

# SKETCHWORLD: AR

Creative augmented reality app for iOS

Av

Snorre Aas & Lars Haugen Gjelle

Bacheloroppgave ved Oslo Metropolitan University  
23. Mai 2019

**Institutt for Informasjonsteknologi**

Postadresse: Postboks 4 St. Olavs plass, 0130  
Oslo

Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00

PROSJEKT NR.

46

Tilgjengelighet

Offentlig

# BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
SketchWorld: AR	23/5/2019
	ANTALL SIDER / BILAG
	36
PROSJEKTDeltakere	INTERN VEILEDER
Snorre Aas	Frode Eika Sandnes
Lars Haugen Gjelle	

OPPDRAUGSGIVER	KONTAKTPERSON
Frode Eika Sandnes	Frode Eika Sandnes

SAMMENDRAG
I dette prosjektet har vi hatt som mål å designe og utviklet en tegneapp for utvidet virkelighet. Vi har tatt i bruk moderne teknologiske løsninger, og verktøy. Dette har resultert i en iOS applikasjon.

3 STIKKORD
Apputvikling for iOS
Swift
Augmented Reality

# Innholdsfortegnelse

Innholdsfortegnelse.....	2
1. Innledning.....	3
1.1 Om prosjektet .....	3
1.2 Dagens situasjon .....	3
1.3 Mål .....	3
1.4 Målgruppe .....	4
2. Prosessdokumentasjon .....	5
2.1 Introduksjon.....	6
2.2 Planleggingsfase .....	6
2.3 Arbeidsmetoder .....	7
2.4 Prototype .....	9
2.5 Utviklingsfase.....	10
2.6 Sluttfase.....	14
3. Produktdokumentasjon.....	18
3.1 Introduksjon.....	19
3.2 Brukergrensesnitt.....	19
3.3 Tilgangstillatelse.....	25
3.4 Systemarkitektur.....	26
4. Refleksjoner og læringsutbytte .....	28
4.1 Introduksjon.....	29
4.2 Refleksjon rundt arbeidsmetoder .....	29
4.3 Produkttiterasjoner.....	29
4.4 Resultatet.....	29
4.5 Potensiell videreutvikling.....	30
5 Litteratur .....	32
6 Vedlegg.....	33
Vedlegg 1: Oppgaveteksten .....	33
Vedlegg 2: Skjermbilder fra prototype .....	34
Vedlegg 3: Skjermbilder av produktet.....	35
Vedlegg 4: Kopirettigheter.....	36

## 1. Innledning

Denne rapporten dokumenterer et bachelorprosjekt utført ved OsloMet våren 2019. Prosjektet er gjennomført av Snorre Aas og Lars Haugen Gjelle.

### 1.1 Om prosjektet

Oppgaven vår gikk ut på å utvikle en mobil applikasjon som lar flere brukere lage skisser og tegne sammen i utvidet virkelighet. Vi valgte å jobbe med dette da vi begge ønsket å få erfaring innenfor apputvikling og fremtidsorientert teknologi.

I dette prosjektet har vi designet og utviklet en mobilapplikasjon som lar flere personer lage tegninger på smarttelefoner i fellesskap. Dette har vi gjennomført ved hjelp av Apple sitt utvidet virkelighets-bibliotek ARKit 2, og vi har da utviklet eksklusivt for iOS enheter.

Det vi ønsket å utforske med vårt prosjekt er et aspekt av utvidet virkelighet som ikke har blitt utforsket til sitt fulle potensiale, nemlig kreativitet og tegning i fellesskap. Vi hadde som mål å bevise at dette er et mulig konsept å gjennomføre, og har i denne rapporten dokumentert vår arbeidsprosess og produktet vi har utviklet.

### 1.2 Dagens situasjon

Teknologien innenfor utvidet virkelighet - på engelsk *Augmented Reality* eller AR, har vært under rask utvikling de siste årene. AR har raskt inntatt det kommersielle markedet, da muligheten til å benytte denne teknologien kommer innebygd i en høy prosentandel av nyproduserte smarttelefoner.

Utvidet virkelighet er fortsatt i tidlig utviklingsfase, og det er først nå de siste årene det har kommet programvare og maskinvare med støtte for AR, selv om konseptet med å kombinere fysisk virkelighet med den digitale verden ble utforsket allerede sent på 60-tallet av Sutherland et al. (1968).

Progresjonen har vært eksponensiell de siste ti årene grunnet direkte korrelasjon til utvikling av raskere, mindre, og mer mobil maskinvare. Dette har ført til utvikling av blant annet AR-kompatible enheter i form av briller, slik som Google sitt *Google Glass* og Microsoft sin *HoloLens*. Med høy konkurranse blant store aktører ser fremtiden til AR interessant ut, og vil innen kort tid ta større plass i det kommersielle markedet.

### 1.3 Mål

Målene vi satte for oss selv i forhold til selve prosjektet var:

- Å produsere en produksjonsklar applikasjon til iOS
- Implementere tegnefunksjonalitet i AR
- Ha en form for flerspillermodus

Som tilleggsmål vil vi at applikasjonen skal ha potensiale for å kunne videreutvikles til en sosial plattform.

#### 1.4 Målgruppe

En studie gjort av Thrive Analytics (2018) påstår at AR-teknologi benyttes mest av personer i alderen 25-34. Av den grunn kan det være naturlig for oss å ha denne aldersgruppen som målgruppe, men vi har et ønske om at folk i alle aldre vil kunne synes at applikasjonen vår er et interessant konsept.

## 2. Prosesdokumentasjon

---

Sketchworld: AR

*Bachelorprosjekt 2019*

## 2.1 Introduksjon

I dette kapitlet er prosjektets utviklingsfase dokumentert. Dette inkluderer å fortelle hvordan arbeidsprosessen er blitt gjennomført, og å gå i dybden på verktøy vi har benyttet.

## 2.2 Planleggingsfase

### 2.2.1 Valg av teknologi

Første del av prosjektet handlet mye om å utforske muligheter og gjøre oss kjent med hvilke alternativer vi hadde for hvilken retning vi ønsket at prosjektet skulle ta. Det er per dags dato stor konkurranse innenfor utvikling av AR-rammeverk, og alle ønsker å være de første til å implementere nye funksjoner. En av de ledende aktørene er teknologi giganten Apple, og de var med ARKit 2 de første som tilrettela for deling av objekter og posisjoner slik at man kan implementere flerspillermodus på en sømløs måte. Vi har valgt å benytte oss av deres rammeverk grunnet at Apple har integrert teknologien tett med smarttelefonene deres, som de illustrerer godt gjennom tjenester sånn som f.eks. FaceID og Animoji. De jobber aktivt for å tilrettelegge for fremtiden innenfor utvidet virkelighet.

### 2.2.2 Teknologiene vi har benyttet oss av

Teknologier	Beskrivelse
Swift	Objektorientert programmeringsspråk som benyttes i utvikling av iOS applikasjoner.
SceneKit	Rammeverk som genererer tredimensjonal grafikk som kan plasseres og manipuleres
UIKit	Rammeverk som setter opp side hierarki for fremvisning av brukergrensesnitt
ARKit	Rammeverk som inneholder funksjoner for utplassering av tredimensjonale objekter i utvidet virkelighet.
MapKit	Rammeverk som kan vise kart og satellittbilder. Det benytter seg av <i>Google Mobile Maps</i> tjenesten for å utlevere kartdata.

### 2.2.3 Tilgang til Mac

Tidlig i planleggingsfasen møtte vi på en komplikasjon som vi var nødt til å finne en løsning på, nemlig at vi kún hadde tilgang til én Mac. Dette var et problem da man faktisk trenger en fysisk Mac maskin når man skal utvikle en applikasjon til ARKit. Det er to grunner til dette, den ene er at Xcode kún er tilgjengelig på macOS, som er det optimale utviklerverktøyet for iOS, og den andre grunnen er at man må være fysisk tilkoblet en Mac med en USB kabel for å kunne testkjøre ARKit applikasjoner på en smarttelefon.

Her måtte vi utforske alternativer, og så på ulike måter å skulle kunne fordele tilgangen til Macen. Vi kom frem til en meget god løsning der vi opprettet to separerte kontoer på Macen, installerte en fjerntilgangsapplikasjon ved navn Google Remote Desktop, og lar denne applikasjonen kjøre til en hver tid på en av kontoene. Dette ga oss muligheten til å benytte oss av samme Mac samtidig.

## 2.3 Arbeidsmetoder

For vårt prosjekt ønsket vi å ha jevn progresjon gjennom hele utviklingsfasen, og benytte oss av arbeidsmetoder og planleggingsverktøy som tilrettela for dette. Vi har tidligere erfaringer med å bruke github og discord som kommunikasjonsverktøy for gruppeprosjekter, og har derfor valgt å ta disse i bruk under utviklingen av denne applikasjonen.

Vi konfigurerte et privat kodelager eller *repository* på GitHub der mesteparten av vår utviklingsaktivitet gikk gjennom. GitHub tilbyr flere organiseringsverktøy for prosjektarbeid og vi tok i bruk blant annet deres Kanban Board. Ved å ta i bruk Github og Kanban Board satte vi opp en god grunnstruktur for å kunne jobbe effektivt over nettet.

Kanban Board kan brukes til å organisere oppgaver og delegere ansvar. I praksis vil dette si at prosjektmedlemmer kan effektivt påbegynne en gitt oppgave og implisitt informere resten av gruppen om hva de jobber med, og når de er ferdig med en enkelt oppgave. Brettet er delt opp i kolonner som hver inneholder oppgaver med forskjellig progresjonsstatus slik som; "Uferdig", "Under arbeid" eller "Fullført". Innenfor disse kolonnene kan man prioritere oppgaver, delegere hvem som skal gjøre hver enkelt oppgave og flytte elementer mellom de forskjellige kolonnene. En praktisk egenskap med å benytte GitHub sitt Kanban Board er at det oppfører seg dynamisk, og vil under utviklingen flytte oppgaver automatisk rundt på brettet.

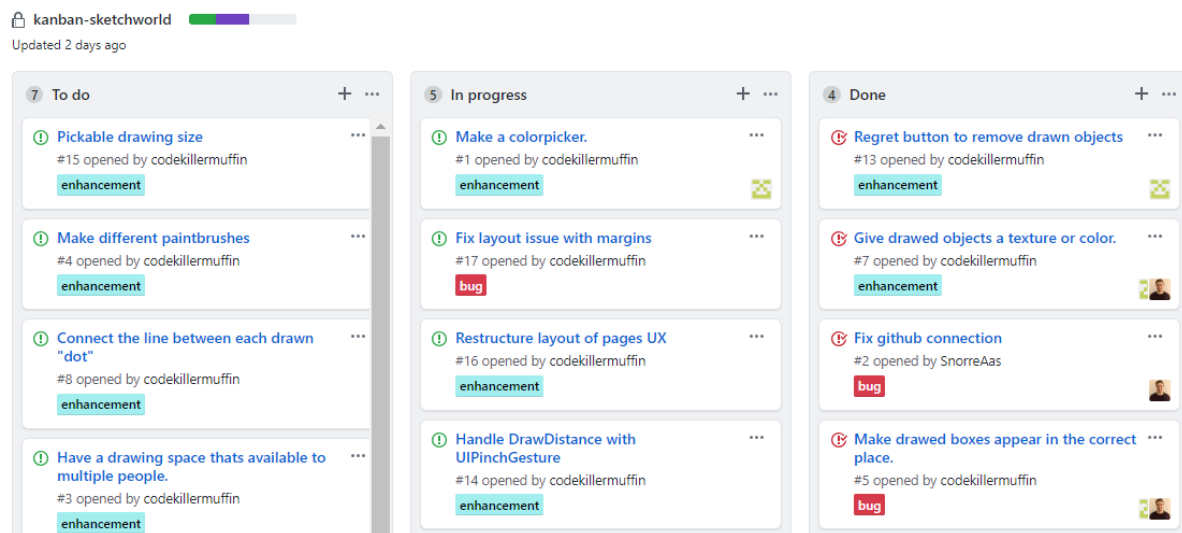


FIGURE 1 SKJERMBILDE FRA GITHUB SITT KANBAN BOARD

For videosamtaler, gruppemøter og kode sprinter tok vi i bruk programmet Discord. Hovedgrunnen til at vi valgte Discord som kommunikasjonsplattform er at programmet støtter skjermdeling i høy oppløsning og den er en bidragsyter til at møter blir effektive og kan gjennomføres uten komplikasjoner. GitHub og Discord har god integrasjon med hverandre, og vi konfigurerte en kobling mellom plattformene hvor Discord fungerte som et varslingssystem. Dette fungerer på den måten at Discord lytter på utviklingsaktivitet gjennom GitHub, og vil varsle alle gruppe medlemmene når det er aktivitet.



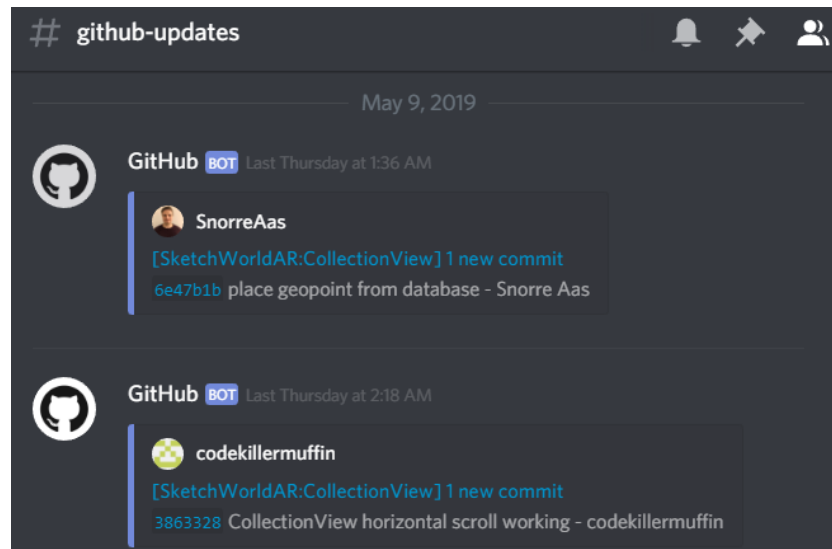


FIGURE 2, UTKLIPP FRA DISCORD KANALEN DER VI IMPLEMENTERTE ET VARSLINGSSYSTEM

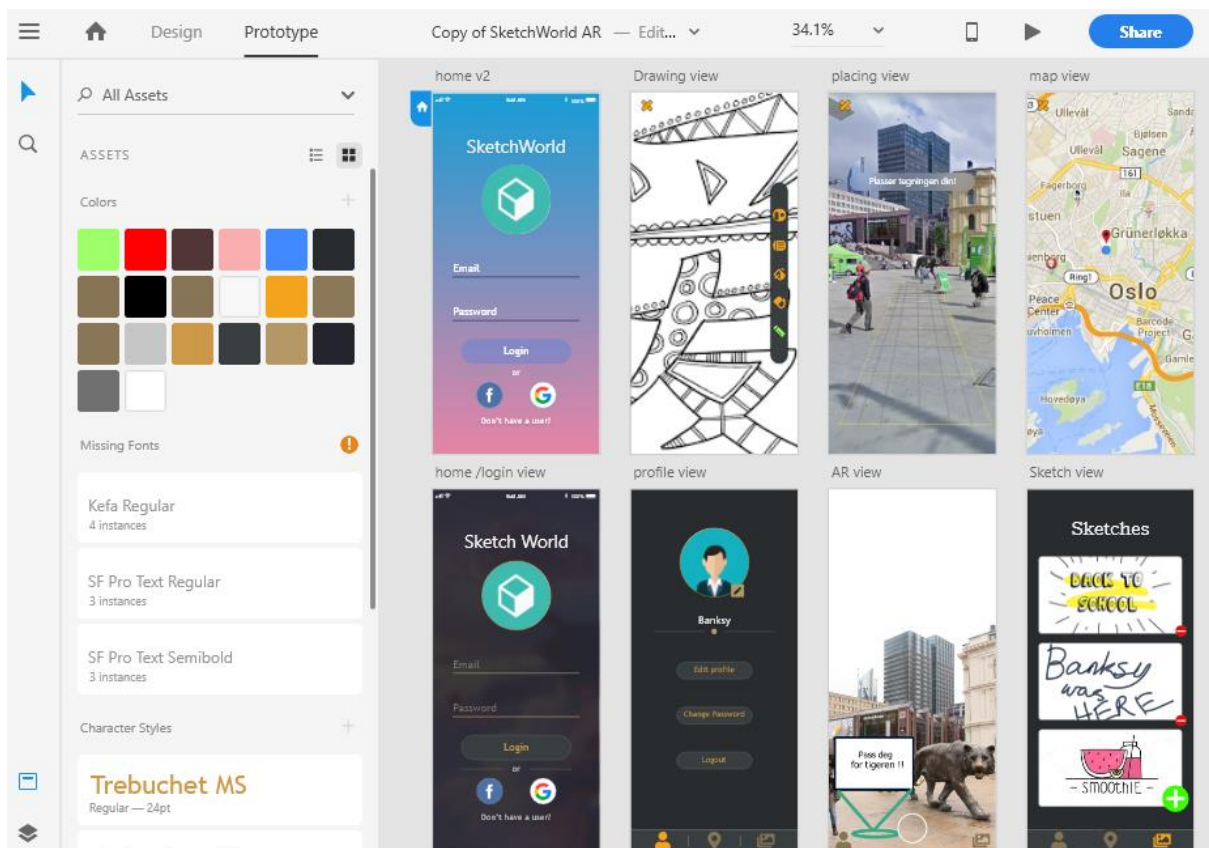
### 2.3.1 Verktøy

I løpet av utviklingsprosessen har vi benyttet oss av flere verktøy for utvikling, design og kommunikasjon. I tabell 1 er en liste over alle verktøyene vi har benyttet oss av.

Verktøy	Beskrivelse
	Adobe XD Designverktøy fra Adobe som kan benyttes til å lage interaktive prototyper med live links som kan trykkes på og som kan kjøres på mobilenheter.
	Xcode Xcode er Apple sitt innebygde utviklingsmiljø. Dette medfører at man har feilretting, autofullføring av kodesnutter og debugging. Programmet har også en simulator som kan kjøre apper.
	GitHub GitHub er en plattform mest kjent for versjonskontroll, men de har også flere prosjekthjelpende funksjoner, blant annet et kanban board og kommentarfelt.
	Discord Discord er et videosamtale program som også har støtte for gruppesamtaler, tekstsamtaler og mye mer.
	Google Docs Google Docs er et samarbeidsverktøy der man kan skrive tekst samtidig i samme dokument.
	Firebase Firebase er et type databasesystem som Google har utviklet. Det har god tilkobling med apputvikling. De oppretter automatisk et web-api som kan benyttes.
	Postman Postman er et verktøy som kan benyttes til å gjøre spørringer etter data fra nettsider.

## 2.4 Prototype

Før vi startet utviklingsfasen valgte vi å produsere en prototype. Vi benyttet oss av verktøyet Adobe XD for å designe et konseptuelt brukergrensesnitt og en brukerflyt passende til applikasjonen vi skulle utvikle. Programmet har god støtte for utforming og design av enkeltkomponenter, og alle attributter til hvert element loggføres, noe som tilrettela for overgangen mellom prototype og produkt. Vi brukte denne fremgangsmåten for å gi oss selv og oppdragsgiver oversikt over hvordan et potensielt sluttprodukt vil ta form. Med denne prototypen som utgangspunkt kunne vi diskutere valg av ulike funksjoner, og komponenter, og bruke dette for å få et bedre utgangspunkt når vi skulle utvikle applikasjonen.



**FIGURE 3, SKJERMUTKLIPP AV ADOBE XD PROTOTYPEN**

Produksjon av en tidlig prototype bidro til idémyldringen, og vi fant ut at tegnefunksjonen vi hadde planlagt var suboptimal. Den opprinnelige planen var å lage en tegnefunksjon som skulle gi brukeren muligheten til å plassere et AR-objekt i form av et virtuelt skilt som han/hun kunne fylle med tekst/ grafikk.

Originalt var planen å bruke "AR Sketchpad" som prosjektnavn, noe som implisitt refererer til en applikasjon som tilbyr en todimensjonal tegnefunksjon. Når vi inspiserte prototypen, kom vi fram til at dette navnevalget ikke stemmer overens med hva vi ønsket å eksperimentere med i dette prosjektet. Vi valgte derfor å oppdatere prosjektnavnet til et mer representativt navn, SketchWorld AR. Målet vårt var å gi brukerne muligheten til å tegne i et tredimensjonalt rom, og derfor er dette et mer passende navn.

## 2.5 Utviklingsfase

Xcode er Apple sitt utviklermiljø for programmering til iOS. Programmet gir utvikleren mulighet til å designe utseende og funksjonalitet til en applikasjon, via et symbiotisk forhold mellom kodeark og *Storyboard*, som er en visuell representasjon av applikasjonen sine sider og elementer. Storyboard har “dra og slipp” funksjonalitet hvor en utvikler kan enkelt plassere elementer, og konstruere skjelettet til en applikasjon effektivt. Ideen er at man skal kunne programmere visuelt og se endringer i mer enn bare tekstbasert kode.

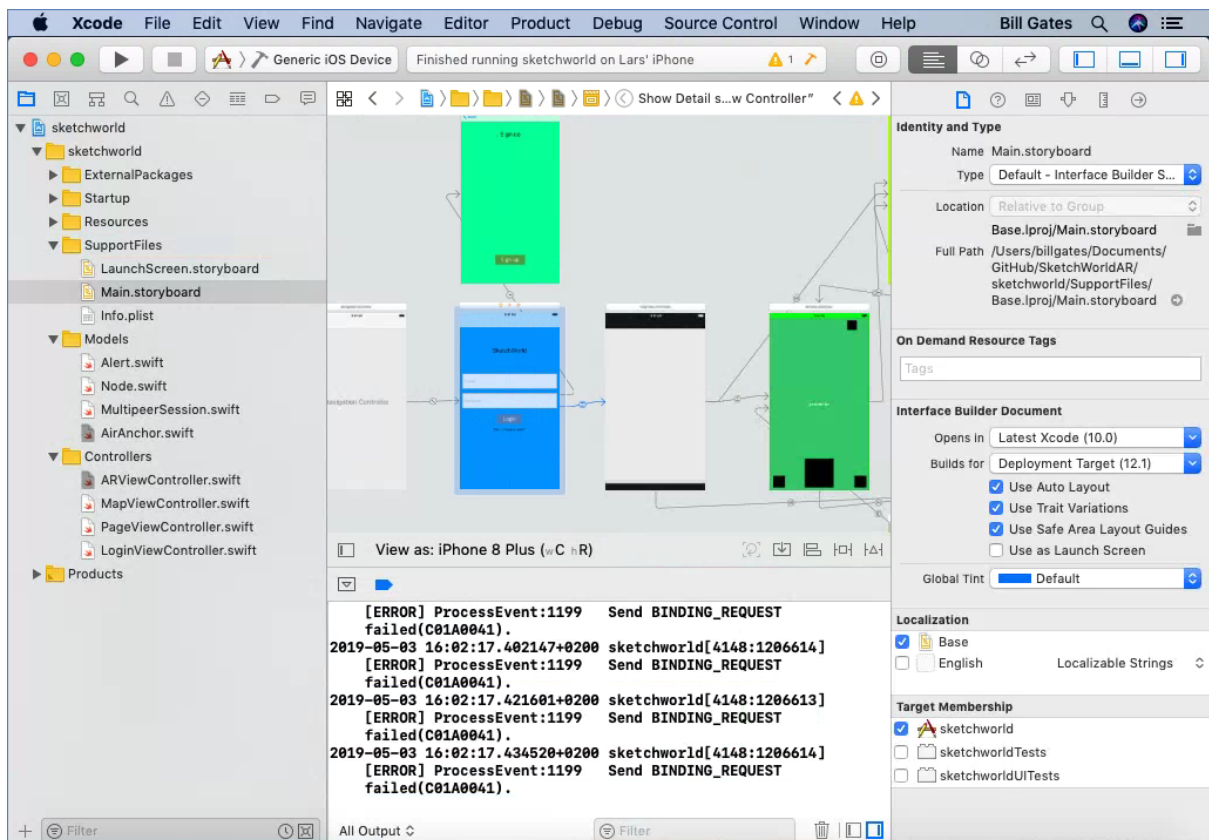


FIGURE 4, SKERMBILDE AV XCODE

I første del av utviklingsfasen arbeidet vi med å utforske basisløsninger for prototypen vår, der vi hadde et representativt sideoppsett og knapper som var tilkoblet funksjoner i et relatert kodeark. Vi startet utviklingen slik fordi det bygger et godt grunnlag der vi enklere kan videreutvikle applikasjonen, i tillegg til at denne perioden tillot oss å bli kjent med brukergrensesnittet til Xcode og kodestrukturen til Swift, samt forskjellene deres med andre verktøy og kodespråk vi hadde benyttet oss av tidligere.

Vi valgte å strukturere prosjektmappen slik at den var oversiktlig og tilpasset for påbygging. Senere i utviklingsfasen vil applikasjonen inneholde mange filer og risikere å fremstå ulesbar, og endringer i mappestrukturen vil kunne føre til feil i referanser som kan føre til komplikasjoner.

### 2.5.1 CocoaPods

For å håndtere eksterne utvidelser, *packages*, har vi tatt i bruk CocoaPods. Disse utvidelsene har forskjellige funksjoner og vi benytter oss av de i utvikling av applikasjonen. I tabell 2.1.3 er en liste over de utvidelsene vi har tatt i bruk.

Ekstern utvidelse	Beskrivelse
ColorPicker	Utvidelse som inneholder en skyveknapp eller <i>slider</i> , som man kan plassere og bruke for å velge en farge i et spektrum.
FireBase	Rammeverk som håndterer seg av autentisering og database koblinger.
SquareRegion	Utvidelse som brukes for å markere områder på et kart, for å unngå at folk skal være aktive ved f.eks. kulturhistoriske eller and sensitive lokasjoner.
MaterialDesign	Inneholder mange GUI elementer som kommer med god visuell styling og funksjoner.
NameThatColor	Finner et passende tekstbasert navn til en RGB-basert farge. Basert på et javascript bibliotek fra Chirag Mehta,

### 2.5.2 ARSceneView

For at applikasjonen raskt skulle gå fra å være en prototype til et fungerende produkt, samt kartlegge om prosjektet var gjennomførbart i praksis valgte vi å prioritere AR-implementasjonen da dette er det viktigste aspektet ved prosjektet.

Det første vi utforsket var funksjoner som krevde lite utviklingstid for å implementere, slik som å plassere enkle tredimensjonale objekter i scenen. Her måtte vi se nærmere på geometrien og dimensjonene i ARKit 2, for å finne ut hvordan plassering av objekter faktisk fungerte.

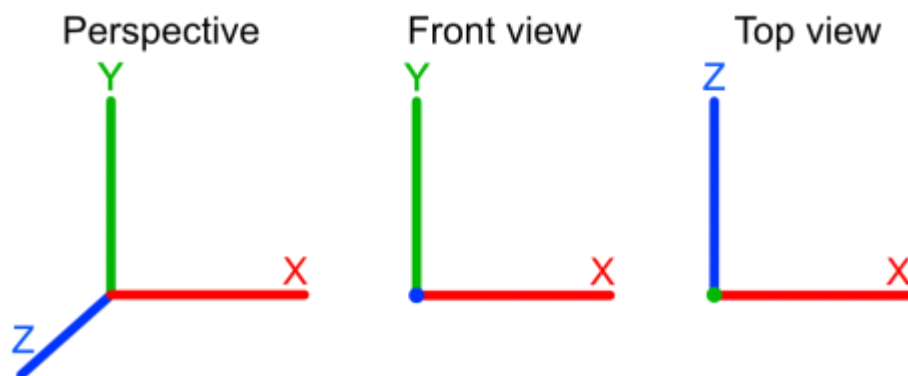


FIGURE 5, ILLUSTRASJONER AV TREDIMENSJONALE AKSER. (BEGBIE. 2017)

3D-objekter responderer på tre forskjellige akser; x, y og z. Ved å hente posisjonsdata fra mobilenheten, og endre enkelte parameter, vil vi kunne plassere objekter relativt til brukeren. På denne måten vil vi kunne gi korrekt plassering av objekter relativt til aksene sett illustrert i figur x. For eksempel hvis det settes en negativ z-index vil grafikken man plasserer bli forskjøvet langs z-aksen,

### 2.5.3 Grafikken som blir tegnet

SceneKit biblioteket inneholder flere forskjellige tredimensjonale strukturer, hvor den mest aktuelle geometriske formen til vårt forbruk var en sfære eller kule. Dette er grunnet vi ville unngå skarpe kanter for å skape flyt når brukeren tegner.

Disse geometriske formene blir lagret i et `LinkedListNode` objekt, som kommer fra en klasse vi har definert selv. Vi valgte å definere klassen slik at den har blant annet pekere til forrige og neste node. Vi satte det opp slik grunnet at noden skulle beholde relasjoner og at den skulle kunne grupperes.

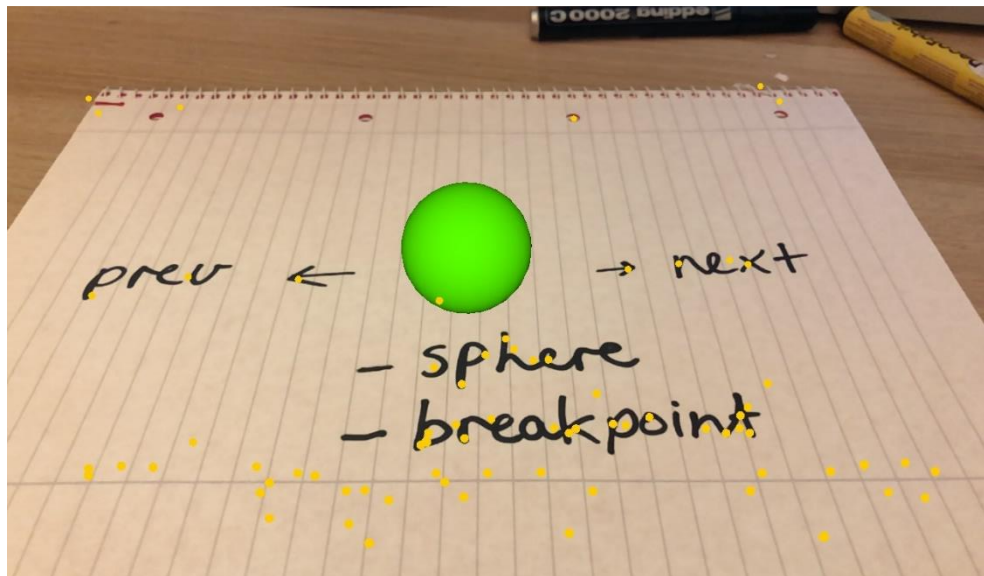


FIGURE 6 , HVORDAN HVER NODE BLIR LAGRET I TELEFONMINNET

Når hver node har blitt lagret med pekere til forrige og neste node, så er datastrukturen en dobbeltlenket liste. Nodene lagres på denne måten spesifikt for at angreknappen skal fungere på optimal måte. For å definere en sammenheng mellom noder implementerte vi et breakpoint system for å fortelle programmet hvor streker har begynnelse og slutt, for eksempel hvis man tegner en sammenhengende strek, så er det logisk at hele streken blir fjernet når man trykker på angreknappen.

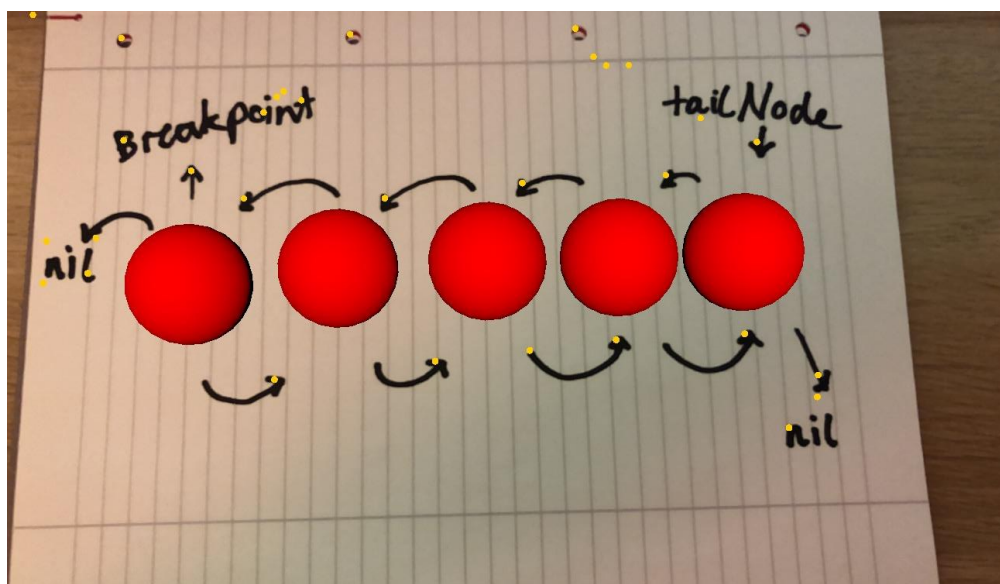


FIGURE 7, DATASTRUKTUREN I VÅR SAMMENKOBLEDE LISTE



## 2.5.4 ARWorldMap-klassen

ARWorldMap-klassen ble lansert som en av de nye utvidelse i ARKit 2 når rammeverket kom ut i midten av 2018. Vi ønsket å ta i bruk denne klassen for vår applikasjon da det åpner for blant annet deling av posisjoner mellom hverandre, noe som indirekte gjør at man kan dele tegninger. Klassen er en samling av punkter brukt i sammenheng med kartlegging av verdenen rundt kameraet, i tillegg til at det inneholder alle ARAnchor-objekter som er plassert i scenen. ARAnchor klassen inneholder egentlig bare en 4x4 matrise som markerer posisjon og rotasjon, og det er en rendering funksjon som har mulighet til å plassere grafikk på den posisjonen som ligger i matrisen.

For å implementere denne klassen var vi først nødt til å bytte posisjoneringssystem fra en plassering som var illustrert med en x, y og z-akse til det nye matrise oppsettet. Vi løste den geometriske differansen ved å implementere en konverteringsfunksjon som oversatte mellom matrisene. Vi valgte å sette denne funksjonen i en utvidelse eller *extension* som gjør at den er tilgjengelig på alle SCNVector3-objekter, da vi benytter oss av denne oversettelsen flere plasser videre i koden.

$$P = [x \quad y \quad z] \rightarrow \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## 2.5.5 Gjenoppstart fra tidligere økt

Lagring og gjenoppsett av ARWorldMap etter gjenoppstart av programmet kalles for *Persistence*. Det vil si at man har mulighet til å komme tilbake på et senere tidspunkt og fortsette fra samme punkt programmet ble avsluttet på. Vi kodet det slik at ARWorldMap og alle ARAnchor-objektene det inneholder blir lagret lokalt på en telefon når appen lukkes, og hentes opp igjen når man åpner appen igjen -> via `saveWorldMap()` funksjonen og `configuration.initialWorldMap`

```
71 // Create a session configuration
72 let configuration = ARWorldTrackingConfiguration()
73 configuration.planeDetection = .horizontal
74 configuration.initialWorldMap = self.loadWorldMap() ?? nil
75 sceneView.session.run(configuration)
```

FIGURE 8, KODEAVSNITT SOM VISER INITIALISERING AV EN ARTRACKINGCONFIGURATION

## 2.5.6 Rekonstruksjon av struktur

Grunnet et ønske om å ha en statisk navigasjonsløsning på bunnen av skjermen endte vi opp med å bytte fra individuelle `UIViewController`s(scener) til en `UICollectionViewController`(struktur med samling av celler). `UICollectionView` kan brukes for å lagdele elementer, slik at man

kan oppnå en statiske effekt på utvalgte elementer. Vi gjorde dette programmatisk kontra via storyboard grunnet Storyboard ikke har støtte for overlapping av scener.

```
55 override func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
    UICollectionViewCell {
56     let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellId, for: indexPath)
57
58     if indexPath.item == 0 {
59         let controller = storyboard!.instantiateViewController(withIdentifier: "MenuViewController")
60         addChild(controller)
61         cell.addSubview(controller.view)
62     }
63     // commented out other index paths for illustration purposes
64     return cell
65 }
```

FIGURE 9, ILLUSTRASJON AV KODE SOM PLASSERER EN SCENE I EN CELLE

Deretter implementerte vi en navigasjonsbar på bunnen av skjermen ved hjelp av en importert utvidelse ved navn *Material Design*. Denne utvidelsen gir oss et standard design, og et godt utgangspunkt for oppsett av funksjoner. For å personalisere navigasjonen, programmerte vi indikasjonsanimasjoner på navigasjonselementene.

## 2.6 Sluttfase

Avslutningsvis måtte vi ta et par valg når det kom til hvilke funksjoner vi ville prioritere. Her skjedde det mest finpussing som styling på komponenter og programmatisk feilretting, men det var også utvikling av essensielle funksjoner som flerspillermodus som gjenstod å utvikle. Vi hadde en del ideer vi var nødt til å droppe for å få tid til de funksjonene vi tenkte var de viktigste.

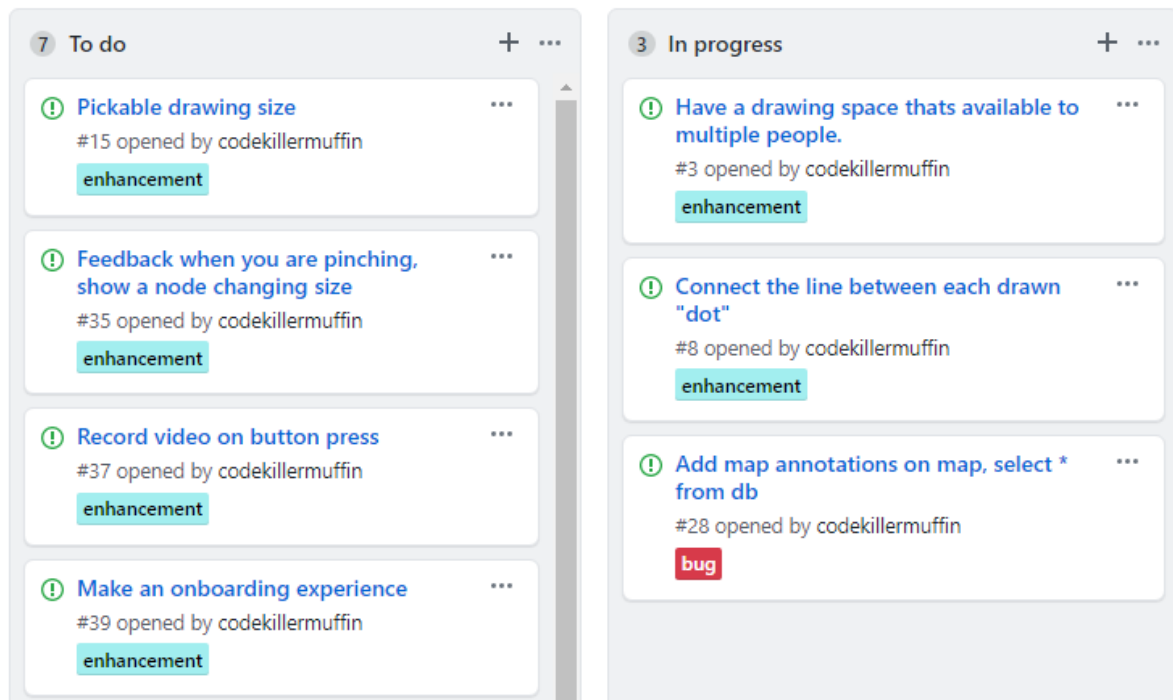
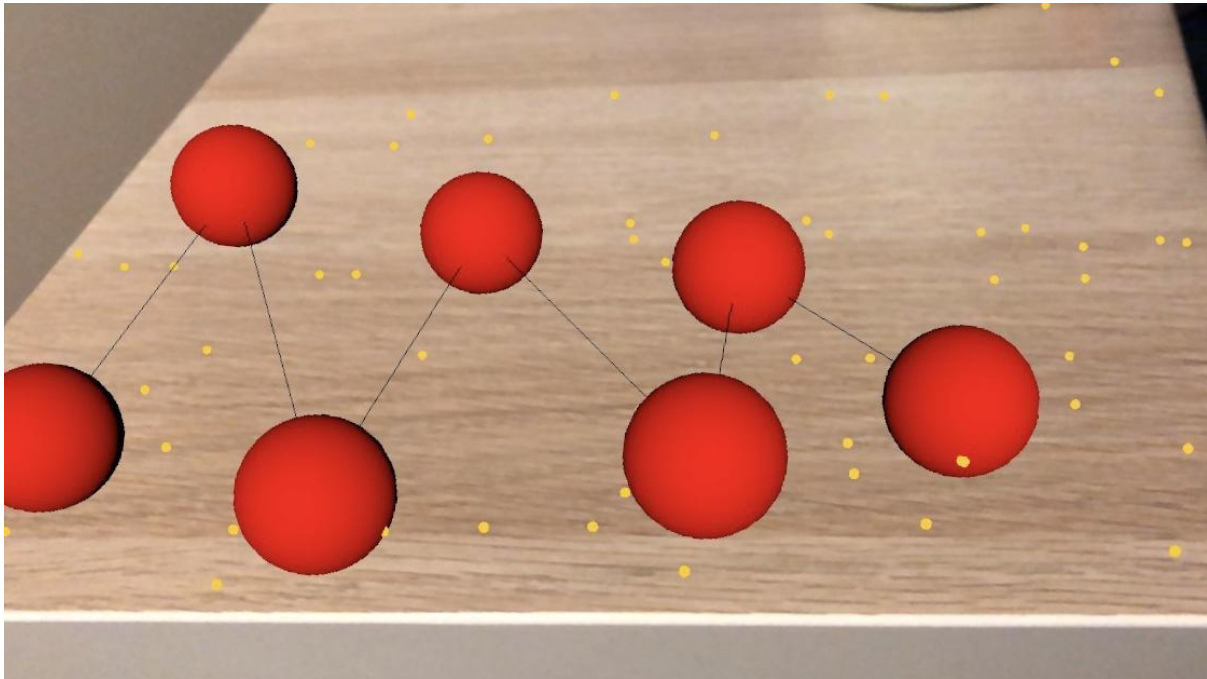


FIGURE 10, PRIORITERINGSLISTE MED KANBAN BOARD PER 2. MAI 2019

### 2.6.1 Visuelt feste noder til hverandre

En oppgradering vi prioriterte var å linke nodene i tegne funksjonen for å oppnå ubrutt tegning. Først forsøkte vi å gjøre dette med en SCNGeometry-klassen sin linjegravikk som fungerte fint og det var lett å plassere linjene på korrekt lokasjon i den virtuelle tredimensjonale verdenen

vår, da måten man implementerer linjen på er å koble start posisjonen og sluttposisjonen på to noder, og linjen vil da plassere seg midt mellom. Den oppnådde ikke alt vi ønsket da linjen ikke hadde noe tykkelse eller farge på seg, eksempel kan ses i figur x.



**FIGURE 11, EKSEMPEL PÅ LINJE GJENNOM SCNGEOMETRY**

Neste steg var da å prøve å implementere sylindere i stedet for linjene, ved hjelp av `SCNCylinder`. Ved å plassere sylindrene på samme node som sfæren den hører sammen med vil de få samme nullpunkt på samme plass, der `ARAnchor`-objektet er plassert. Her måtte vi posisjonere og rotere sylindrene slik at de la seg på korrekt måte. For å gjøre dette måtte vi lage en utvidelse for å regne ut gjennomsnitt avstanden mellom to noder, slik at vi kunne plassere cylinder objekter midt mellom nodene. Deretter var vi nødt til å rotere sylindrene, og vinkle de slik at de var vendt mot nodene før og etter.

```
12 extension SCNVector3 {  
13     func average(vector: SCNVector3) -> SCNVector3 {  
14         return SCNVector3((x + vector.x) / 2, (y + vector.y) / 2, (z + vector.z) / 2)  
15     }  
16 }
```

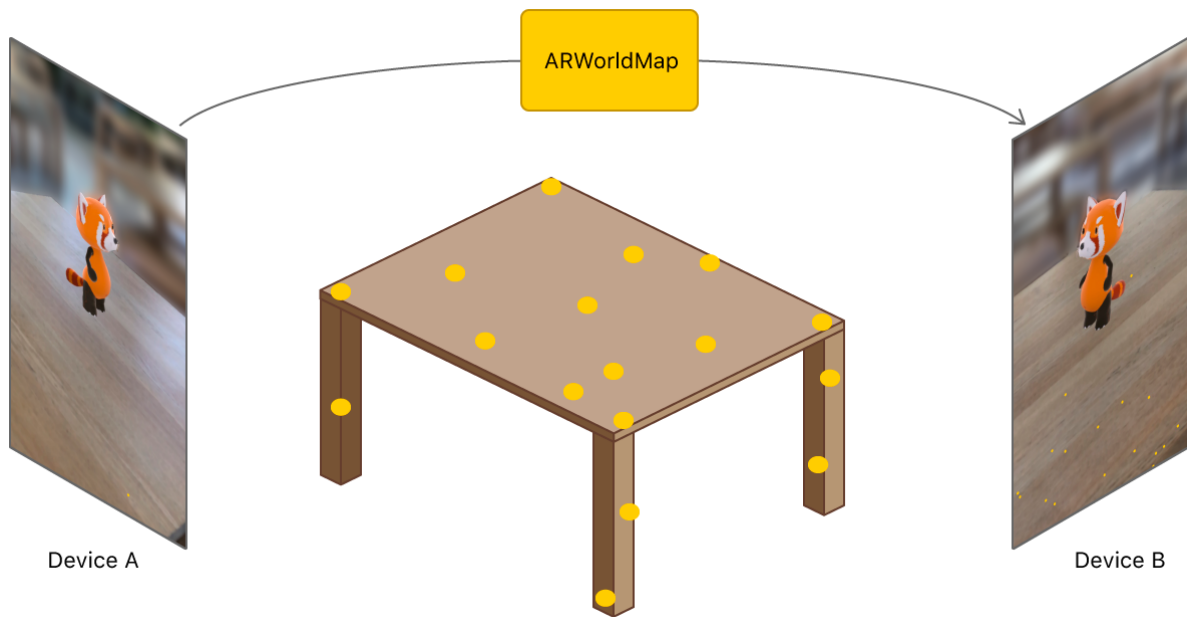
**FIGURE 12, KODESNUTT SOM FINNER MIDTPUNKTET MELLOM TO POSISJONER**

Et problem vi møtte under denne prosessen var at noder ikke kunne bli posisjonert korrekt, og ble låst til nullpunktsposisjonen til ankeret. Dette var grunnet at lokasjonen til en kule eller cylinder er lagret lokalt i forhold til `ARAnchor` objektet de er festet til, og den lokasjonen har verdi (0, 0, 0). Så når vi skulle plassere en cylinder i midten av noder og tok gjennomsnittet av to slike lokasjoner for å få midtpunktet fikk vi fortsatt et nullpunkt som svar.



Vi løste problemet ved å benytte posisjonene til selve ARAnchor objektene i stedet for sfærene de hører sammen med.

#### Flerspillermodus



**FIGURE 13, EKSEMPEL PÅ DELING AV ET ARWORLDMAP-OBJEKT (APPLE, 2018C)**

Oppsett av fungerende flerspillermodus krevde mye tid, og det var noe vi ble stående fast på lenge på under implementasjon. Dessverre er det lite dokumentasjon rundt dette temaet, grunnet teknologien er relativt ung, og det lille vi fant av dokumentasjon var ikke komplett, eller fungerte dårlig. For å feilsøke problemet metodisk slik at vi kunne kartlegge hvor i programmet koden feilet, plukket vi fra hverandre objektene vi ville sende, slik at det bare var ARAnchor-objekter tilhørende navn og en posisjons attributter, som til vår lettelse ga gode resultater. Dette gjorde at vi oppdaget at vi var nødt til å sette unike attributter og funksjoner på våre tilpassede ARAnchor klasser for å kunne komprimere de for overføring mellom mobilenheter. Når vi da hadde implementert disse endringene løste vi også et problem vi møtte på innenfor persistence, da kompilatoren nå klarte å kryptere dataene på korrekt måte, og lagre de i telefonminnet.

På dette punktet hadde vi en fungerende versjon av flerspillermodusen, men den var avhengig at spillerne var posisjonert på samme punkt, og rotert i samme retning, for at tegningene man laget skulle plassere seg på korrekt koordinater i forhold til hverandre.

En løsning til dette var å ikke bare sende posisjonene til tegningene, men også hele verdenskartet. Dette kartet inneholder informasjon om verdenen rundt, og hvis det deles med en medspiller blir alt posisjonert på rett plass fordi man har samme utgangspunkt.

Først lages en tilkobling mellom to eller flere brukere, ARWorldMap sendes, og posisjonen samkjøres.

#### Komprimering

ARAnchor-objektene blir lagret i ARWorldMap, deretter blir ARWorldMap komprimert og sendt via peer-to-peer gjennom bluetooth eller wifi tilkobling. ARWorldMap blir mottatt av andre brukere, og dekodet slik at ARAnchor-objektene kan vises for medspillere.

*Multipeer Connectivity*, som er koblingsteknologien for flerspillermodus, sender kontinuerlig spørringer til enheter innenfor rekkevidde til den finner en annen aktiv bruker. Disse enhetene synkroniseres og kan deretter se hverandres aktivitet i sanntid.

```
44 func send(anchor: AirAnchor) {  
45     print("sending anchor to \(session.connectedPeers.count) peers...")  
46     if session.connectedPeers.count > 0 {  
47         do {  
48             guard let data = try? NSKeyedArchiver.archivedData(withRootObject: anchor,  
49                 requiringSecureCoding: false) else { fatalError("can't encode anchor") }  
50             try self.session.send(data, toPeers: session.connectedPeers, with: .reliable)  
51         }  
52         catch let error {  
53             print("Error for sending: \(error)")  
54         }  
55     }  
56 }
```

**FIGURE 14, FUNKSJONEN SOM KOMPRIMERER OG SENDER AIRANCHOR-OBJEKTER**

## 3. Produktdokumentasjon

---

Sketchworld: AR

*Bachelorprosjekt 2019*

### 3.1 Introduksjon

Dette kapitlet dokumenterer applikasjonen sine funksjoner og brukergrensesnitt. Hensikten er å illustrere hvordan applikasjonen fungerer, og gi en teknisk gjennomgang av funksjonalitetene.

### 3.2 Brukergrensesnitt

For brukergrensesnittet til applikasjonen benyttet vi en moderne tilnærming for designet, og har latt oss inspirere av andre applikasjoner slik som Snapchat og Reddit. Begge appene har responsivt design der navigasjon mellom delsider føles intuitivt. For å oppnå en lignende brukeropplevelse, har vi benyttet oss av en *MaterialDesign* navigasjonsmeny sammen med et *UICollectionView*, som arver alle metodene til *UIScrollView*, og gjør at man kan sveipe gjennom de forskjellige delsidene. Sideoppsettet vår er et tredelt og brukeren ankommer hovedmenyen først, for så å kunne navigere seg til høyre eller venstre ved å dra fingeren på kanten av skjermen eller trykke på navigasjonsknappene.

#### 3.2.1 Navigasjon med VoiceOver

Med universell utforming i tankene har vi implementert hjelpetekst på alle menyer og interaksjonselementer gjennom at de har fått en *AccessibilityLabel*, og et *AccessibilityHint* der det passet seg. For eksempel så sier VoiceOver-programmet navnet på de forskjellige navigasjonsknappene slik som “map button” eller “menu button”, og det er også slik at når man endrer på verdien i fargevelgeren vil VoiceOver si ifra hvilken farge det byttes til gjennom en notifikasjon fra *UIAccessibility* klassen.

#### 3.2.2 Brukerinteraksjoner

En prioritet under utviklingen av brukergrensesnittet var å åpne muligheten for å gjennomføre alle funksjoner med færrest mulig tastetrykk. For eksempel funksjoner som endring av farge, endre størrelse på pensel osv, er alle tilgjengelig rett på skjermen, uten at man trenger å gå innom menyer. I tillegg ønsket vi å åpne for effektiv enhandsbruk, gjennom å plassere alle interaksjonselementer innenfor rekkevidden til tommelen.

#### 3.2.3 Dynamisk plasserte UI elementer

Vi hadde et mål om at mobilapplikasjonen skulle fungere over flere skjermstørrelser, og vi har derfor utviklet det slik at alle elementer i appen skal være responsive. Vi bruker *AutoLayout* og *constraints* systemet til Xcode for å sette opp dynamiske forhold mellom kanter og flater, slik at brukeren skal kunne få relativt lik brukeropplevelse på ulike enheter.

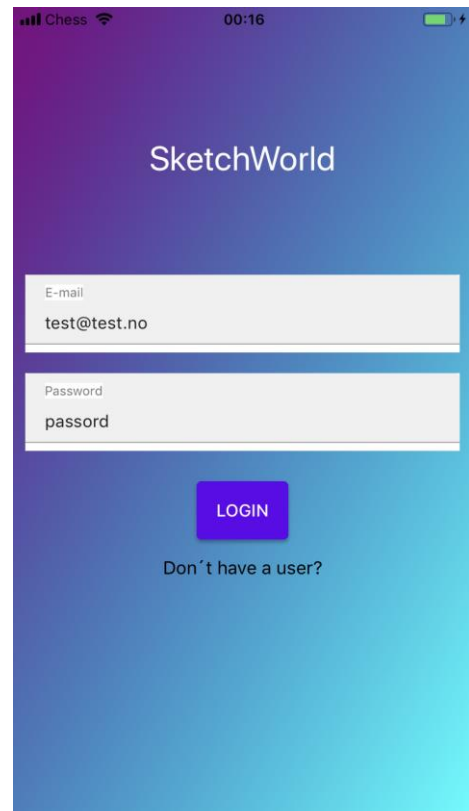
### 3.2.4 Innlogging og registrering

Når brukeren først åpner applikasjonen blir de først møtt av innloggings- og registrerings-sider. Her kan man benytte seg av en eksisterende konto eller velge å gå til registreringssiden for å registrere en ny brukerkonto.

På begge sidene befinner det seg to tekstfelt, et for e-post og et for passord. På tekstfeltene er det validering av input slik at man må skrive inn en ekte e-postadresse for å logge inn. I tillegg til tekstfeltene er det også en handlingsknapp som håndterer innlogging og opprettelse av ny bruker. Enter-knappen kan her brukes som navigasjon mellom elementene og det er også implementert at tastaturet lukker seg hvis man trykker på andre plasser enn tekstfeltene.

Autentisering foregår gjennom Firebase, det er en innebygd funksjon som kan avslå brukeropprettelse dersom e-posten ikke er unik. Om autentiseringen er godkjent vil brukeren bli navigert til applikasjonens hovedside. Se kapittel 3.5 for dokumentasjon av autentisering.

For å forenkle innloggingsprosessen ved flergangsbruk av applikasjonen, har vi implementert lokal lagring på den individuelle mobile enheten via *UserDefaults*, som er en lokal database på telefonen som opprettholdes selv ved omstart av enheten. Vi har benyttet denne datalagringen til å automatisk fylle inn både e-post og passord i innloggingsfeltene neste gang applikasjonen starter.



**FIGURE 15, SKJERMBILDE AV  
INNLOGGINGSSKJERMEN**

### 3.2.5 Hovedside

Hovedfunksjonaliteten i applikasjonen vår ligger i AR-siden, så vi har designet brukerflyten slik at brukeren kommer rett inn i AR-opplevelsen. Her kan brukeren umiddelbart utforske ulike funksjonaliteter slik som tegning og interaksjoner med tredimensjonale objekter. Interaktive elementer slik som angreknapp og fargevelgeren er plassert direkte på vinduet øver på høyre side, mens det befinner seg et tekstfelt øverst på venstre side, som vi har benyttet for tilbakemeldinger.

#### Tegning

I UIKit rammeverket finnes det innebygde funksjoner for håndtering av interaksjoner med skjermen, disse kalles for *GestureRecognizers*, og vi brukte disse for implementeringen av de forskjellige tegnemethodene. Hver av disse interaksjonfunksjonene sender tredimensjonale som er ekstrapolert fra lokasjonen man trykker på som koordinater til individuelle ARAnchor objekter. Disse objektene er ARKit objekter som har ansvar for å lagre posisjoner til virtuelle objekter relativt til et nullpunkt. Nullpunktet blir satt på den fysiske lokasjonen man befinner seg på når scenen blir instansiert. ARAnchor objekter lagrer en relativ posisjon, og dynamisk plasserer noder på posisjonene lagret i på dem gjennom en *ARSCNViewDelegate* rendering funksjon, som lytter etter nyplasserte ARAnchor objekter. Vi valgte å utvide ARAnchor klassen, slik at de ikke bare inneholder en posisjon, men også farge og størrelse, slik at grafikken som tegnes på lokasjonen til ARAnchor objektet er dynamisk.



FIGURE 16, SKJERMBILDE AV  
TEGNESIDEN

Tegne funksjonen kan aktiveres på tre ulike måter;

- Ved å trykke på skjermen.
- Ved å dra fingeren over skjermen.
- Ved å holde inne et punkt på skjermen og bevege smarttelefonen.

Det er også mulig å kombinere sveiping over skjermen mens man samtidig beveger smarttelefonen rundt i rommet. Grunnen til at vi implementerte flere måter å tegne på var at vi ønsket ikke at brukere skulle føle at vår "spesifikke" tegnefunksjonalitet er et hinder for deres kreativitet.

```

118     @IBAction func handleTap(sender : UITapGestureRecognizer) {
119         let location = sender.location(in: sceneView)
120         guard let cameraPosition = sceneView.session.currentFrame?.camera.transform
            else {
121             return
122         }
123         let worldPosition = cameraPosition + getDirection(for: location, in:
            sceneView)
124         if sender.state == .ended {
125             placeAnchors(onPoint: worldPosition, breakpoint: true)
126         }
127     }

```

FIGURE 17, KODESNUTT MED TOUCHRECOGNIZER

### Display av status

For å optimalisere plasseringen av objekter i tredimensjonale verden, bør ARKit sin romkalibreringsprosess få tildelt tid til å gjenkjenne nok objekter og overflater i omgivelsene rundt brukeren. Derfor valgte vi å implementere et statusfelt hvor brukeren får informasjon når ARKit trenger å kalibrere, eller rekalkulere rommet for å kunne plassere objekter korrekt i forhold til hverandre. Om ARKit merker at kalibrasjonen er feil, vil statusbaren gi brukeren varsel om at kalibrering er nødvendig for videre interaksjon med miljøet skal være optimalt.

Statusbaren kan sees i vedlegg 3, der det er et skjermbilde av hovedsiden mens den er i en relokasjonsprosess

.

### Angreknapp

Som nevnt tidligere i kapittel 2.X er hver node lagret med en breakpoint attributt som forteller hvor mange elementer som hører til hver grafiske linje. For å gi funksjonalitet til angreknappen har vi satt opp en kodeløkke eller *do-while loop* som sletter siste node og fortsetter å slette noder helt til den finner en node med et breakpoint attributt.

```

215     /// Recurcivly removes nodes until it hits a breakpoint or there are no nodes left.
216     func removeTailNode() {
217         if (self.tailNode != nil){
218             //do-while loop which runs atleast once if there is a tailNode
219             repeat {
220                 sceneView.session.remove(anchor: tailNode!.anchor)
221                 self.tailNode?.next = nil
222                 self.tailNode = self.tailNode?.prev ?? nil
223             } while (self.tailNode != nil && self.tailNode?.anchor.breakpoint == false)
224         }
225     }

```

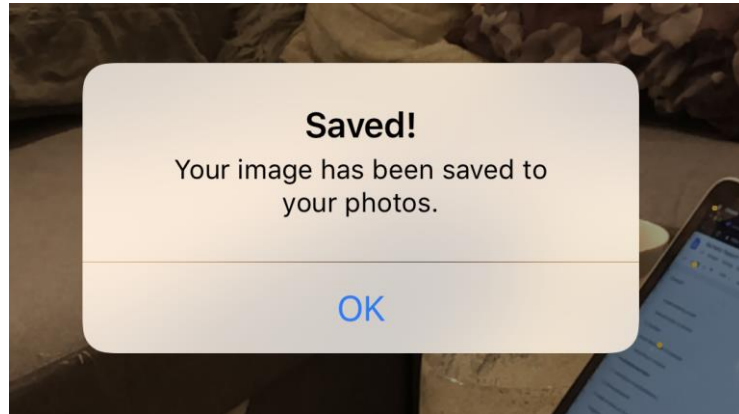
FIGURE 18, KODESNUTT MED WHILE LOOP

### ColorSlider

For å gi brukern muligheten til å lage mer dybde i tegningene, har vi implementert en komponent for valg av farge. Denne er plassert øverst i høyre hjørne slik at det er enkelt å ta i bruk knappen med tommelen ved enhåndsbruk. Denne komponenten har en dynamisk mot den globale fargeattributten i prosjektet, og den verdien blir benyttet når det tegnes nye elementer.

### Innebygd skjermbilde

For at opplevelsen skal kunne deles med andre ville vi inkludere en mulighet for bildelagring, og har derfor valgt å implementere en skjermbildeknapp. Knappen er kun aktiv når brukeren er inne på hovedscenen. Dette er grunnet at knappen ligger i en annen klasse enn der skjermbildefunksjonen befinner seg. Vi har løst kommunikasjon gjennom klassene med implementasjon av et notifikasjon system gjennom Swift sitt NSNotification bibliotek, der vi har en sendefunksjon i klassen hvor knappen befinner seg, og en mottakerfunksjon i klassen der skjermbildefunksjonen befinner seg.



**FIGURE 19, SKJERMBILDE AV TILBAKEMELDING AV BILDELAGRING**



### 3.2.6 Kartside

Kart- og navigasjonssiden følger samme designprinsipp som hovedsiden, det alle brukerinteraksjoner skjer på høyresiden. Hensikten med denne siden er å gi en oversikt over hvor det er mye tegneaktivitet. Dette gir muligheten til å finne andre brukere som er aktive, og potensielt i framtiden se hva de har tegnet.

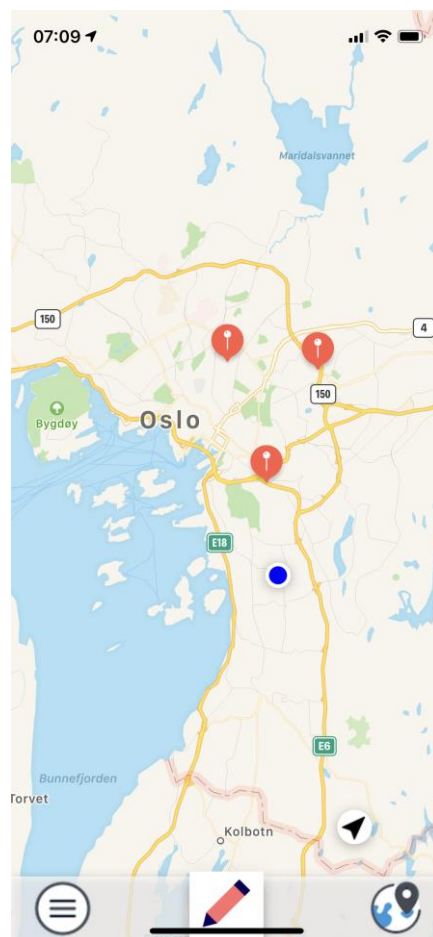
#### Annotasjoner

Når brukeren navigerer seg over til kartet gjøres en spørring til databasen som er lokalisert i skyen til Firebase. Først oppretter vi en kobling til databasen, for så å iterere over alle objektene i tabellen. For hvert objekt griper vi tak i koordinatverdier som vi bruker til å plassere annotasjoner i kart scenen.

Koordinatene til brukeren sine tegninger blir benyttet til opprettelse av markører som deretter blir plassert direkte på kartet. Konstruksjonen av markører gjennomfører vi ved å bruke MKPointAnnotation som er en modell for opprettelse av markører gjennom biblioteket MapKit.

#### Posisjonshåndtering

For å gi brukeren oversikt over egen lokasjon, implementerte vi en knapp som beveger kartet bort til brukeren sine koordinater, og deretter følger brukeren sin plassering. Funksjonen er tilgjengelig via MapKit. Denne funksjonaliteten gjør det enklere for brukere å finne tilbake til tidligere tegninger, og navigasjon under bruk av applikasjonen.



**FIGURE 20, SKJERMBILDE AV KARTSIDEN**

### 3.2.7 Profilside

Vi diskuterte mellom oss selv om vi skulle lage en enkel meny løsning kontrollert gjennom hovedsiden der generelle funksjonaliteter relatert til andre ting utenom AR skulle håndteres. For å unngå rot og komplisert navigasjon valgte vi å trekke innholdet av en slik meny ut og inn i en separat side.

Her vil brukeren få oversikt over hvilken konto han/ hun er logget inn på, og kan logge seg ut av applikasjonen ved behov. Denne siden inneholder per dags dato bare basisfunksjoner slik som å se brukernavnet sitt og å logge ut-

#### Utlogging

Muligheten til å logge ut/ bytte konto er en viktig funksjon for å kunne tillate deling av samme mobil enhet uten å overskride hverandres progresjon. Gjennom *Firebase* implementere vi håndtering for utlogging og har plassert denne funksjonaliteten gjennom en knapp. Ved godkjent utlogging kalles det på en overgang tilbake til innloggingsiden.

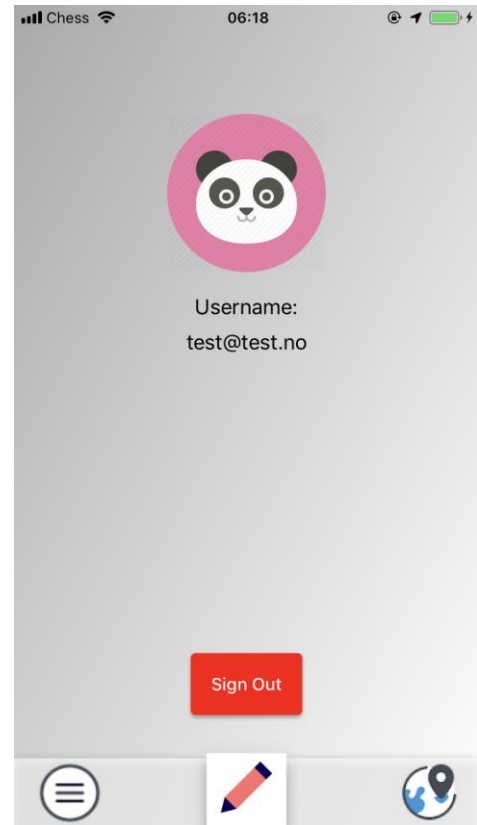


FIGURE 21, SKJERMBILDE AV PROFIL / MENYSIDEN

### 3.3 Tilgangstillatelse

For å få tilgang til nødvendig instrumenter på smarttelefonen, sender vi en notifikasjon til brukeren med forespørsel om tilgang til disse instrumentene. Dette må vi gjøre for å gi brukeren muligheten til å godta eller avslå slike forespørsler.

Vi har av denne grunn implementert varsling til brukeren med alternativ om å godkjenne at vår applikasjonen får tilgang til kamera for AR-funksjonaliteten, og tilgang til stedstjeneste for kart vinduet.

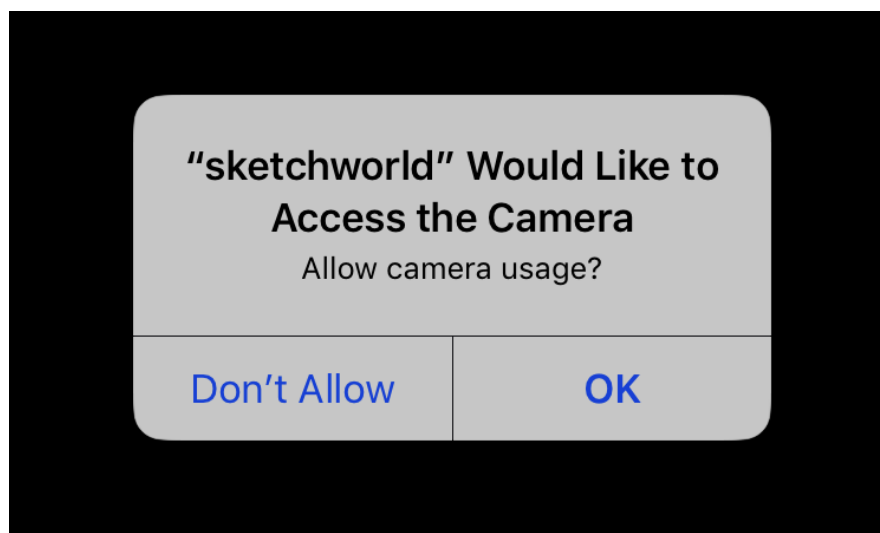


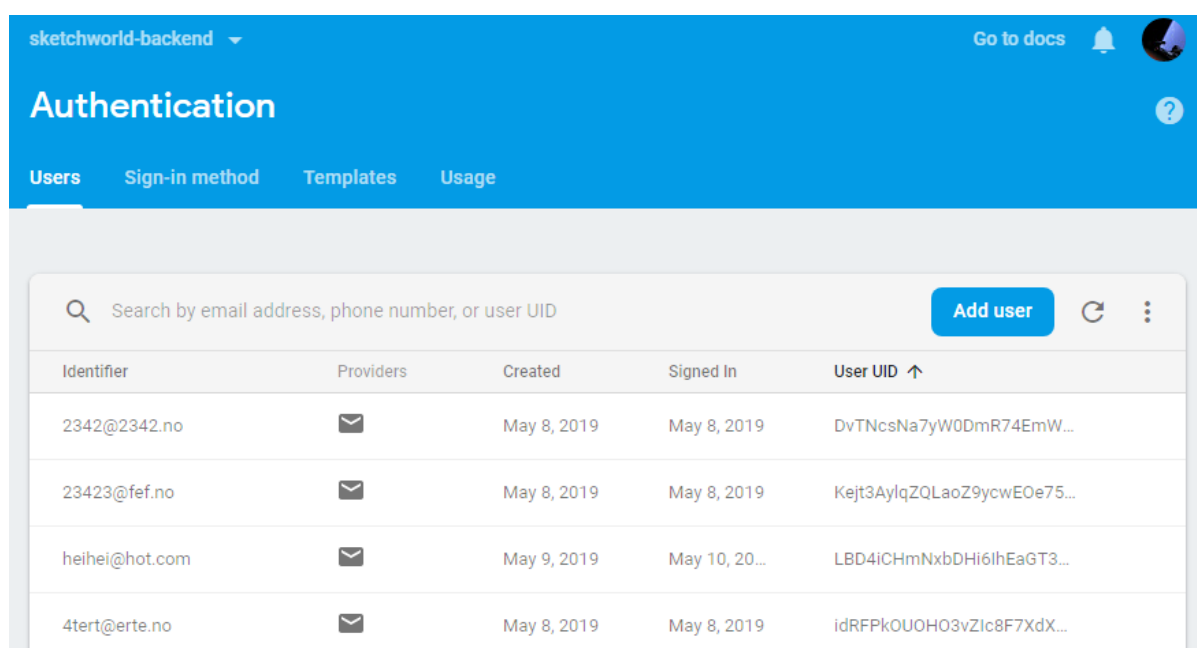
FIGURE 22, EKSEMPEL PÅ SPØRSMÅL OM TILANG

## 3.4 Systemarkitektur

For datalagring i applikasjonen vår vurderte vi et par alternativer. I tidligere prosjekter har vi benyttet oss av både Azure og AWS sine skytjenester, men valgte her å gå for Firebase. Dette er grunnet de har god integrasjon med apputvikling, i tillegg til at tjenesten er gratis for bruk som ikke krever mye nettverkstrafikk. Firebase er en skytjenesteplattform som eies av Google, og passer bra til applikasjoner som er i en tidlig utviklingsfase og som trenger å komme fort på bena. De tilbyr gode løsninger for oppsett av databaser og automatisk oppsett av REST API tjenester for databasespørringer gjennom *Post* og *Get* metoder, og de tilbyr en brukervennlig portal.

### 3.4.1 Autentisering med Firebase

Firebase tilbyr autentiseringsmetoder for å enkel implementasjon av innlogging og utlogging av brukere.



The screenshot shows the 'Authentication' page in the Firebase console. At the top, there's a blue header with 'sketchworld-backend' and 'Go to docs'. Below the header, there's a navigation bar with 'Users', 'Sign-in method', 'Templates', and 'Usage'. The 'Users' tab is selected. Below the navigation bar, there's a search bar with the text 'Search by email address, phone number, or user UID'. To the right of the search bar is an 'Add user' button. Below the search bar is a table with the following columns: 'Identifier', 'Providers', 'Created', 'Signed In', and 'User UID'. The table contains four rows of user data.

Identifier	Providers	Created	Signed In	User UID ↑
2342@2342.no	✉	May 8, 2019	May 8, 2019	DvTNcsNa7yW0DmR74EmW...
23423@fef.no	✉	May 8, 2019	May 8, 2019	Kejt3AylqZQLaoZ9ycwEOe75...
heihei@hotmail.com	✉	May 9, 2019	May 10, 20...	LBD4iCHmNxbDHI6lhEaGT3...
4tert@erte.no	✉	May 8, 2019	May 8, 2019	idRFPkOUOH03vZlc8F7XdX...

FIGURE 23, SKJERMBILDE AV BRUKERE LAGRET I EN AUTENTISERINGSDATABASE FRA FIREBASE

### 3.4.2 Sikkerhet

Firebase er en skylagring plattform som kan aksesseres direkte av hvilken som helst bruker. Derfor har de et regelsystem som gjør at alle brukere som ønsker å sette opp en database er nødt til å skrive regler for tilkoblinger. Om en bruker ikke skriver ordentlige regler er dataen åpen for trusler og angrep. Regelsystemet styres gjennom et JSON-objekt som eier av database kan redigere, og personalisere til spesifikk bruk. Vi utvikler en enkel applikasjon som kun kan lastes ned ved tilgang til utviklerkontoen vi bruker i prosjektet, og har av den grunn satt opp relativt sikre men enkle regler.

### 3.4.3 Datalagring i Firebase

I likhet med autentisering i Firebase så er opprettelse og vedlikehold av database tabeller en enkel og rask prosess. I det administrative vinduet kan eier av databasen legge til, endre eller slette elementer når som helst.

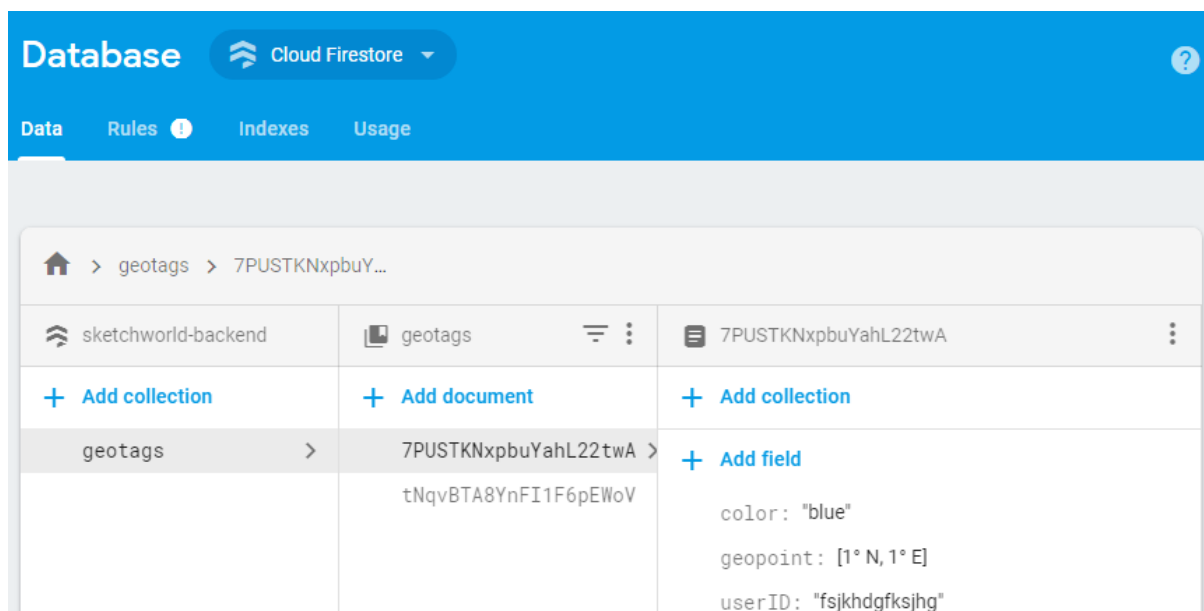


FIGURE 24, SKJERMBILDE AV VÅR DATABASE FRA FIREBASE

## 4. Refleksjoner og læringsutbytte

---

Sketchworld: AR

*Bachelorprosjekt 2019*

## 4.1 Introduksjon

I dette kapittelet vil resultatene bli drøftet. Dette inkluderer arbeidsmetodene våre, valg av teknologi, tanker om det endelige produktet, og hvordan applikasjonen kan videreutvikles.

## 4.2 Refleksjon rundt arbeidsmetoder

Vi mener gjennomgående arbeidsflyt for prosjektet har vært effektiv og godt organisert. Kanban board kombinert med Discord har bevist seg å være svært god arbeidsmetode for en arbeidsprosess der hvor partene arbeider store deler av prosjektet over nettet.

Kanban tillot gruppe medlemmene å starte prosjektarbeidet uten en konkret, og ferdigstilt arbeidsplan. Det å ha et interaktivt aktivitetsbrett der oppgaver kan tildeles underveis og endres med stor frihet, passet vårt prosjekt perfekt grunnet vi var nødt til å lære mye underveis i utviklingen og tilpasse målene våre kontinuerlig.

## 4.3 Produktiterasjoner

Produktet var gjennom forskjellige iterasjon av design, hvor første prototype konstruert med Adobe XD var et godt utgangspunkt, selv om det ikke endte opp med å representere verken design eller hovedfunksjoner på en korrekt måte. Vi kan konkludere med at vi er fornøyd med Adobe XD som verktøy for prototyping, da det fungerte godt for å sette opp struktur, og få et overblikk over applikasjonens vinkling.

Neste iterasjon av produktet ble produsert gjennom Xcode under utvikling, hvor det var enklere for oss å diskutere valg av komponenter og utseende underveis mens vi utviklet produktet, grunnet vi hadde en bratt læringskurve samtidig som vi planla designet. Disse versjonene var bedre funksjonell representasjon av produktet, men manglet et presentabelt design.

Vi er tilfreds med applikasjonens totale livssyklus, fra ide til produkt, og ville gjort prototype prosessen på en lignende måte ved et lignende prosjekt. Eneste hovedforskjellen hadde vært kompetansen vi har nå opparbeidet oss ville vært gunstig til design av en applikasjon med samme teknologiske grunnlag.

## 4.4 Resultatet

Tilbakeblikk til prototypen vi først konstruerte, viser en stor endring i kompetanse innenfor AR. Vi hadde lite kunnskap og ingen erfaring med ARKit og var derfor usikre på hva som var realistiske å forvente av sluttproduktet.

Produktet endte opp med å ha en godt integrert AR funksjon, som overgikk hva vi forventet å kunne gjennomføre i løpet av prosjekttiden.

### 4.4.1 Måloppnåelse

Vi gjennomførte de primære målene våre som besto av å utvikle en AR applikasjon for iOS enheter, med integrert tegnefunksjonalitet av tredimensjonale objekter, og en fungerende flerspillermodus.

#### 4.4.2 Performance

AR funksjonalitet er i utgangspunktet en relativt energikrevende teknologi, hvor det er flere instrumenter i konstant aktivitet under kjøring. Negative effekter som kan følge dette er høy ressursbruk og overoppheting. Derfor er det viktig å effektivisere funksjoner og unngå kode som fører til unødvendig prosessor- og minnebruk. Se figur x. for oversikt over prosessorbruk og minnebruk for vår applikasjon.

Et aspekt som vi la merke til i plassering av sfærer var at dette kunne føre til hyppig overlapping av objekter når en bruker tegnet trege streker. Konsekvensen av dette resulterer i at applikasjon blir mer ressurskrevende, og fører til økt CPU- og minnebruk. Et ARWorldMap som inneholder masse AR-objekter vil muligens kunne gjøre brukeropplevelsen mindre konsistent grunnet redusert oppdateringshastighet av skjermbilde.

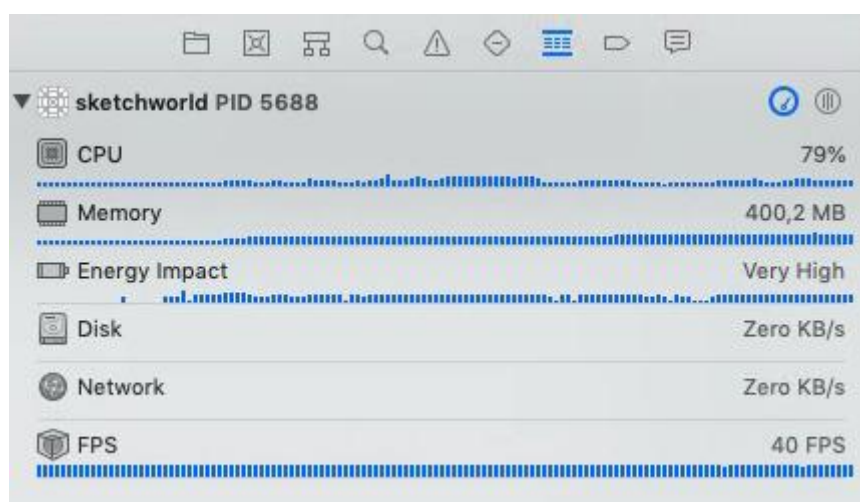


FIGURE 25, UTKLIPP FRA XCODE SOM VISER

#### 4.4.3 Deployment og hosting

Basert på resultatene, hadde vi et ønske om å publisere applikasjonen på Apple sin nettbutikk, App Store. Dette kunne vi dessverre ikke gjennomføre grunnet Apple kun tillater utviklere som er deltagere av *Apple Developer Program*, rettigheten til å publisere private applikasjoner i deres nettbutikk. For å bli deltaker av dette programmet er en nødt til å betale en årsavgift.

### 4.5 Potensiell videreutvikling

Under utviklingen av applikasjonen og læringsprosessen rundt AR, fikk vi flere potensielle ideer for videreutvikling av applikasjonen. Noen av disse ideene er allerede realisert og befinner seg i produktet, mens andre fikk vi ikke tid til å gjennomføre grunnet tidsrammen.

#### 4.5.1 Serverbaserte ARWorldMap-objekter

En funksjon som var plassert høyt på oppgavelisten var å utvikle en inndeling av geografiske områder der grafikk automatisk lastes inn, og en kan se hva folk tidligere har tegnet innenfor avgrensningene. Dette er per dags dato ikke det som skjer da ARWorldMap-objektene våre kun deles med personer innenfor bluetooth rekkevidde.

En potensiell løsning på denne problemstillingen hadde vært å lagre ARWorldMap-objekter på en server, og et objekt blir sendt til en bruker hvis de befinner seg innenfor et geografisk område. Geografisk avgrensning eller *Geofencing* handler i korte trekk om å kunne avgrense/ gruppere geografiske lokasjoner, og bestemme om en bruker befinner seg innenfor det området. I en presentasjon av AR-applikasjonen SwiftShot utviklet av Apple, fortalte Rosenberg et al. (2018, 10:12 - 14:10) at de hadde brukt iBeacons som en server eller vert på lokasjoner der applikasjonen deres skulle bli brukt, og at de manuelt hadde kartlagt områdene. De nevner også at det er mulig å lagre ARWorldMap-objekter på en server, så det å få et nytt ARWorldMap når man kommer innenfor et område er mest sannsynlig mulig å implementere.

#### 4.5.2 Gamification elementer

Å øke underholdningsverdien i produktet er en viktig investering om applikasjonen skulle vært publisert til offentligheten. Her er det da spesielt viktig å fylle produktet med innholdsrike funksjoner som kan gi applikasjonen en sosial nytteverdi.

##### Avstemningssystem på tegninger

Tegninger får en inkremental verdi knyttet til seg som brukere kan bidra til å stemme frem kreasjoner. Dette vil raskt øke sosiale verdien til applikasjonen og kan bidra positivt til kreativ utfolding samt vennlig konkurranse.

##### Abonnering

Om brukere får muligheten til å abonnere på andre brukere sin aktivitetslogg, hvor da ved en slik konseptuell situasjon en bruker kunne fulgt sine venner og bli oppdaterte på deres siste tegninger. Eventuelt kunne reise ved å ta i bruk kartsiden hvor man kan filtrer frem en utvalgt abonnents sine geografiske annotasjoner for hvor de har tegnet.

##### Utmerkelser

Utmerkelser er en veldig vanlig implementasjon i spill/ applikasjoner som ønsker å inspirere sine brukere til å oppnå personlige mål. Her vil en potensiell ide være å konstruere et utmerkelses system for alle brukere med oppgaver som bidrar til aktivitet og underholdningsverdi slik som f.eks; "Tegne totalt 100 tegninger", "Aktiv bruker i 20 dager", "Samlet 1000 stemmer på en tegning". Hovedpoenget med en slik utvidelse vil være å invitere til progressiv utvikling av egen profil.

#### 4.5.3 OAuth for Facebook innlogging

Innlogging med Facebook er veldig vanlig måte å håndtere innlogging i moderne apputvikling. Å logge inn med Facebook er en enklere prosess for en bruker enn å måtte sette opp en ny bruker, da man slipper å godkjenne en e-postadresse pluss å finne på enda et nytt passord. Innloggingsflyten blir enklere gjør at man får høyere kundebevaring, og det at Facebook logoen ligger på innloggingsskjermen gjør at en bruker vil føle at appen er mer pålitelig. Vi valgte å nedprioritere denne funksjonen da vi allerede har en autentiseringsfunksjon gjennom Firebase.



## 5 Litteratur

Apple Inc. (2018a). ARKit documentation. Hentet fra <https://developer.apple.com/documentation/arkit>

Apple Inc. (2018b). Archiving World Map Data for Persistence or Sharing. Hentet fra [https://developer.apple.com/documentation/arkit/arworldmap/archiving\\_world\\_map\\_data\\_for\\_persistence\\_or\\_sharing](https://developer.apple.com/documentation/arkit/arworldmap/archiving_world_map_data_for_persistence_or_sharing)

Apple Inc. (2018c). Creating a Multiuser AR Experience. Hentet fra [https://developer.apple.com/documentation/arkit/creating\\_a\\_multiuser\\_ar\\_experience](https://developer.apple.com/documentation/arkit/creating_a_multiuser_ar_experience)

Apple Inc. (n. d. ). Class: MCSession. Hentet fra <https://developer.apple.com/documentation/multipeerconnectivity/mcsession>

Begbie, C. (2017). Augmented Reality and ARKit Tutorial. Hentet fra <https://www.raywenderlich.com/378-augmented-reality-and-arkit-tutorial>

Esplin, C. (2017). Firebase Security & Rules. Hentet fra <https://howtofirebase.com/firebase-security-rules-88d94606ce4a>

Google LLC. (2018). Get data with Cloud Firestore. Hentet fra <https://firebase.google.com/docs/firestore/query-data/get-data>

Jayven, N. (2018). Saving and Restoring World-mapping Data to Create a Persistence AR Experience. . Hentet fra <https://www.appcoda.com/arkit-persistence/>

Raval, S. (2019). Grouping elements for better accessibility on iOS. Hentet fra <https://thoughtbot.com/blog/grouping-elements-for-better-accessibility-on-ios>

Rosenberg, A. (2018). Inside SwiftShot: Creating an AR Game. Video. Hentet fra <https://developer.apple.com/videos/play/wwdc2018/605/>

Samuel, R. (2019). Making better app login screens—a UX case study. . Hentet fra <https://uxdesign.cc/making-better-app-login-screens-6d8b9c4f80c5>

Sutherland, I. E. (1968). A head-mounted three dimensional display\*. The University of Utah. Salt Lake City, Utah

Thrive Analytics, L. (2018). AUGMENTED REALITY - A CLOSER LOOK AT APP USAGE. In. <http://www.thriveanalytics.com/images/Thrive%20Analytics-%20AR%20Infographic.pdf>.

Voong, B. (2016). Swift: YouTube - How to Swipe between sections horizontally. Hentet fra [https://www.letsbuildthatapp.com/course\\_video?id=75](https://www.letsbuildthatapp.com/course_video?id=75)

## 6. Vedlegg

### Vedlegg 1: Oppgaveteksten

#### AR sketchPAD

Dette prosjektet går ut på å utvikle en mobilapplikasjon som lar flere brukere lage skisser ved hjelp av augmented reality på smarttelefoner i felleskap. Brukerne skisser på toppen av sine omgivelser med smarttelefonen og kan innsisere dette og andres skisser fra samme sted. Resultatet kan eksporteres i forskjellige formater, blant annet som panoramaskisser.

#### Teknologi

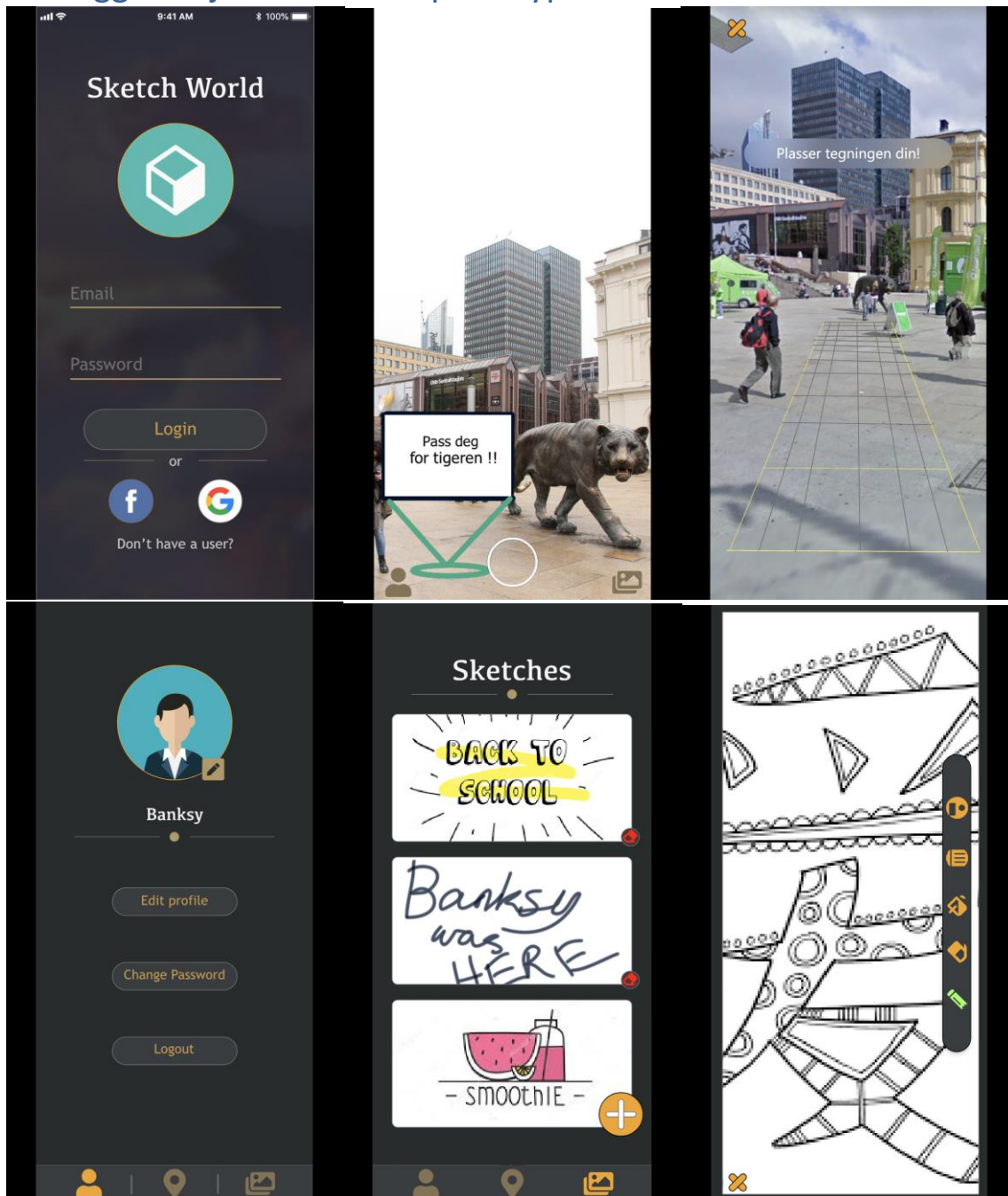
Det er tenkt at Googles ARcore API blir grunnsteinen for prosjektet og at Android blir hovedplattform. En del av prosjektet går ut på å utvikle diverse utvidelser utover enkle skisser.

#### Interessert

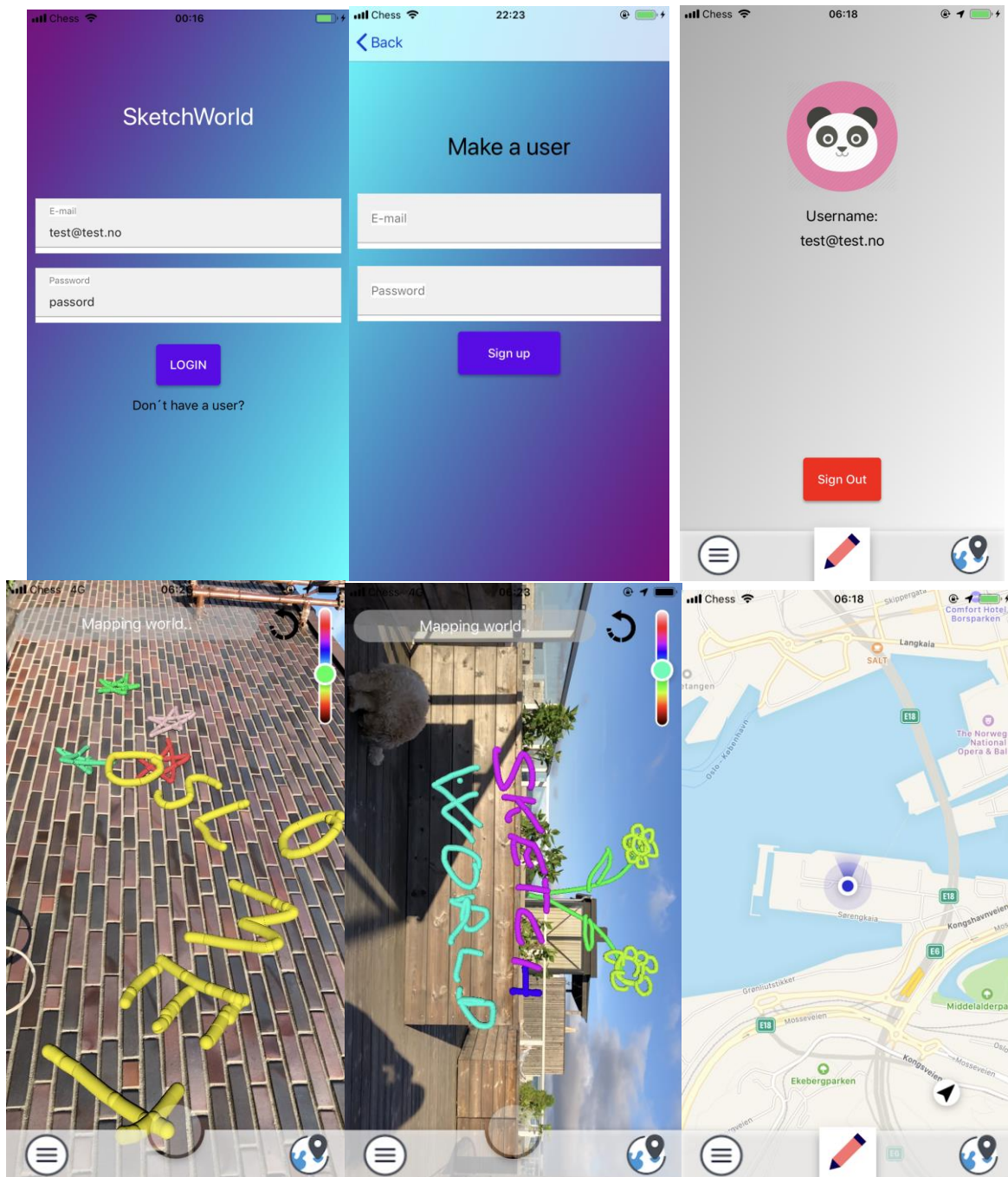
Dersom du er interessert, ta kontakt med Frode Eika Sandnes

[frodes@oslomet.no](mailto:frodes@oslomet.no)





## Vedlegg 2: Skjermbilder fra prototype



### Vedlegg 3: Skjermbilder av produktet tatt med iPhone 8



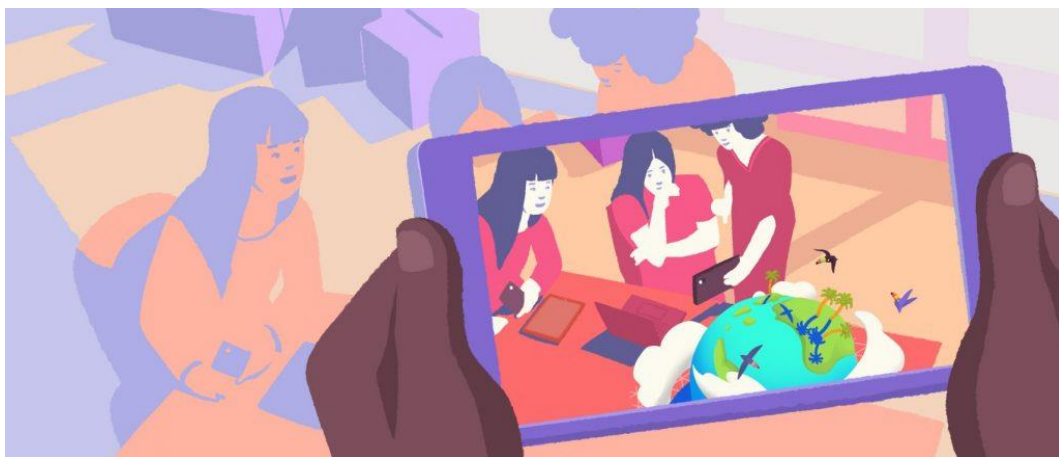
## Vedlegg 4: Kopirettigheter

			
Fra NounProject, Creative Commons	Fra iconfinder, Free for commercial use	Fra Icon8, Creative Commons Attribution- NoDerivs 3.0	



Fra iconfinder

Free for commercial use



Fra [developers.google.com](https://developers.google.com)

Creative Commons Attribution 4.0 License