprogram. Vi har også benyttet det for å endre filtype på skiltbildene vi hentet fra Statens Vegvesen sine sider.

For å dokumentere prosjektet har vi anvendt diverse Office-applikasjoner; Word for rapporter, referater og møteinnkallinger, Excel for å føre timelister, MS Project til å lage Gantt-diagram og Visio til å lage diverse andre diagram. Vi har også brukt Teams for koordinering og kommunikasjon, hovedsakelig over video-møter. Gjennom Teams har vi også tatt i bruk lagring av alle dokumenter i OneDrive. For en mer meldingsbasert kommunikasjon har vi tatt i bruk Slack. Slack har også vært koblet opp mot Git, slik at hver gang en ny endring har blitt lagt til har alle fått beskjed om det.

2.4. Arbeidsfordeling

Vi ønsket at alle skulle få ta del som Scrum Master i løpet av prosjektet, og satte derfor opp en plan med en rullerende Scrum Master som ble byttet ut ved hver sprint. En Scrum Master skal ha et overordnet ansvar for at oppgavene og målene for sprinten blir gjennomført.

SPRINT	UKENUMMER	SCRUM MASTER
1	1-2	Oliver
2	3-4	Joakim
3	5-6	Silje
4	7-8	Oliver
5	9-10	Joakim
6	11-12	Silje
7	13-14	Oliver
8	15-16	Joakim
9	17-18	Silje
10	19-20	Oliver

Tabell 28 - Oversikt rullerende Scrum Master

I likhet med rollen som Scrum Master har vi hatt en rullerende plan for hvem som er møteleder og hvem som skal føre referatet fra møtene. Dette har fungert veldig bra, men vi ser at dette kunne vært problematisk i et team som ikke kjenner hverandre.

MØTE	UKE NR.	DATO	MØTELEDER	REFERENT
1	1	8. januar	Oliver	Silje

2	3	20. januar	Joakim	Oliver
3	5	3. februar	Silje	Joakim
4	7	17. februar	Oliver	Silje
5	10	10. mars	Joakim	Oliver
6	13	31. mars	Silje	Joakim
7	16	21. april	Joakim	Silje
8	19	12. mai	Oliver	Joakim

TABELL 29 - OVERSIKT RULLERENDE MØTELEDER OG MØTEREFERENT

Vi er tre personer som har jobbet sammen over en ganske lang periode, vi har derfor et klart bilde av våre styrker og svakheter. I den forbindelsen har vi ikke nødvendigvis gitt noen et offisielt ansvar for eksempelvis koding eller dokumentasjon, men heller mer en rådgivende rolle innenfor de forskjellige delene. Oliver har for eksempel mer kunnskap innen programmering enn oss andre, Silje er veldig god på strukturering av dokumentasjon, og Joakim er veldig god på Visio og MS Project. Tabellen under, som viser hvordan de ulike arbeidsoppgavene ble fordelt mellom oss, reflekterer dette.

Høy	Middels	Lav

TABELL 30 - FORKLARENDE TABELL TIL TABELL 5

Arbeidsart	Oliver	Joakim	Silje
Programmering			
Dokumentasjon			
Testing			
Design			
Diagrammer og modeller			
Research			
Administrasjon			

TABELL 31 - FORDELING AV ARBEID MELLOM TEAMMEDLEMMER

Når det kommer til selve fordelingen av oppgavene har vi hatt et møte i starten av hver sprint, hvor vi har oppdatert backloggen og fordelt oppgaver. Dette ble gjort i plenum. Ble man ferdig med oppgavene man hadde tatt på seg var man fri til å selv velge andre oppgaver fra backloggen. Kom man over et problem eller en oppgave som var glemt, var det bare å legge de til i backloggen.

2.5. OVERSIKT OVER UTARBEIDET DOKUMENTASJON

Gjennom prosjektet har vi utarbeidet følgende dokumentasjon.

1) Forstudierapport
    a. Prosjektplan
2) Kravdokument
3) Driftsdokument
    a. JSdoc
4) Sluttrapport
5) Prosjekthåndbok
    a. Arbeidskontrakt
    b. Avtaleskjema
    c. Framdriftsplan
    d. Møteinnkallinger
    e. Møtereferat
    f. Timelister
    g. Sprintrapporter
6) Dokumentasjon av brukertester
7) Høynivå prototype

3. GJENNOMFØRING AV PROSJEKTET

Prosjektet er gjennomført innenfor en tidsramme på cirka 20 uker fra 7. januar til 21. mai. Resultatet av prosjektet er en Android applikasjon som ligger tilgjengelig for nedlastning via Google Play Store. Videre i dette kapittelet skal vi vurdere måloppnåelse ut ifra prosjektstart, der vi tar for oss hva som gikk bra og hva vi kunne gjort annerledes. Deretter vurderer vi måloppnåelse i forhold til prosjektplanen, før vi drøfter hvorvidt resultatene fra forstudierapporten og kravdokumentet er blitt oppnådd i henhold til målene satt i starten av prosjektet. Til slutt tar vi for oss andre ting som dokumenterer prosessen.

3.1. MÅLOPPNÅELSE UT IFRA PROSJEKTSTART

Her er fokuset på prosessen og sluttresultatet i forhold til det vi planla ved prosjektstart. Vi reflekterer over det arbeidet vi har gjort i forhold til de målene vi satte oss tidlig i prosjektet. Herunder går vi gjennom hva gikk bra, hva som gikk dårlig og hva som kunne vært gjort annerledes. Vi går også gjennom begrensninger på systemet.

3.1.1. HVA GIKK BRA

Prosjektet har utformet seg ganske bra i forhold til de målene som ble satt i forstudierapporten og alle på gruppen er fornøyde med det resultatet som blir levert. Vi har klart å oppfylle så å si alle målene som ble satt i forstudierapporten og alle på gruppen har fått opparbeidet seg erfaring, lært mye nytt og vi har som gruppe blitt enda bedre kjent.

EFFEKTMÅLENE

Først kan vi se på effektmålene som vi satte i forstudierapporten:

1. Forenkle måten trafikklærere illustrerer trafikksituasjoner for sine elever
2. Forbedre utbyttet av trafikkopplæringen ved at illustrering av trafikksituasjoner blir mer forståelig for elevene

Det er vanskelig å presisere om disse målene faktisk er oppnådd. Tar vi utgangspunkt i kunden, Hege Tilset, så har vi fått veldig gode tilbakemeldinger på testene vi har kjørt med henne. Hun har også fått en testversjon installert på sitt nettbrett som hun som trafikklærer nå benytter seg av. Ut fra tilbakemeldingene vi har fått fra henne kan man anse at disse målene er oppnådd, i alle fall til en viss grad. For å vite om disse er helt klart oppfylt må vi ha tilbakemelding fra flere kilder og spesielt da personer som jobber som trafikklærere, eller foresatte som driver kjøreopplæring med sine ungdommer. Ut ifra den informasjonen vi har så kan vi se oss ganske fornøyde med oppnåelsen av effektmålene. Vi anser produktet vårt som en god ressurs og et behjelpelig verktøy innenfor dette nisjemarkedet.

RESULTATMÅLENE

Resultatmålene vi satte oss i forstudierapporten var som følger:

1. Ha utviklet en Android-applikasjon i løpet av prosjektperioden, der trafikklærere kan illustrere ulike trafikksituasjoner
2. Publisere applikasjonen på Google Play Store

Prosjektet har oppnådd begge resultatmålene som ble satt. Vi har både klart å levere en Android-applikasjon som kan benyttes av trafikklærere og applikasjonen er tilgjengelig via Google Play Store som en gratis ressurs som kan lastes ned på brukerens enhet. Applikasjonen kan lastes ned både på nettbrett og mobil, selv om prosjektet i utgangspunktet bare hadde fokus på nettbrett kompatibilitet.

PROSESSMÅLENE

Prosessmålene vi satte i forstudierapporten var følger:

1. Erfaring og økt kompetanse om smidig utvikling og spesifikt Scrum
2. Erfaring og økt kompetanse om applikasjonsutvikling med rammeverket React Native
3. Erfaring med fjernsamarbeid
4. Oppnå karakter B eller bedre som resultat i emnet IDRI3001 Bacheloroppgave i drift av datasystemer

Disse målene er relatert til selve prosessen med å gjennomføre prosjektet, i stedet for produktet som skal produseres. Derfor kan vi se at målene baserer seg på erfaring og økt kompetanse. Gruppen har gjennom dette prosjektet opparbeidet seg mye erfaring på alle punktene som prosessmålene referer til. Det som spesielt kan utmerke seg **punkt 2**, applikasjonsutvikling gjennom rammeverket React Native. På starten av prosjektet hadde ingen i gruppen jobbet direkte med React Native, men hadde litt erfaring fra lignende rammeverk som React. Vi har brukt mye tid på å lese dokumentasjon og å teste ut kode for å bli bedre kjent og mer komfortabel med dette rammeverket.

Punkt 1 og 3 kan også sies å være godt oppfylt. Gruppen hadde tidligere litt erfaring med Scrum som utviklingsmetodikk, men gjennom prosjektet har vi hatt mye fokus på å følge denne praksisen, samt å tilrettelegge den for vårt bruksområde. Annenhver uke har vi gjennomgått en *sprint review* hvor vi har diskutert hva som gikk bra, hva som gikk dårlig og hva som kunne forbedres til neste sprint. Når det gjelder **punkt 3** har vi spesielt klart å opparbeide oss god erfaring med fjernsamarbeid. Da 1 av 3 gruppemedlemmer sitter i en annen by har det vært viktig for oss at kommunikasjonen skjer på en måte som gjør at alle får med seg viktig informasjon og endringer. Som gruppe har vi hatt videosamtaler nesten hver arbeidsdag i løpet av prosjektet, og på de dagene vi ikke har hatt videosamtale så har vi brukt verktøy som *Slack* og *Teams* for å oppdatere hverandre. Vi har klart å samarbeide veldig godt selv om vi har oppholdt oss på ulike steder i landet.

Når det gjelder **punkt 4**, har vi under arbeidet med dette dokumentet ingen besvarelse på hvilken karakter gruppen oppnådde. Derfor kan vi ikke kommentere på om dette punktet gikk som ønsket eller ikke.

GENERELT

Som nevnt på starten av dette kapittelet har samarbeidet på dette prosjektet gått meget bra. Som gruppe har vi oppnådd alle de målene vi ønsket å oppnå, i tillegg til at vi har et velfungerende produkt som er klart for overtakelse eller videreutvikling. illusTrafikk-applikasjonen er et produkt vi

er ganske stolte av og har lagt inn mange timer med arbeid på. Selv med komplikasjoner som å måtte arbeide hele semesteret gjennom fjernsamarbeid, anser vi samarbeidet som velfungerende.

Prosjektet har vært ryddig, hatt en fin flyt og vi har satt klare mål for hver sprint. Hver sprint har vi gjennomgått hvilke oppgaver som gjenstår, hva som må gjøres videre og hvordan vi ligger an i forhold til den helhetlige planen. Vi har også fått gode tilbakemeldinger i løpet av veiledningsmøtene vi har hatt med Stein Meisingseth og Georg Kippernes.

3.1.2. HVA GIKK DÅRLIG OG HVA KUNNE VÆRT GJORT ANNERLEDES

Når det gjelder prosesser som har gått mindre bra i løpet av disse 20 ukene, ligger hovedproblemet rundt det å holde 'stand-up'-møte minst 3 ganger i uken. Akkurat dette ga varierte resultater hver uke og vi har ikke alltid vært like flinke til å opprettholde denne regelen. Stand-up er korte møter på cirka 10-15 minutter på starten av hver arbeidsdag (tirsdag, onsdag og fredag) hvor vi informerte hverandre om hva vi har arbeidet med og hva det neste vi skal gjøre er. I starten av prosjektet ble dette gjort hver uke, men når vi begynte å nærme oss rundt uke 10, ble disse møtene redusert eller erstattet med vanlige møter hvor vi bare jobbet sammen.

Noen andre prosesser som ikke gikk helt etter planen var planlegging av tester. Her hadde vi ganske ambisiøse planer om å teste med flere personer for å kunne få så god tilbakemelding som mulig før vi lanserte produktet. Dessverre viste det seg å være vanskelig å finne testere grunnet flere årsaker, blant annet at covid-19 situasjonen gjorde det vanskelig å møte andre personer og få kontakt med de rette personene. Vi prøvde å kontakte Nord Universitet som har opplæring av trafikklærere, men her var det lite respons fra både lærere og elever. Noen andre grunner til at dette gikk dårlig var nok at gruppen ikke planla dette godt nok, vi skulle ha planlagt og satt opp tester i forkant mye tidligere i prosjektet.

Når det gjelder selve produktet har vi klart å implementere nesten alle de funksjonene som ble satt som krav fra starten av. Det er noen få funksjonaliteter som ikke ble med, som: «*Mulighet til å velge flere kombinasjoner av sykkelfelt, busslomme og gangfelt (for eksempel busslomme og sykkelfelt)*» for veikryss og rundkjøring. Disse funksjonene ble prioritert lavest da det innebar mye arbeid å lage disse illustrasjonene og fokuset til gruppen har vært på programmering, dokumentering og funksjonalitet som er viktigst. Det er mulighet for å kunne velge både sykkelfelt og gangfelt, for alle de ulike veikryssene og rundkjøringene vi har lagt inn. Men det er ikke mulig å velge begge deler (sykkelfelt og gangfelt), og valg for busslomme er blitt nedprioritert.

Et punkt som vi hadde satt oss som et internt krav var å oppfylle WCAG 2.0 kravene [3]. Dette er krav i forhold til utforming og tilgjengelighet for mennesker med nedsatt funksjonsevne (fargeblinde, døve osv). Dette innebærer at fargene møter en viss kontrast og at det ligger muligheter til rette for at døve eller blinde kan bruke applikasjonen. Det er blitt lagt fokus på at fargene i applikasjonen møter visse standarder i forhold til størrelse og farger, men dette har gruppen dessverre ikke lagt inn mye arbeid på i forhold til hva som var planlagt. Siden applikasjonen i utgangspunktet er en offline applikasjon, så er det også mindre strenge krav til dette.

Vi har også fått veldig gode tilbakemeldinger fra de brukertestene vi klarte å gjennomføre. Det eneste som har vært litt problematisk underveis har vært at vi ikke har hatt noen veiledere som har kunnet hjelpe oss med selve programmeringen. Dette har ført til at vi har brukt mye tid på å selv lære oss det som trengs for å kunne få den funksjonaliteten vi ønsket. Mangelen av en fast

programmeringsveileder har ført til at vi har brukt mye tid på 'prøve og feile'-metoden, noe det ble mindre av etter hvert som vi ble mer komfortable med språket og rammeverket.

Når det kommer til publiseringen av appen var dette litt dårlig planlagt. Applikasjonen har vært tilgjengelig for interne testere på Google Play Store en god stund, men ble ikke før helt på slutten av prosjektet tilgjengelig for produksjon (altså tilgjengelig for alle brukere). Dette gjør at vi ikke har fått testet appen i full skala. Google informerer om at vurderingene tar litt lengre tid enn før, på grunn av Covid-19, og hadde vi tatt litt forhåndsregler og gjort litt undersøkelser kunne gruppen nok ha lastet den opp litt tidligere for å sikre oss om at den var lagt ut i god tid før innleveringen av dette prosjektet.

3.1.3. BEGRENSNINGER PÅ SYSTEMET

I denne delen skal vi gå igjennom hvilke begrensninger systemet har på hver av de ulike sidene brukeren kan få tilgang til.

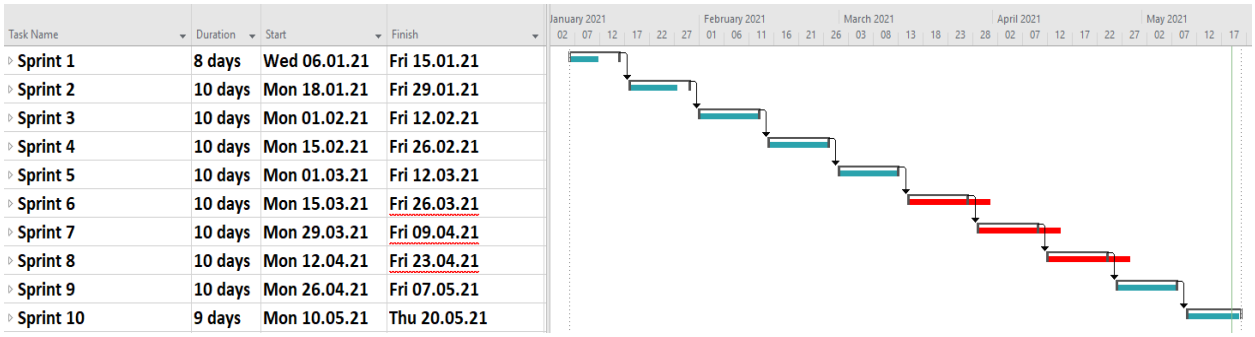
SKJERMER/KOMPONENTER	FUNKSJONALITET	BEGRENSNING
Startskjermen	Oversikt over alle andre sider man kan gå til, som klikkbare elementer. Siden er inndelt i funksjonalitet (illustrering, andre funksjoner og nyttige lenker). Det er også en knapp for å gå direkte til innstillingene fra denne skjermen.	• Kan ikke gå direkte til karttegneskjermen, men må gå via kart • På veldig små mobiler er ikke alle elementene tilgjengelig
Tegneskjermene	Dette er skjermen hvor man kan tegne/illustrere på. Det finnes 4 forskjellige tegneskjermer, en for hver type vei (veikryss, landevei, rundkjøring og fartsøknings- og reduksjonsfelt). Alle tegneskjermene har samme innhold med mulighet for å tegne, endre veidesign/utforming, viske, legge til drabare elementer og angre siste valg man tok.	• Man kan ikke angre posisjonering av drabare elementer, bare når de ble lagt til. • Trykker man på noe i menyen til et drabart element (popout), blir det tegnet på tegneskjermen • Ingen mulighet for å legge til busslomme
Læreplanmålskjermen	En oversikt over læreplanmålene for klasse B og klasse B kode 96 og BE. Her kan man velge mellom de forskjellige trinnene i klassene og se alle læreplanmålene for hvert trinn.	• Ingen søkemuligheter for å finne ønsket informasjon
Myndighetspyramiden	Et bilde som illustrerer myndighetspyramiden.	
Skiltskjermen	Denne siden viser en oversikt over alle tilgjengelige skilt sortert på kategori. Her kan man trykke på et skilt for å se det i større format. Trykker man så på det igjen, vil man få opp navnet på skiltet samt en beskrivelse. Kan	• Kan ikke søke på spesifikke skilt • Navn eller kode på skiltene vises ikke i oversikten som gjør det vanskelig å finne

	brukes for å teste elever om skiltkunnskaper.	spesifikke skilt basert på kode eller navn
Kartskjermen	En side med Google Maps integrert, som viser lokasjonsdata om den er aktivert. Her kan man sette ned en markør å lagre for senere bruk, samt zoome inn og ut på kartet. Herfra kan man også ta en skjermdump av hva man ser for å bruke på Karttegneskjermen.	• Ikke mulighet for å lagre flere markører • Man må trykke 'lagre markør' for at den valgte posisjonen skal lagres og kunne brukes når man laster appen på nytt • Lokasjonsdata kan være litt upresist
Karttegneskjermen	Samme som tegneskjermene, men her blir illustrasjonsbakgrunnen hentet fra skjermdumpen som ble tatt på kartskjermen. Muligheter til å tegne, legge til drabare elementer, viske og slette alt på skjermen.	• Ingen knapp for å gå tilbake til kartet (for utenom tilbake-knappen på enheten, eller hamburgermenyen). • Ingen mulighet til å bruke tidligere skjermdumper
Innstillinger	I innstillingene kan man sette globale verdier som også blir lagret lokalt på enheten. Disse innstillingene er: - Om det brukeren har tegnet skal fjernes når man skifter skjerm - Innledende farge på penn og drabare elementer - Viskelærstørrelse - Hvilke drabare elementer som er tilgjengelige på tegneskjermene	• Muligheten til å velge fargetema er fjernet • Litt få innstillingsmuligheter for brukeren
Om Appen	Informasjon om denne appen med to lenker, en til å sende e-post til utviklerne og en til Google Play Store for å kunne vurdere appen.	

TABELL 32 - BEGRENSNINGER PÅ SYSTEMET

3.2. MÅLOPPNÅELSE I FORHOLD TIL PROSJEKTPLANEN

Gantt-diagrammet under viser de planlagte sprintene, på grunn av gjentakende oppgaver i hver sprint er diagrammet for stort til å kunne vises i sin helhet. Vi viser til den vedlagte prosjektplanen – *Ferdig prosjektplan* – og project-fila for en mer detaljert oversikt.



Task Name	Duration	Start	Finish
Sprint 1	8 days	Wed 06.01.21	Fri 15.01.21
Sprint 2	10 days	Mon 18.01.21	Fri 29.01.21
Sprint 3	10 days	Mon 01.02.21	Fri 12.02.21
Sprint 4	10 days	Mon 15.02.21	Fri 26.02.21
Sprint 5	10 days	Mon 01.03.21	Fri 12.03.21
Sprint 6	10 days	Mon 15.03.21	Fri 26.03.21
Sprint 7	10 days	Mon 29.03.21	Fri 09.04.21
Sprint 8	10 days	Mon 12.04.21	Fri 23.04.21
Sprint 9	10 days	Mon 26.04.21	Fri 07.05.21
Sprint 10	9 days	Mon 10.05.21	Thu 20.05.21

FIGUR 17 - PROSJEKTPLANEN

Ettersom vi har valgt å jobbe ved bruk av Scrum-metodikk, var det veldig vanskelig å benytte MS Projects og Gantt for utviklingen av en prosjektplan. Det er rett og slett ikke laget for det. MS Project har en egen mal kalt Agile Project Management, men den er veldig utdatert og vi klarte ikke å finne noen opplæringsvideoer som tok for seg denne.

Vi har derfor laget et grovt estimat av fordelingen av dager per sprint på de respektive oppgavene. Dette estimatet er basert på antall timer vi har tildelt de forskjellige hovedoppgavene i en prosjektplan vi har laget i Excel. For å kunne benytte dette estimatet var vi nødt til å gjøre de om fra timer til dager, noe som igjen gjorde det mindre fleksibelt og upresist. Vi har derfor ikke benyttet og oppdatert Gantt-diagrammet underveis i prosjektet, men heller lagt mye arbeid ned i timelister, sprint-reviews, samt backloggen og timeføring i Jira. Vi har derimot fulgt prosjektplanen for når vi har avsluttet og startet de forskjellige oppgavene.

Oppgave	Planlagt oppstart	Planlagt avsluttet	Påbegynt	Avsluttet	Detaljer
Forstudierapport	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Overskred antall budsjetterte timer.
Kravdokument	Sprint 2	Sprint 9	Sprint 2	Sprint 10	Mesteparten ferdigstilt i sprint 6, små endringer i senere sprinter.
Alpha	Sprint 2	Sprint 4	Sprint 2	Sprint 4	Ferdig til planlagt tid.
MVP	Sprint 5	Sprint 10	Sprint 5	Sprint 10	Brukte færre timer enn det som var budsjettert.
Sluttrapport	Sprint 10	Sprint 10	Sprint 10	Sprint 10	Ferdig til planlagt tid.
Driftsrapport	Ikke planlagt	Ikke planlagt	Sprint 5	Sprint 10	Fikk originalt beskjed at vi ikke trengte driftsrapport, var derfor ikke planlagt.

TABELL 33 - PLANLAGTE MOMENTER

Da vi la ned planen for hele prosjektet lå kravene på 500 timer per person (pluss/minus 5%). Dette viste seg i ettertid å kun være et mål for å få studentene til å faktisk jobbe, og at faktisk fremgang

overstyrer dette kravet. I den virkelige prosjektplanen har noen av oppgavene overskredet den budsjetterte tiden allokert, men alt i alt ligger vi godt under det samlede estimerte timeantallet. Viser til *Tabell 34 – Timeliste* under Timeregnskap for en oversikt over estimerte timer mot faktiske timer brukt.

3.2.1. TIMEREGNSKAP

I tabellen under kan man se oversikt over timene som er brukt for dette prosjektet for hvert teammedlem. Oppgavene er delt inn og har et prefiks som er brukt for å holde orden på hvilken oppgave vi har arbeidet med underveis. Hele timeregnskapet finnes i Excel format ligger tilgjengelig som vedlegg.

		Teammedlem				
Prefiks	**Aktivitet**	**Silje**	**Oliver**	**Joakim**	**SUM**	**Budsjettert**
D	1. Forstudierapport	20	14	24	**58**	**50**
A	2. Prosjektadministrasjon	20.5	11	10.5	**38**	**70**
DE	3. Design av applikasjon	54	15	34	**103**	**90**
R	4. Research	33.5	33	41	**107.5**	**100**
KD	5. Kravdokument	13.5	3	6	**22.5**	**50**
K	6. Alpha	61	120	48.5	**229.5**	**220**
M	7. Møter	28.5	26.5	29.5	**84.5**	**95**
DR	8. Driftsdokument	63.5	25	60.5	**149**	**100**
T	9. Testing/forbedring	51	58	21	**130**	**100**
BT	10. Brukertesting	8	8.5	4.5	**21**	**30**
K	11. MVP	107.5	137	99	**343.5**	**530**
SD	12. Sluttrapport	18	24	19	**61**	**65**
	SUM TIMER PER PERS	**479**	**475**	**397.5**	**1347.5**	**1500**

TABELL 34 – TIMELISTE

3.2.2. FORHOLDET MELLOM PLANLAGT OG FAKTISK TIDSBRUK

Vi skal i denne delen drøfte hvordan timeplanleggingen gikk i praksis i forhold til hvordan det faktisk gikk ved å bruke tabellen over. Til høyre i tabellen ser vi antall timer som gruppen budsjetterte til hver oppgavetype og dette er basert på timene vi forventet å bruke på denne oppgaven i løpet av prosjekttiden. Den andre grå tabellen som viser SUM, viser sammenlagt timer fra alle teammedlemmer på en spesifikk arbeidsoppgave.

Som man kan se fra tabellen så har gruppen til en viss grad klart å budsjettere timene riktig i forhold til hva som faktisk har blitt brukt. De fleste oppgavene har en liten prosentandel som er litt over eller under budsjetterte antall timer, men den største forskjellen ligger nok på planlagt bruk på MVP mot faktiske timer brukt. Det skal da også sies at gruppen mangler sammenlagt cirka 100 timer som gjør at vi ikke nådde målet vårt på 1500 timer totalt. MVP (*Minimum viable product*) er versjonen av applikasjonen som blir lagt ut til produksjon i Google Play Store. Denne versjonen skal inneholde de viktigste funksjonene slik at applikasjonen kan brukes av brukere og ikke mangle noe som er viktig for kunden. Her ble det satt av hele 530 timer, men bare rundt 350 er faktisk brukt. Dette kommer nok av at gruppen har brukt litt flere timer enn planlagt på *Testing og Forbedring* som innebærer å fikse feiler eller mangler i nåværende versjon av appen. Den andre grunnen er at som nevnt over vi ikke har nådd alle planlagte 1500 timer til hele prosjektet og flere av disse ville nok ha blitt brukt på MVP-versjonen.

Selv om det er litt avvik fra planlagt tid på hver oppgave så betyr ikke det at det arbeidet som er utarbeidet under hver oppgave ikke har blitt ferdigstilt og grundig produsert, det betyr at gruppen har enten overvurdert eller undervurdert hvor lang tid disse oppgavene kom til å ta.

For å utligne dette så skulle det vært satt av flere timer til *forstudierapporten* og mindre til *kravdokumentet*. *Design av applikasjonen* skulle også vært økt til å ligge på rundt 100 timer og heller redusert antall budsjetterte timer på *prosjektadministrasjonen*. Det har vært stort fokus på å ikke bruke flere timer enn nødvendig på oppgaver som er fordelt til gruppemedlemmer, det er noe som har blitt diskutert flere ganger for å hjelpe hverandre å gjennomføre oppgavene innenfor de budsjetterte timene, men også for å hjelpe hverandre å ikke sitte fast med noe som andre kan hjelpe med. Det har vært stort fokus på å hjelpe hverandre og det har hjulpet veldig med å få fullført arbeidsoppgaver og gruppemoralen.

Selv om det er avvik fra planlagt til faktisk tidsbruk anser vi ikke denne som svært betydelig, men forventet. Det lille avviket vi ser kan forklares og læres av.

3.3. Måloppnåelse mot forstudierapporten og kravdokumentet

I denne delen skal vi gå gjennom problemstillingene som ble satt i forstudierapporten og UC-kravene fra kravdokumentet. Det skal drøftes over hvorvidt disse ble løst, delvis løst eller ikke løst. Drøftingen har blitt fremstilt ved hjelp av tabeller og litt tekst.

3.3.1. Oppnådde resultater ut ifra forstudierapporten

Målene og problemene som ble satt i forstudierapporten er her referert til som 'problemer'. Hvorvidt de har blitt løst er markert i status-kolonnen, og en kommentar forklarer hvordan eller hvorfor de er løst.

Problem	Status	Kommentar
Funksjonelle egenskaper		

Illustrering med penn og papir blir lite presist	Løst	Applikasjonen gjør at trafikklærere ikke lengre trenger bruker penn og papir, men kan bruke appen som verktøy i stedet.
Forskjeller i tegneferdigheter blant trafikklærere	Løst	Appen har ferdigillustrerte illustrasjoner av forskjellige veisituasjoner og trafikksituasjoner som er standardisert for å forenkle opplæringen av elevene. I tillegg er det mulighet til å kunne legge til elementer som representerer bilder, mennesker, trær osv. Slik at trafikklærere ikke må tegne disse selv.
Det kan ta lang tid å illustrere en situasjon med penn og papir slik at eleven forstår det.	Løst	illusTrafikk er laget for å gjøre illustreringene raskere, enklere og mer presist. Ved hjelp av enkle funksjoner på tegnesidene og ulike basisillustrasjoner å velge mellom, vil det redusere tiden som brukes av lærere for å vise til spesifikke situasjoner.
Papir som benyttes for å illustrere på, forsvinner	Løst	Ved å ha alt på en applikasjon, må man ikke tegne opp alle situasjonene på nytt igjen hver gang. Dette gjelder da selve 'bakgrunns-illustrasjonene' som viser veioppsettet osv.
Manglende system for illustrering av trafikksituasjoner	Løst	Det finnes nå et system for Android som kan benyttes av trafikklærere og andre brukere for å enkelt kunne illustrere trafikksituasjoner.
Ikke-funksjonelle egenskaper		
Brukervennlighet	Løst	Applikasjonen følger standarder for Android applikasjoner. Applikasjonen er ikke for komplisert og har klare knapper og valg som brukeren kan benytte seg av. Det har vært fokus på å ha et oversiktlig og enkelt oppsett under hele utviklingen.
Responstid	Løst	Alle bilder som kan være av mindre størrelse har blitt forminsket. Det er tatt hensyn til at bilder og illustreringer kan påvirke responstiden til applikasjonen og det er lagt inn funksjoner for å minke lastingen av disse. Antall tredjepartsbiblioteker har blitt redusert til det mest nødvendige og gruppen har fokusert på å utvikle egne komponenter der det har vært mulig og nødvendig for å ha bedre kontroll på funksjonaliteten.
Brukbarhet	Løst	Alle illustrasjonene er laget med tanke på standarder i trafikksituasjoner. Det er tatt hensyn til å ikke har for mange 'ledd' for å komme seg til en spesifikk side. Hver side kan enkelt navigeres til via enten

		hamburgermenyen tilgjengelig på alle sidene eller fra startsiden.

TABELL 35 - RESULTATER FRA FORSTUDIERAPPORTEN

3.3.2. RESULTATER SAMMENLIGNET MED KRAV SATT I KRAVDOKUMENTET

I denne delen går vi gjennom hvorvidt alle UC-kravene har blitt oppfylt i forhold til hvordan de ble satt i kravdokumentet. Use Casene blir presentert til venstre i tabellen, status som løst eller ikke løst i midten og en kommentar til hvorfor eller hvordan implementering har gått til høyre i tabellen.

Use Case	Status	Kommentar
UC1: Illustrere trafikksituasjon, veikryss	Gjennomført	**Endring**: «Trafikklærer drar bilvektor ut i veikonstruksjonen.» Endret til å trykke på istedenfor å dra. Unntak: Busslomme ikke tilgjengelig
UC 2: Illustrere trafikksituasjon, rundkjøring	Gjennomført	**Endring**: «Trafikklærer drar bilvektor ut i veikonstruksjonen.» Endret til å trykke på istedenfor å dra. Unntak: Busslomme ikke tilgjengelig
UC 3: Illustrere trafikksituasjon, vei	Gjennomført	**Endring**: «Trafikklærer drar bilvektor ut i veikonstruksjonen.» Endret til å trykke på istedenfor å dra. Unntak: Busslomme ikke tilgjengelig
UC4: Vise trafikkskilt og informasjon om skilt	Gjennomført	
UC5: Vise posisjon på Google Maps og tegne over kartet	Delvis gjennomført	**Endring**: Hovedflyt punkt 3 og 4 – «Trafikklærer trykker på knapp for å "fryse bildet"» og «applikasjonen viser tegnemenyen». Endret til at brukeren tar en skjermdump og kan senere velge å 'bruk skjermdump' for å kunne tegne på denne skjermdumpen.
UC6: Vise læreplanmål for klasse B	Gjennomført	
UC7: Vise nyttige lenker til viktige trafikkopplæringsforskrifter	Delvis gjennomført	**Endring**: Hovedflyt punkt 2. – «Bruker får opp en skjerm med lenker til forskjellige forskrifter og eventuelt andre nyttige nettlenker.» Endret til at brukeren ikke får opp noen skjerm med nyttige lenger, men alle nyttige lenker ligger tilgjengelige via startskjermen.
UC8: Vise myndighetspyramiden	Gjennomført	

UC9: Sette egne innstillinger for appen	Gjennomført	**Endring**: Hovedflyt punkt 5 – «En modal vil da åpne seg og bruker kan velge opptil 20 bilder/ikoner som kan brukes som drabare elementer på tegneskjermen» Endret til at brukeren kun får velge opptil 15 elementer for å minke problemer med lasting av for mange bilder/ikoner på sidene.

TABELL 36 - RESULTATER FRA KRAVDOKUMENTET

Oppsummert fra tabellen over, har alle UC-ene blitt implementert, men flere av de har små endringer. Noen av disse endringene er så minimale at de er fortsatt ansett som gjennomførte UC-krav, mens andre har endringer som endrer komponentene de beskriver og er derfor markert som '*Delvis gjennomført*'. I løpet av prosjektet har det vært fokus på at disse Use-Casene ble implementert da de inneholder den mest fundamentale funksjonaliteten til applikasjonen. Endringer som er blitt gjort underveis er enten gjort på grunnlag av at vi har funnet en bedre eller mer intuitiv måte å utforme de på, eller på bakgrunn av tilbakemeldinger fra kunden og testere.

3.4. ANNET SOM KAN DOKUMENTERE PROSESSEN

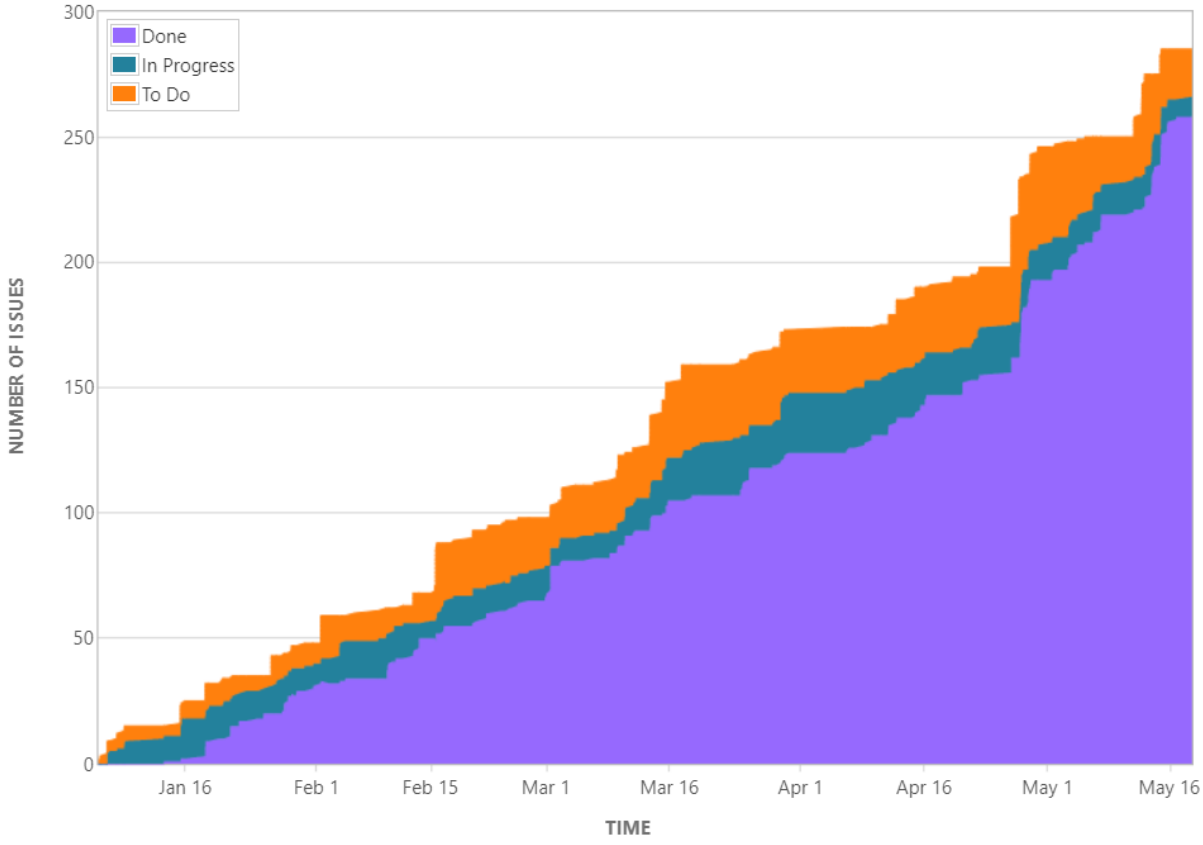
Når det gjelder annet som kan dokumentere prosessen, skal vi i denne delen ta for oss verktøy som er brukt for å forenkle utviklingsprosessen og hvordan de er brukt til å holde prosessen oversiktlig. Det første og viktigste verktøyet er Jira. Deretter skal vi snakke litt om GitHub og til slutt vise litt skjermdumper fra en tidligere versjon, som skal vise til endringene som er blitt gjort rundt design valg.

3.4.1. JIRA

For dette prosjektet valgte prosjektgruppen å ta i bruk et produksjonsverktøy som heter Jira. Jira er laget for å hjelpe utviklingsgrupper å lage, sortere, dokumentere og holde styr på alle oppgaver og underoppgaver som må gjøres. Jira kan enkelt settes opp til å følge en Scrum-metodikk med sprinter og en *backlog.* På starten av hver sprint gikk gruppen gjennom alle oppgavene som skulle gjennomføres i løpet av to uker og fordelte dem på deltakere. Det ble brukt et ekstra verktøy, direkte integrert i Jira, til å holde oversikt over timene som ble registrert. Dette blir forklart i neste kapittel.

Jira settes opp som et '*board*' hvor man har oversikt over ikke-påbegynte, påbegynte og fullførte oppgaver for hver sprint som kjøres. Oppgavene kan flyttes mellom de forskjellige kategoriene ettersom statusen på de endres. I grafen under kan vi se hvordan dette har pågått fra starten av januar til midten av mai. Under hele prosessen har det vært opprettet nye oppgaver som må gjøres, for så å fullføre disse. Totalt har det vært opprettet om lag 300 oppgaver som har blitt laget og

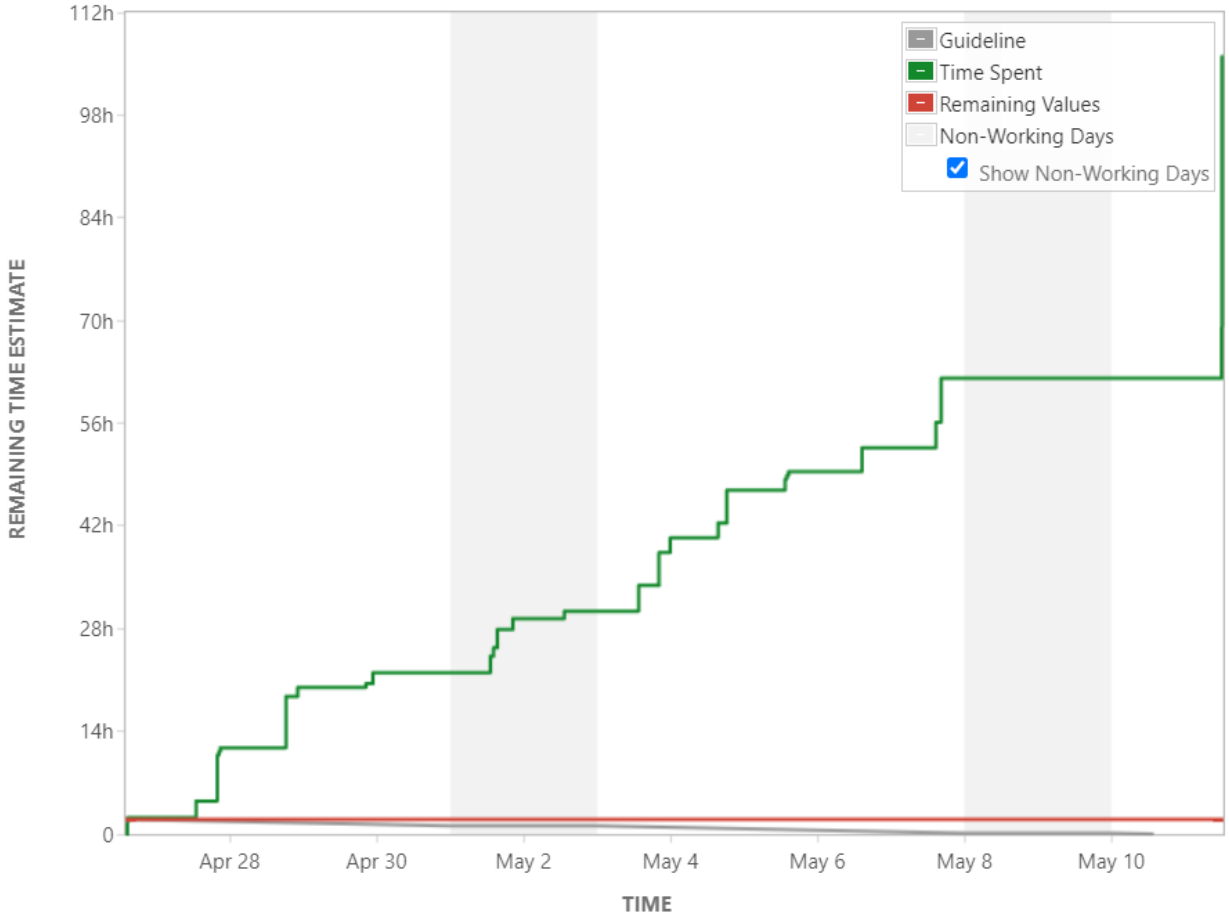
fullført fortløpende for annenhver uke (hver sprint).



FIGUR 18 – ALLE FULLFØRTE, PÅBEGYNTE OG FERDIGE OPPGAVER I JIRA, I LØPET AV HELE PROSJEKTPERIODEN

Som man kan se ut ifra grafen har gruppen klart å holde en stødig vekst i antall oppgaver som har blitt opprettet og fullført. Det er noen oppgaver som ikke er registrert som fullførte da disse er oppgaver som har pågått gjennom hele prosjektet. Disse oppgavene har vært laget for å kunne legge inn timer uten at de settes som fullført. Eksempler på slike oppgaver er møter, tester, arbeid rundt forbedring og kommentering, og prosjektadministrasjon. Ved å ha disse oppgavene liggende i «In Progress»-kolonnen vil de ikke bli fjernet mellom hver sprint og det har da gjort det enkelt for oss å bare legge inn timer på disse oppgavene for å bruke senere i timeliste-tabellen vår. Timene som registreres blir også lagt inn i det ekstra verktøyet som er brukt for timeregistrering som viser når timene var lagt inn (dato og klokkeslett).

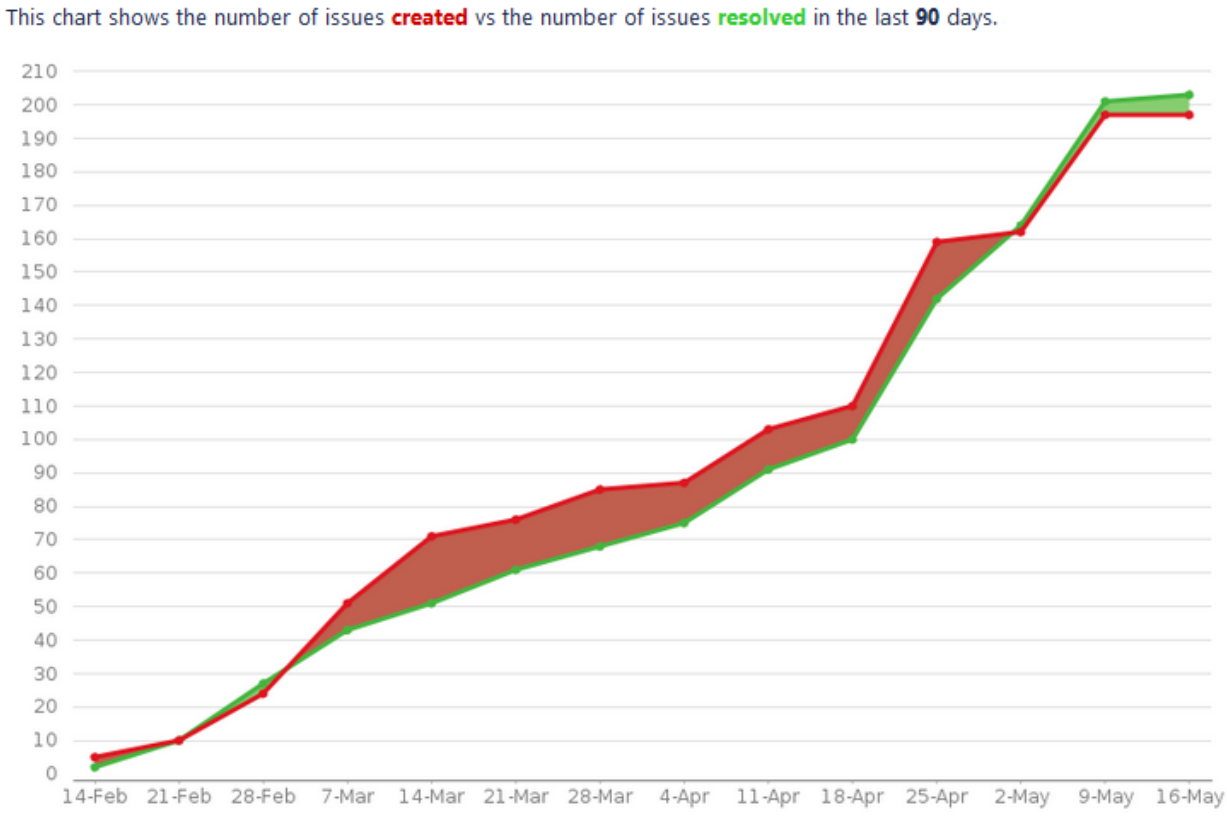
I Jira har man også muligheten til å kunne se hver uke hvordan det har gått med sprinten i forhold til antall timer brukt. Dette vises ikke i dette dokumentet da det ikke går å få en 'sammenlagt' versjon av det, men vi viser i figuren under et eksempel for Sprint 9.



FIGUR 19 - TIMER BRUKT FOR SPRINT 9

Vi har ikke registrert antall timer forventet på hver time, men heller fokusert på hvor mange timer som faktisk er brukt, derfor er den røde linjen alltid null (som viser antall timer gjenstående) og den grønne linjen går oppover (som viser antall timer brukt).

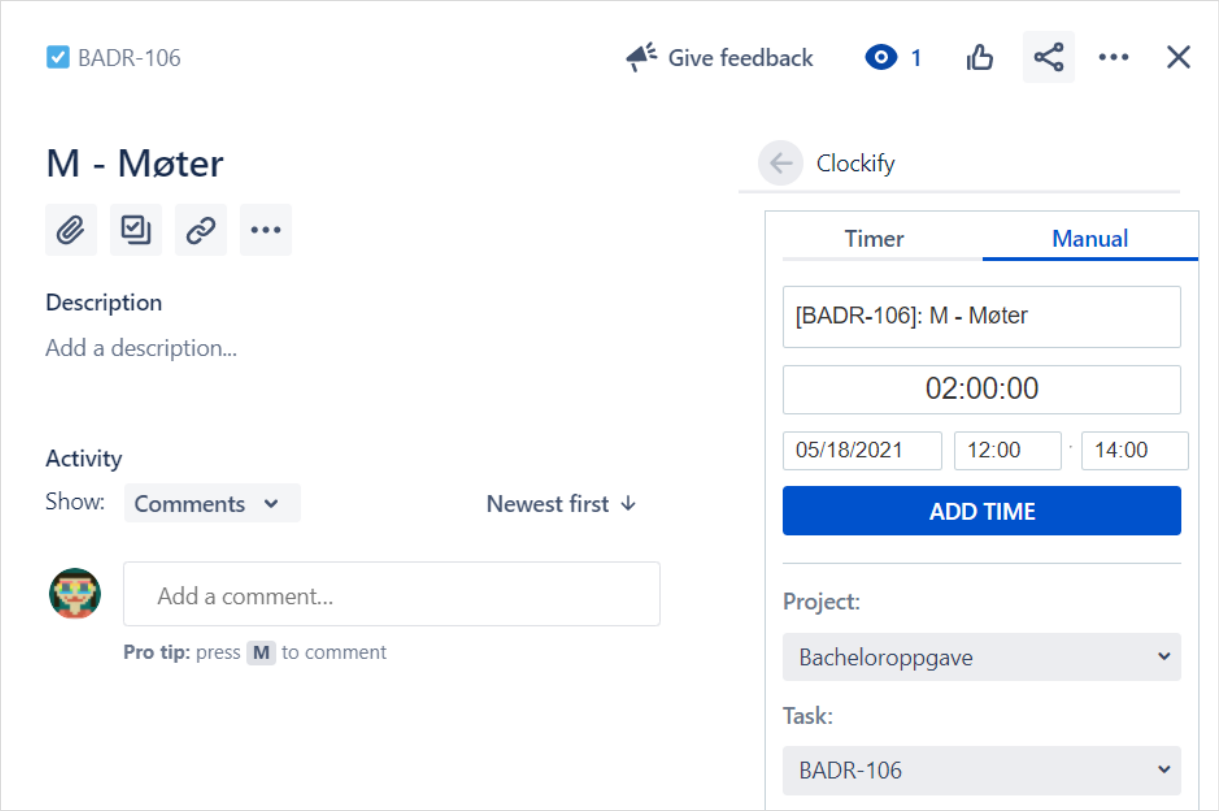
Det siste som kan vises i Jira, er antall oppgaver opprettet vs. antall fullførte oppgaver. I tabellen under, ser vi i løpet av de siste 90 dagene at oppgaver har blitt laget hyppig og også fullført enten på samme sprint eller sprinten etter.



FIGUR 20 - JIRA OPPGAVER STARTET VS. FULLFØRT

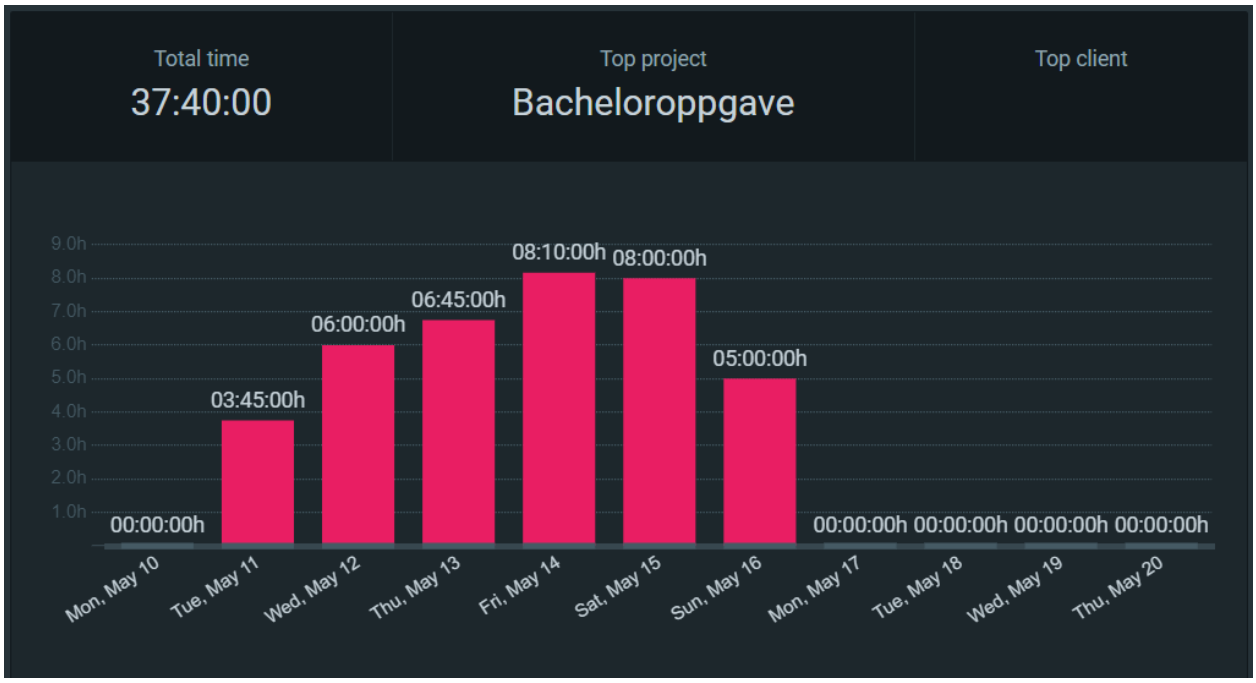
3.4.2. CLOCKIFY

Clockify er navnet på tilleggsverktøyet vi har benyttet for å kunne føre timer direkte på oppgaver i Jira. Skjermbildet under viser hvordan vi fra en spesifikk oppgave enkelt kan skrive inn start- og sluttidspunkt for oppgaven. Navnet på oppgaven og timene brukt på den, blir deretter sendt til Clockify.



FIGUR 21 - FØRING AV TIMER FRA JIRA MED CLOCKIFY

Alle i prosjektgruppen har måttet vedlikeholde sin egen Clockify, som er koblet direkte mot brukeren i Jira. Fra Clockify sitt webgrensesnitt kan man blant annet se statistikk over timer brukt per dag over en valgt periode. Figuren under viser et av teammedlemmene sitt timebruk fra perioden 10. mai til 20. mai.



FIGUR 22 - STATISTIKK OVER TIMER I CLOCKIFY

Fra Clockify kan man også eksportere timebruken til Excel, som også er grunnen til at vi har benyttet en *prefiks* for alle oppgaver i Jira. I det første skjermbildet, Figur 21, ser vi at oppgaven for møter er navngitt som "M – Møter". Dette er gjort for å gjøre det enklere å regne sammen timer brukt per hovedaktivitet fra det eksporterte Excel-arket med en formel.

	A	B	C	D
1	**Project**	**Description**	**Time (h)**	**Time (decimal)**
2	**Bacheloroppgave**		**37:40:00**	**37.67**
3		[BADR-19]: A - Prosjektadministra	02:15:00	2.25
4		[BADR-128]: K - Koding (småfiksi	03:45:00	3.75
5		[BADR-215]: K - Legg til applikasjo	03:10:00	3.17
6		[BADR-253]: DR - Div dokumenta	02:00:00	2.00
7		[BADR-119]: DR - Driftsrapport	03:35:00	3.58
8		[BADR-106]: M - Møter	00:30:00	0.50
9		[BADR-252]: TB - Diverse testing	06:15:00	6.25
10		[BADR-218]: DR - JSDoc: Lese o	07:00:00	7.00
11		[BADR-294]: SD - Sluttrapport	09:10:00	9.17
12	**Total (05/10/2021 - 05/20/2021)**		**37:40:00**	**37.67**

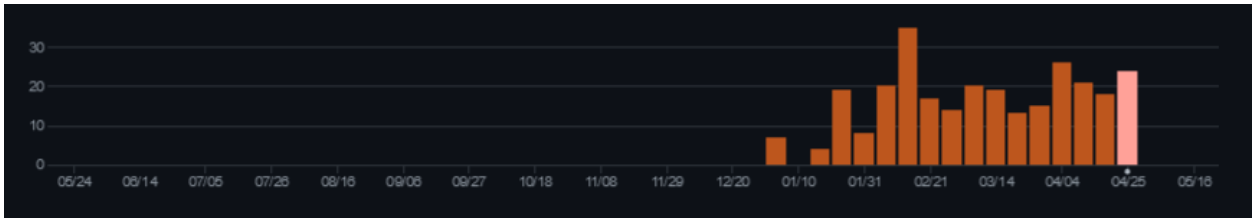
FIGUR 23 - EKSPORTERT OVERSIKT OVER ALLE TIMER FØRT

Formelen vi har benyttet summerer timene fra time-kolonnen ut ifra prefiksen for oppgaven. Eksempelet under summerer timene for alle oppgaver som har prefiks M:

```
SUMIF($B$3:$B$11, "*]: M -*", $D$3:$D$11)
```

3.4.3. GITHUB

GitHub har blitt brukt for å holde oversikt over selve koden. GitHub gir også en god oversikt over antall *'commits'* eller antall ganger ny kode har blitt lagt til. I oversikten under kan vi se at det har fra starten av vært nye commits hver måned. Hyppigheten av commits har vært ganske jevn og det vises i tabellen under.



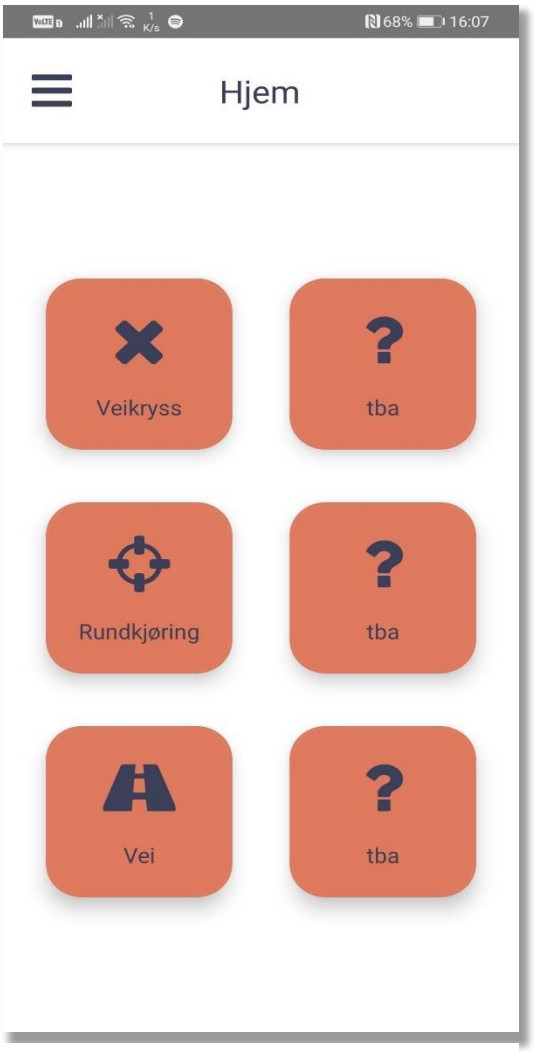
FIGUR 24 - GITHUB COMMITS

Dessverre viser ikke GitHub antall commits gjort over hele prosjektet for mer enn 90 dager, men ifølge GitHub har det blitt gjort 104 commits de siste 90 dagene.

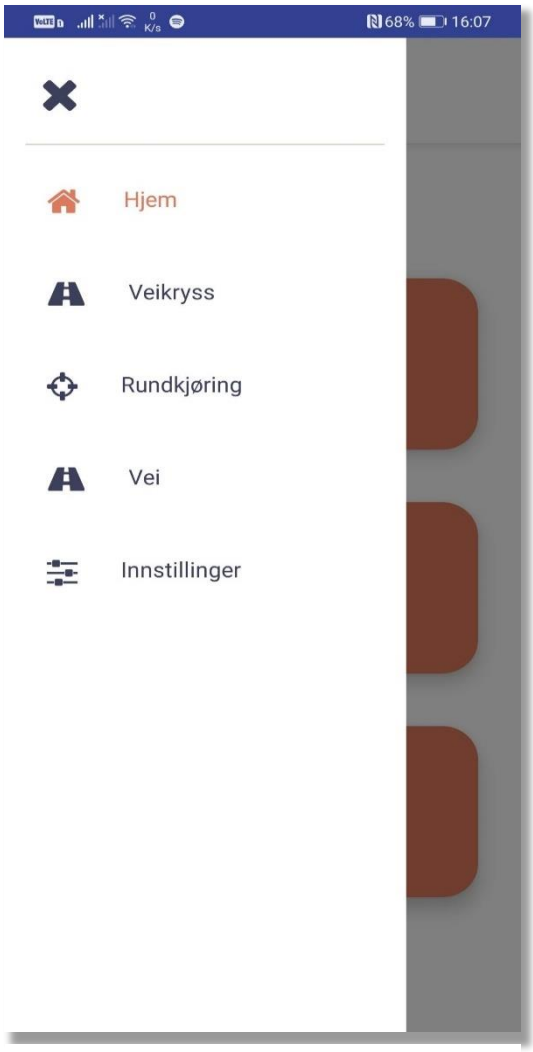
GitHub er en fantastisk plattform for å kunne samarbeide med kode på. Det er enkelt å se hvilke endringer som er blitt gjort, hvem som har gjort de og når. Vi har hatt full oversikt over alle de forskjellige utgreningene fra kodebasen og derfor også kunne rettet opp i feil som har oppstått underveis. GitHub gir oss også full tilgang til koden, muligheter til å enkelt kunne flytte mellom grener og muligheten til å enkelt kunne gi tilgang til oppdatert kode fra hverandre.

3.4.4. SKJERMDUMPER AV TIDLIGERE VERSJONER

Det ble tatt noen skjermdumper som viser en tidligere versjon av applikasjonen. Disse skjermdumpene representerer de første 2 til 3 sprintene av arbeidet vi utførte. Det er ikke lagt til mye funksjonalitet enda, men fokuset har vært på det viktigste akkurat da: tegneskjermen/illustrasjonsskjermen og en navigasjonsside.



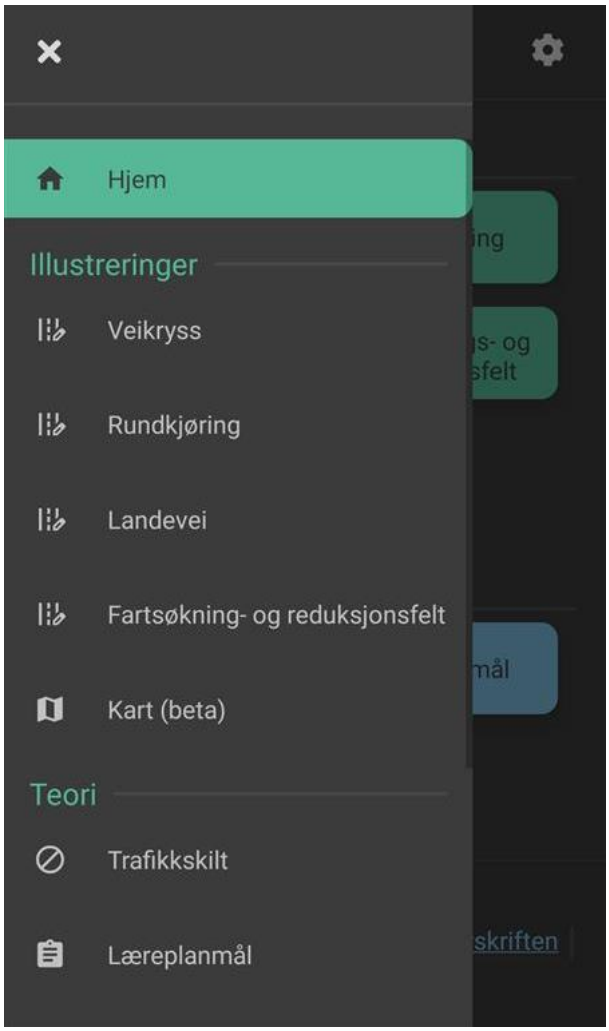
FIGUR 26 - TIDLIGERE VERSJON AV STARTSIDEN



FIGUR 25 - TIDLIGERE VERSJON AV HAMBURGERMENY

Som kan sees på skjermdumpene over ser vi den gamle versjonen. Her fulgte vi de originale wireframene som ble laget, og som kan ses i *Kravdokumentet*. Ut ifra nåværende versjon, som vi ser under, ser vi spesielt at fargene har fått en endring, utformingen av startsiden ble utvidet til å inneholde mer og å dele opp innholdet i seksjoner. Det samme gjelder hamburgermenyen.

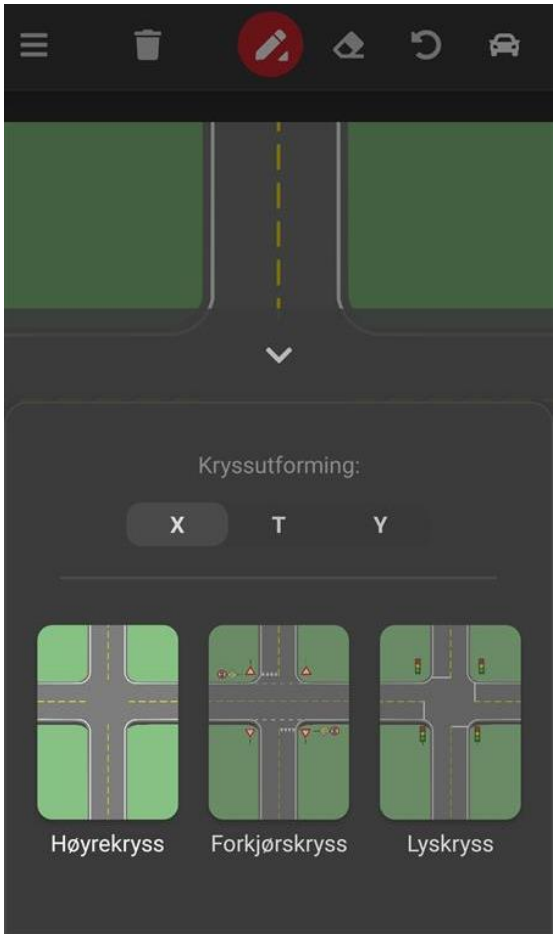
FIGUR 28 - NÅVÆRENDE VERSJON AV STARTSIDEN



FIGUR 27 - NÅVÆRENDE VERSJON AV HAMBURGERMENYEN

I figur 13- nåværende versjonen, og figur 14– tidligere versjon, ser vi noen av de største forandringene. Tegneskjermen som disse to bildene viser, ble implementert først og det ble lagt til en bunnmeny tidlig for å kunne navigere de forskjellige type illustrasjoner. Denne menyen har vært i kontinuerlig endring for å forbedre brukeropplevelsen. Originalt så var denne menyen en tekstbasert meny som senere ble endret til å ha bilder og lignende til å representere de ulike valgene.

FIGUR 29 - TIDLIGERE VERSJON TEGNESKJERMEN



FIGUR 30 - NÅVÆRENDE VERSJON TEGNESKJERMEN

Ved å se på nåværende versjon mot den gamle kan man blant annet se arbeidet som er gjort rundt design, men også hvor mye mer funksjonalitet som er blitt lagt til underveis. Nåværende versjon ser mer utfyllende ut, har finere animasjoner (kan ikke sees på skjermdumpene) og fargene er mer samstemt.

Den tidligere versjonen hadde i utgangspunkt mye mer fokus på funksjonalitet, når vi så dette begynte å komme seg, så begynte fokuser å drives mot design i form av UX (User Experience) og UI (User Interface).

4. VIDERE ARBEID

Applikasjonen er lansert på Google Play, som er den offisielle distribusjonsplattformen for applikasjoner for enheter med Android operativsystem. Applikasjonen er lagt ut for gratis nedlastning. Selv om vi i løpet av denne prosjektperioden har implementert de viktigste funksjonene i applikasjonen, og satt den i produksjon, er det likevel en del tiltak som kan gjøres videre. Dette går på å optimalisere enkelte funksjoner, optimalisere applikasjonen for flere enheter enn nettbrett, og implementere mindre viktige funksjoner som ble nedprioritert i prosjektperioden.

I tillegg er det sannsynlig at det vil kunne dukke opp ting med appen som ikke fungerer optimalt, når flere personer laster den ned og bruker den over tid. Selv om vi har testet applikasjonen grundig både under utvikling, før og etter lansering, vil man aldri klare å fange opp alt. Dette er derfor også et punkt som bør med videre, der man bør følge opp tilbakemeldinger fra brukerne og rette opp i ting som ikke fungerer slik de skal.

Et annet punkt det kan gjøres noe med videre, er betaling for applikasjonen. Vi har som sagt lagt den ut for gratis nedlastning, men hvis den blir lastet ned og brukt av mange, som vi jo håper vil skje, er dette med betaling noe som må vurderes for å kunne klare å vedlikeholde applikasjonen på en god nok måte. Det tar tid å rette opp i feil, implementere nye funksjoner og holde applikasjonen oppdatert, noe som kan føre til at vi kan bli nødt til å ta betalt for å laste ned applikasjonen. Det kan også være en mulighet å selge applikasjonen til noen som driver med applikasjonsutvikling.

Oppsummert kan følgende gjøres videre:

- Optimalisere kartfunksjonen i appen. Denne er foreløpig lagt inn som en betafunksjon, noe som vil si at den fungerer, men vil kunne ha enkelte bugs.
- Optimalisere applikasjonen for små enheter, dvs. mobiler. Applikasjonen er laget for å brukes på nettbrett, og den fungerer også på mobil, men den er ikke helt optimalisert.
- Implementere de kravspesifikasjonene vi ikke rakk å implementere. Dette inkluderer blant annet flere illustrasjoner å tegne på. Her er det en mulighet å få inn noen med kompetanse innenfor illustrasjon til å lage flere bakgrunner. Oversikt over alle kravspesifikasjonene til applikasjonen, inkludert de vi ikke har implementert, er tilgjengelig i Kravdokumentet.
- Gå gjennom og sørge for at applikasjonen oppfyller kravene til universell utforming (WCAG 2.0). Applikasjoner som ikke trenger internettilgang for å fungere, er ikke pålagt å følge kravene til universell utforming, men vi ønsker likevel at den i framtiden skal oppfylle disse kravene.
- Følge med på om det kommer eventuelle tilbakemeldinger fra brukere, om funksjoner og andre ting i appen som ikke fungerer best mulig, og deretter rette opp i dette.
- Følge med på når det kommer oppdateringer for Android, React, React Native og de tredjepartspakkene vi har benyttet i applikasjonen, og oppdatere applikasjonen i henhold til disse.
- Vurdere om vi skal ta oss betalt for applikasjonen, eventuelt se på om vi kan selge den til noen som driver med applikasjonsutvikling og er interessert i å videreutvikle appen.

5. REFERANSER

[1] M. Schwarzmüller, «The Practical Guide to React Native,» Udemy.com, 2020. [Internett]. Available: https://www.udemy.com/course/react-native-the-practical-guide/. [Funnet 27 Desember 2020].

[2] Facebook, Inc, «Introduction · React Native,» Facebook, Inc, 18 Mars 2021. [Internett]. Available: https://reactnative.dev/docs/getting-started. [Funnet 6 Januar 2021].

[3] S. L. Henry, «w3.org,» W3C, 29 April 2021. [Internett]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/. [Funnet Mai 2021].


NTNU
Kunnskap for en bedre verden