



## **MATBIT PROSJEKTRAPPORT**

Thomas Angeland

Høgskolen i Østfold, ITF21013 Android-programmering

01.12.2017

## **Prosjektinfo**

### **Sammendrag**

Rapporten tar for seg prosjektet Matbit, utført av dataingeniørstudent Thomas Angeland i Android-Programmering-faget ved Høgskolen i Østfold 2017. Her beskrives prosjektoppgaven og hvordan prosjektdeltaker tenker å utføre prosjektet. Prosjektdeltaker vil dokumentere planleggingsfasen og utførelsen av prosjektet, og forklare hvilke valg som ble tatt i prosjektperioden. Etter det vil det være en refleksjon som lister utfordringer, forslag til forandringer og prosjektdeltakerens vurdering av prosjektet. Tilslutt legges ved en skryteliste og en brukerveiledning.

### **Takk til**

Varm takk til faglærer Lars Emil Knudsen og veileder Adrian Jensby Sandaker for hjelp og gode råd. Mange takk til samboer, familie og venner for støtte under prosjektperioden.

## Innholdsfortegnelse

<b>Kapittel 1: Introduksjon</b>	<b>4</b>
1.1 Om prosjektdeltaker	4
1.2 Oppgaven	4
1.3 Hvorfor, hva og hvordan: Formål, leveranser og metode	5
1.4.1 Formål	5
1.4.2 Leveranse	5
1.4.3 Metode	6
<b>Kapittel 2: Analyse og planlegging</b>	<b>7</b>
2.1 Hva og hvorfor	7
2.2 Kravspesifikasjon	7
2.2.1 Matbit skal inneholde følgende:	7
2.2.2 Matbit bør inneholde følgende:	8
2.2.3 Matbit kan inneholde følgende:	8
2.2.4 Matbit skal ikke inneholde følgende:	9
2.2 Oppbygging og navigasjon	9
2.3 Database	12
2.4 Design	16
2.5 Ressurser	17
2.6 Eksterne biblioteker	17
2.7 Klasser	18
2.8 Usikkerhet	19
<b>Kapittel 3: Gjennomføring</b>	<b>20</b>
3.1 Oppsett	20
3.2 Database	20
3.2.1 Skrive og lese data	21
3.2.2 MatbitDatabase-klassen	24
3.3 Aktiviteter	25
3.3.1 SignInActivity	26
3.3.2 MainActivity	27
3.3.3 SearchActivity	29
3.3.4 RecipeActivity	32
3.3.4.1 RecipeFragmentInfo.java	32
3.3.4.2 RecipeFragmentIngredients.java	35

3.3.4.3 RecipeFragmentSteps.java	35
3.3.4.4 RecipeFragmentComments.java	36
3.3.5 AddRecipeActivity	37
3.3.6 Brukerprofil	39
3.3.6.1 Rediger brukerprofil	40
3.3.6.2 Mine Oppskrifter	41
3.4 Firebase Notifications	42
3.5 Firebase Persistence Mode	43
3.6 Testing	43
3.6.1 Anonyme brukere	43
3.6.2 Internettavbrudd	44
<b>Kapittel 4: Refleksjon</b>	<b>45</b>
<b>Kapittel 5: Konklusjon</b>	<b>47</b>
<b>Skryteliste</b>	<b>48</b>
<b>Installasjonsveiledning</b>	<b>49</b>
<b>Referanser</b>	<b>50</b>
<b>Figurer</b>	<b>54</b>
<b>Vedlegg</b>	<b>57</b>

## Kapittel 1: Introduksjon

### 1.1 Om prosjektdeltaker

Prosjektdeltaker Thomas Angeland, født 1994, studerte som dataingeniørstudent ved Høgskolen i Østfold 2017. Thomas har arbeidserfaring i nettside design, backend og frontend, datagrafikk, algoritmer og datastrukturer, sikkerhet og hacking, databaser, og programmerer i PHP, SQL, C#, C++ og Java.

### 1.2 Oppgaven

Prosjektdeltaker skal gjennomføre et utviklingsprosjekt for en selvvalgt mobilapplikasjon.

Dette prosjektet var grunnlaget for evaluering i faget ITF21013 Android-Programmering ved Høgskolen i Østfold høst 2017. Prosjektdeltaker skal vise beherskelse av å planlegge, designe, dokumentere, implementere og teste en egenprodusert Android applikasjon.

Prosjektdeltaker skal også vise til beherskelse av

- Java
- Oppbygging av brukergrensesnitt
- Landscape/Portrait
- Datalagring
- Støtte for forskjellige Android versjoner
- Feilhåndtering
- Kreativ appdesign
- Henviser til bruk av design guidelines
- Inneholder servicer
- Innhenting av innhold fra eksterne kilder
- Bruk av eksterne biblioteker

### 1.3 Hvorfor, hva og hvordan: Formål, leveranser og metode

#### 1.4.1 Formål

Hovedmål: Produsere en komplett app som kjører uten feil, og som støtter Android SDK 19 til 26.

Delmål 1: Innhente kunnskaper om hvordan å lage en komplett Android app.

Delmål 2: Innhente kunnskaper om hvordan å benytte Google Firebase biblioteker.

Delmål 3: Innhente kunnskaper om hvordan å designe en app ved å følge Google Material Design.

Delmål 4: Innhente kunnskaper om hvordan å planlegge, gjennomføre og reflektere et utviklingsprosjekt.

#### 1.4.2 Leveranse

Prosjektet leveres i deltakerens navn. Det medfølge en liten beskrivelse av prosjektet, en installasjonsveiledning og prosjektrapporten. Det legges med prosjektkoden som en ZIP-fil av Android Studio prosjektet, samt en ferdig signert APK-fil av applikasjonen. Det legges også med JSON-fil av databasen.

#### 1.4.3 Metode

Prosjektet foregikk i 12 uker i perioden mellom 06.09.17 til 01.12.17. Prosjektdeltaker planlagte å benytte minst 16 timer hver uke som samlet blir 192 timer. Det ble først lagt fram flere forslag til type app, med en kort beskrivelse til hver av de. Videre ble et forslag valgt med veileder og det ble laget en utvidet prosjektbeskrivelse, med utdypende beskrivelse, funksjon og grafisk illustrasjon. Basert på prosjektbeskrivelsen ble det laget en detaljert systemdesign som beskriver hvordan appen skal fungere programmatisk, samt hvilke klasser, ressurser og eksterne biblioteker som skal benyttes. Deretter startet gjennomføringen og det ble levert en Alpha-versjon av prosjektet den 18.10.17 og en Beta-versjon av prosjektet den 08.11.17. Den 24.11.17 ble appen demonstrert for faglærer og sensor.

## Kapittel 2: Analyse og planlegging

Dette kapittelet tar for seg planleggingen av appen og en undersøkelse av hva som finnes av relatert arbeid, best praksis og relevant teknologi. Planleggingsdetaljene er slik prosjektdeltakeren forestilte seg appen, og representerer ikke sluttproduktet.

### 2.1 Hva og hvorfor

Prosjektet ble først kalt *Velbekomme*, men ble raskt endret til Matbit og var et av de første forslagene som ble framstilt av prosjektdeltakeren. En matoppskriftapp, hvor brukerne kan finne matoppskrifter, men også laste opp sine egne oppskrifter. Dette er ikke noe nytt, og apper som *MatPrat*[3] og *TVNorge mat*[4] tilbyr allerede søking av matoppskrifter, men ulik disse appene vil Matbit fokusere mer på å la brukerne definere innholdet i appen.

*En app av brukere, for brukere. For alle, uansett alder, kjønn og erfaring.*

Matbit er ikke bare et sted for matoppskrifter, men også et sted for sosiale forekomster. Et samlingspunkt for matglade mennesker som ønsker å dele og motta matkunnskaper. Prosjektet er inspirert av apper som *Facebook*[5], *YouTube*[6], *Twitter*[7] og *LinkedIn*[8] - apper som mest sannsynlig ikke hadde vært suksessfulle uten brukerne sine. Brukerne definerer appen, og hva som skal være tilgjengelig der. Denne ideologien skal Matbit forsøke å følge.

Brukeradministrasjon er hva som skiller Matbit fra de andre mat-appene i Norge. Det er opp til brukerne å bestemme hvilke oppskrifter som skal stå frem, og alle kan bidra til dette.

### 2.2 Kravspesifikasjon

Kravspesifikasjonen er en prioritert liste av appfunksjoner som prosjektdeltaker ønsker å legge til i appen.

#### 2.2.1 Matbit skal inneholde følgende:

Forside med “Dagens oppskrift!” og valgene ny oppskrift, søk og “prøv lykken” (tilfeldig oppskrift).

En søkemotor knyttet mot oppskriftene som er lagret i en database. Søkemotoren skal støtte søkestrenger som består av oppskriftstittel.

Kategorisering av oppskrifter, som grillmat, vegetar, kylling, svin, fisk. Med kategoriene kan oppskrifter filtreres i søkemotoren.

Legg til ny oppskrift - Brukeren skal ha mulighet til å legge inn nye oppskrifter, hvor han/hun kan velge å publisere den hvis oppskriften skal være tilgjengelig for alle. Oppskriftene må bestå av en tittel, tid, ingredienser, framgangsmåte. Oppskriftene må inkludere et bilde.

#### 2.2.2 Matbit *bør* inneholde følgende:

Brukerregistrering, brukerprofiler og innlogging. Med brukerprofiler kan Matbit tilrettelegge for rating, kommentering og diskusjon. Alle brukere skal ha sin egen brukerside med enkel bio, linker og profilbilde. Brukere skal kunne få oversikt over sine oppskrifter. Brukere skal kunne se nyheter om aktivitet på egenpubliserte oppskrifter.

Brukere skal kunne følge andre brukere, samt favorisere, kommentere, dele og rate eksisterende oppskrifter.

Søkemotoren støtter søkestrenger som inneholder ingredienser i oppskriften.

Søkemotoren støtter søkestrenger som inneholder brukernavn hvis brukeren ønsker å finne oppskrifter fra en spesiell person.

Notifikasjon om dagens oppskrift.

#### 2.2.3 Matbit *kan* inneholde følgende:

Filtrer oppskrifter etter tid, bruker-rating, mest sett, alfabetisk, osv.

“Spill av” stegene for oppskriften. Se ett steg av gangen og velg å gå fram og tilbake. Legg til en timerfunksjon som tar tiden på brukeren mens brukeren følger stegene.

Et poengsystem hvor brukeren scorer poeng basert på hva brukeren gjør. Dette skal oppmuntre til mer brukeraktivitet. Med poengsystem kan det legges til topplister hvor brukeren med mest poeng vises øverst. Legg til mulighet til å låse opp spesielle rammer og titler som vil vises sammen med brukerprofilen.

Støtte for linker som åpner en oppskrift eller brukerprofil direkte i Matbit fra nettleser.

Kamerastøtte, hvor man kan ta bilde av maten man har lagd, for så å gå direkte til “ny oppskrift” siden hvor bilde blir lagt inn.



Tilretteleggelse for annonser.

Legg til matpriser på all oppskriften i databasen. Hent disse matprisene direkte fra tilgjengelige ressurser. Sorter matoppskrifter basert på pris, for eksempel *under 100-lappen*.

Filtrering og rapportering av stygg ordbruk, skjellsord og trakassering, samt støtende bilder.

Støtte for flere språk.

## 2.2.4 Matbit *skal ikke* inneholde følgende:

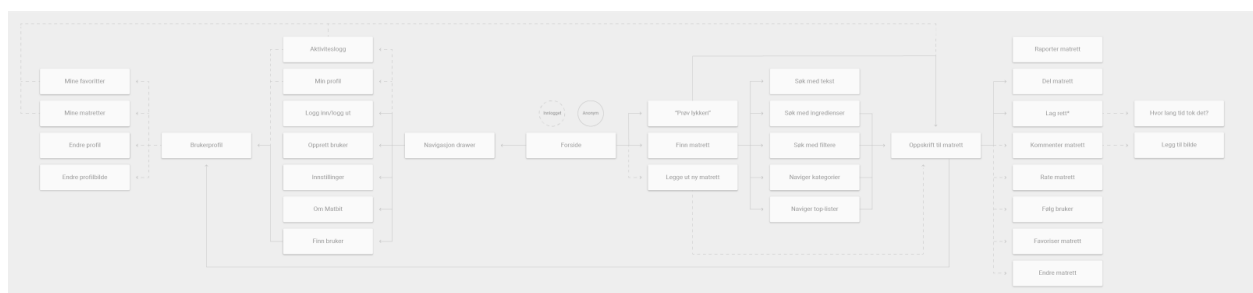
Et oppslag av kokebøker.

Ingredisenser som ikke kan fås tak i Norge.

Vanskelighetsgrad. Dette strider i mot ideologien til Matbit. Matbit er en app av alle, for alle. Hvem som helst kan lage mat; uansett hvem, uansett mat.

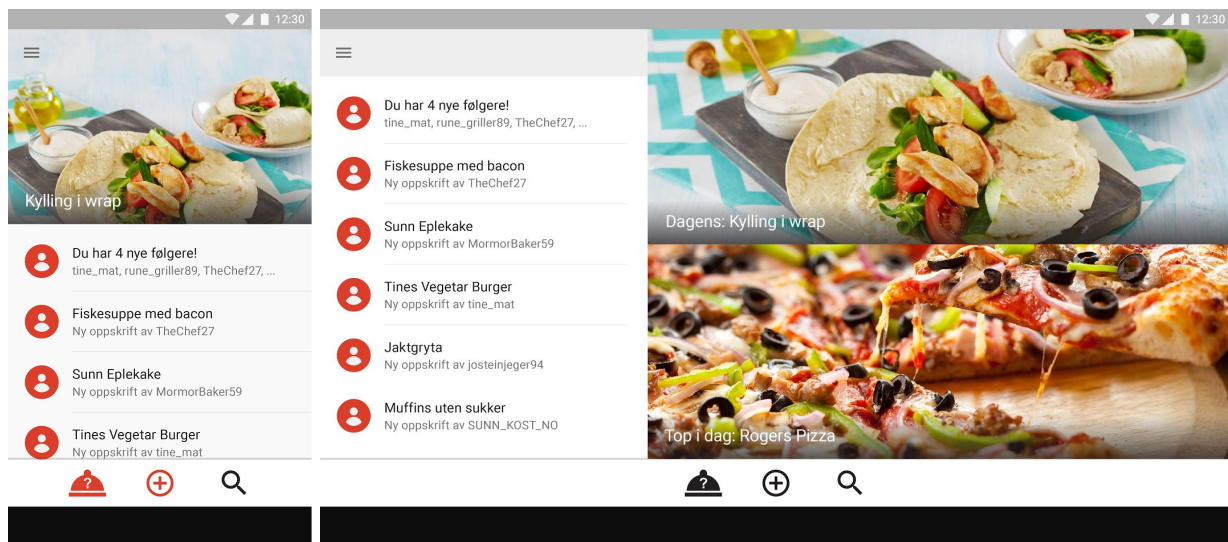
## 2.2 Oppbygging og navigasjon

Det var forventet at Matbit skulle inneholde flere sider. For å etablere en oversikt over hvilke sider som knyttes med hverandre ble det utarbeidet en navigasjonssekvens modell. Denne illustrerer hvordan brukere forventes å navigere rundt i appen. Legg også merke til at heltrukket linje er for *alle*, mens stiplet linje er bare for de som er innlogget.

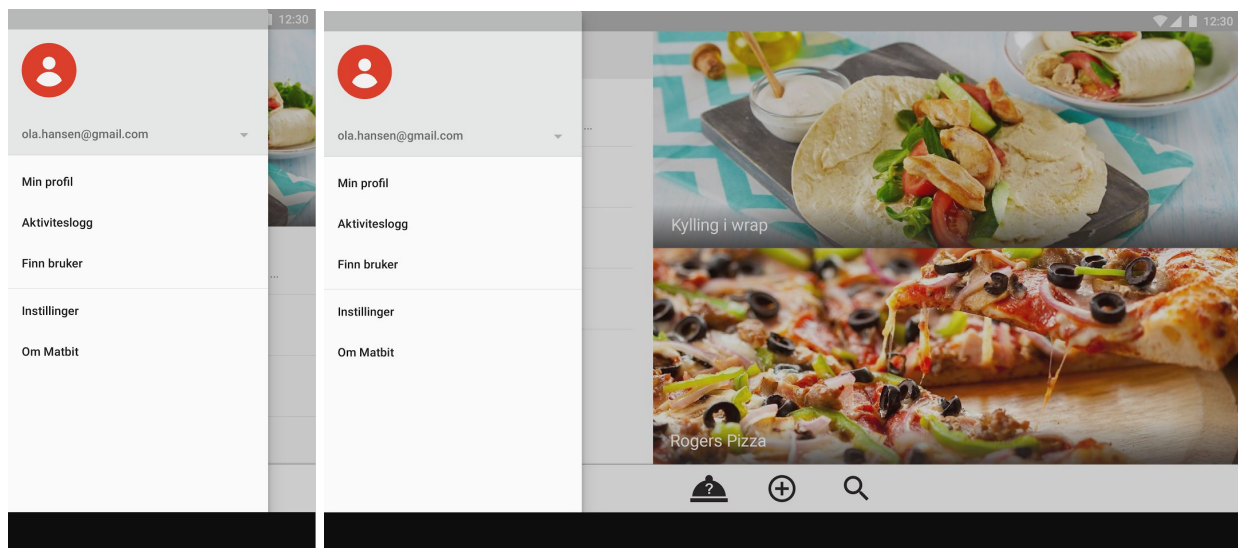


Figur 1, Vedlegg 6

Det ble også laget detaljerte skisser med bruk av ferdige maler utgitt av Google på Material Design[14]. Skissene illustrerer hovedsidene til Matbit og hvordan de planlegges å bygges opp, samt hvilke grafiske elementer som inngår. Det ble også laget en tablet-versjon av skissene.



Figur 2



Figur 3

Det første en innlogget bruker vil se når han/hun åpner Matbit er startsidene. Nye brukere vil bli spurt om å lage bruker/logge inn. På startsidene vil "Dagens rett" vises, samt en forkortet aktivitetsliste som er skreddersydd til brukeren. Lengst ned finnes tre knapper for å søke etter oppskrift, legge til oppskrift eller åpne en tilfeldig oppskrift. Ved å benytte navigasjonsbaren øverst til venstre er det planlagt at brukeren vil ha flere alternativer;

Innlogging/utlogging - Åpner en innloggingside med Google konto.

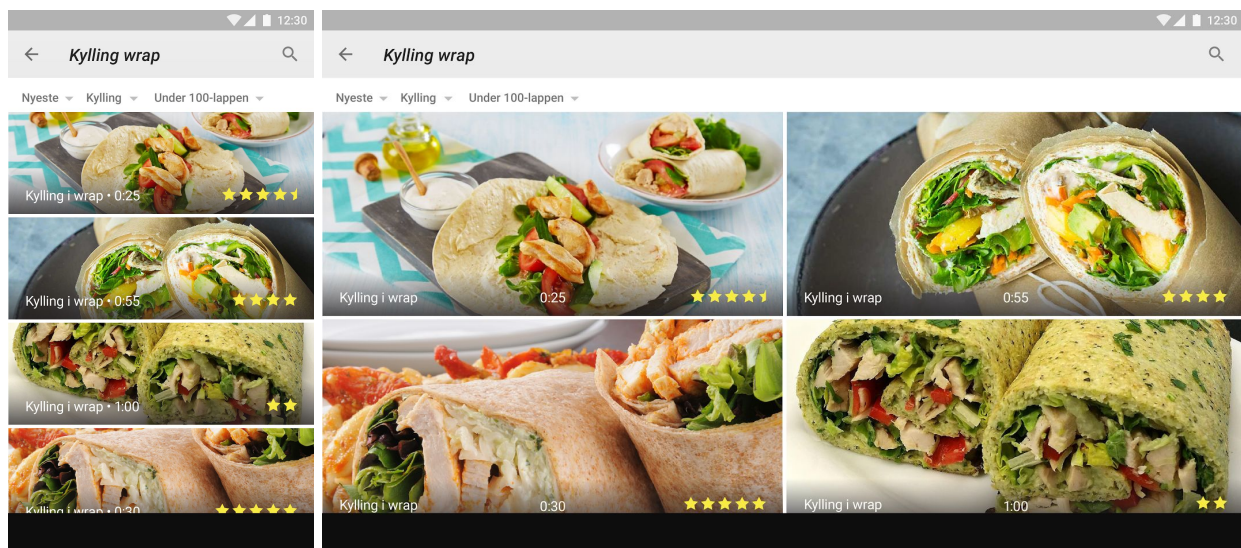
Aktivitetslogg - Åpner en ny side hvor det listes alt som har skjedd relevant til brukeren: nye følgere, nye oppskrifter fra andre brukere som brukeren følger, osv.

Min profil - Åpner profilen til brukeren.

Finn bruker - En søkemotor for å finne brukere (evt. finne dine venner fra Facebook eller lignende).

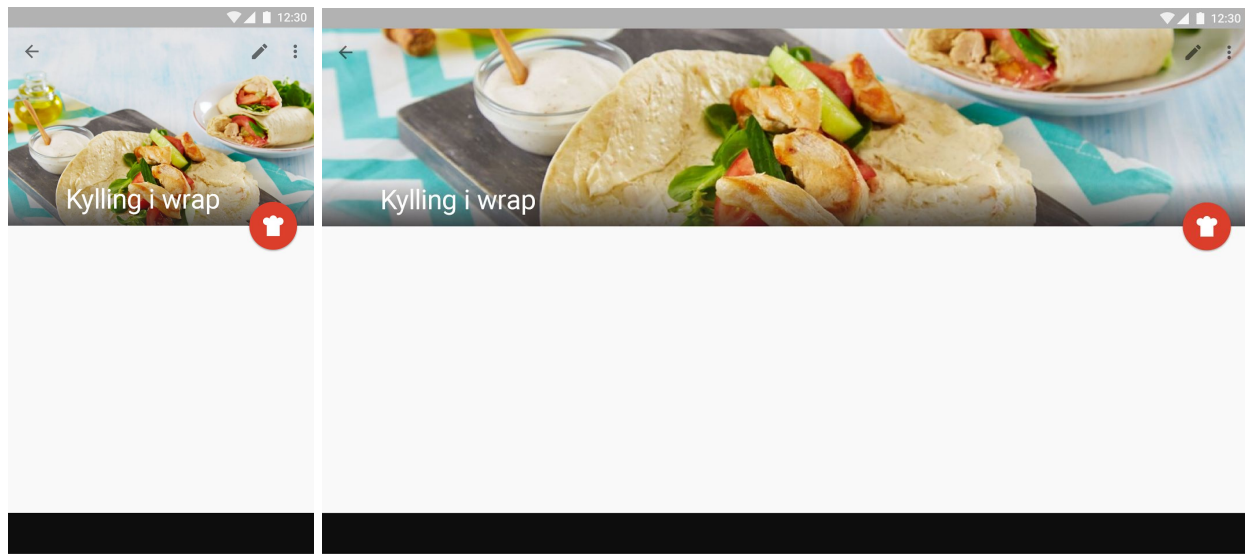
Innstillinger - Åpner Matbit innstillinger for kontroll av varsler, språk, osv.

Om Matbit - Åpner en enkel dialog som forteller brukeren om appen.



Figur 4

Hvis brukeren trykker på søk-knappen på startsiden, åpner søkesiden seg. Der har brukeren mulighet til å søke etter oppskrifter, filtrere søket og velge kategorier. Hvis brukeren bare ønsker å bruke kategorier, er dette også mulig. Hver gang brukeren trykker søk eller endrer filter eller kategori, forandres listen med oppskrifter. Listen presenteres i et *grid view*[17] med en tekstlinje, samt bruker-rating til høyre (stjerner).

*Figur 5*

Når brukeren velger en oppskrift i gridlisten fra søkesiden, vil en ID som tilhører oppskriften benyttes til å hente inn oppskrift-dataene fra databasen og sende disse dataene med brukeren til oppskriftsiden. Her vil brukeren se et større bilde av resultatet og detaljene om oppskriften; hvem som er forfatteren, hvilke ingredienser som trengs og alle stegene som må til for å lage maten. Brukeren vil også kunne se hvor lang tid det kommer til å ta, samt bruker rating. Nederst (eller via en tab) kan brukeren se kommentarer fra andre brukere, samt dele knapper, en favoriser-knapp, samt en rapporter-knapp.

I skissen er det også illustrert en knapp med en kokkelue på. Det planlegges å knytte knappen til lag-oppskrift funksjonen hvis den blir implementert. Når brukeren trykker på denne knappen, vil Matbit vise hvert steg, en om gangen. Brukeren velger selv når neste steg skal vises. Her vil det være mulighet å legge til en timer funksjon (som *CountDownTimer*) som f. eks. de gangene brukeren må sette noe i ovnen eller når noe skal heve. En “Start timer” knapp vil være tilgjengelig. Når brukeren trykker på den, vil Matbit ta tiden for brukeren og ringe når timeren går ut. Så kan brukeren fortsette til neste steg.

## 2.3 Database

Databasen skal minimum ha oppskrifter og brukere. Under legges ved tabeller som lister variablene og tilhørende datatyper som skal lagres i databasen.

Recipe	
Variable	Datatype
title	String
author	String
datetime_created	String
datetime_updated	String
info	String
category	String
time	Integer
portions	Integer
views	Long
ratings	HashMap
comments	HashMap
steps	HashMap
ingredients	HashMap

*Figur 8*

Rating	
Variable	Datatype
datetime_created	String
datetime_updated	String
rating	Integer
user	String

*Figur 9*

Comment	
Variable	Datatype
comment	String
datetime_created	String
datetime_updated	String
user	String

*Figur 10*

Step	
Variable	Datatype
string	String
seconds	Integer

*Figur 11*

Ingredient	
Variable	Datatype
course	String
name	String
amount	Double
measurement	String

*Figur 12*

User	
Variable	Datatype
nickname	String
bio	String
gender	String
exp	Long
birthday	String
signUpDate	String
lastLoginDate	String
following	HashMap(user, datetime)
followers	HashMap(user, datetime)
recipes	HashMap(recipe, datetime)
favorites	HashMap(recipe, datetime)

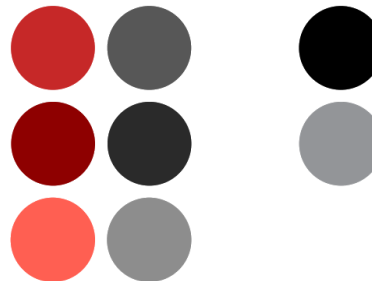
*Figur 13*

## 2.4 Design

Matbit skal være en app som appellere til folk flest. Utseende inspireres av Googles brukervennlige, lyse design. Appen skal også forsøke å følge retningslinjene spesifisert av Google Material Design[1]. Dette betyr at Matbit sitt design skal være lett-fattelig. Tidligere ble det vist forslag til design i figur 2, 3, 4 og 5. Disse forslagene følger design kriteriene satt av prosjektdeltakeren.

Prosjektdeltakeren ønsket å begrense fargevalg til 1 farge med 3 nyanser. Figur 6 under viser det endelige fargepalettet til Matbit. I artikkelen *How Color Affects Your Appetite* skrevet av Care2[15] anses varme farger som appetitvekkende farger. Blant disse fargene finnes fargen rød. Prosjektdeltakeren valgte ut fargen på grunnlag av denne informasjonen.

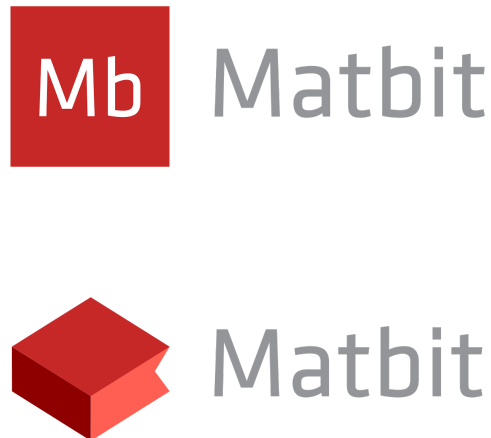
Font: Klavika Regular-Plain: Matbit  
colorPrimary: #c62828  
colorPrimaryDark: #8e0000  
colorAccent: #ff5f52



*Figur 6*

Under planleggingen ble det også utarbeidet logo til appen. Prosjektdeltakeren benyttet Adobe Illustrator[16] til å lage logo og produserte to utkast. Se figur 7.





*Figur 7*

Den nederste logoen ble valgt. Logoen skal forestille en oppskriftsbok.

## 2.5 Ressurser

Appen vil kun benytte ikonene innebygd i Android Studio og tilgjengelig på Mater Design siden[23], laget av Google Inc med Apache versjon 2 lisens[18].

Alle bilder som inngår i appen er enten eid av prosjektdeltakeren eller lagt til av brukere. Bildene lagres i en database.

Oppskriftene som legges til vil være egenproduserte og ikke hentet fra eksterne kilder. Det vil si at prosjektdeltaker vil legge til oppskrifter selv til å begynne med, og akseptere eventuelle oppskrifter som blir lagt til i test fasen.

## 2.6 Eksterne biblioteker

Prosjektdeltakeren tenkte først å benytte sin egen MySQL[19] server til lagring av oppskrift- og brukerdata, men på første veiledning foreslo veileder Adrian Jensby Sandaker å heller sjekke ut Google Firebase[20]. Firebase er en plattform som integrerer databasefunksjonalitet, autentisering, lagring og brukerinnlogging til apper laget i Android Studio. Databasen er forenklet til en stor JSON-fil hvor man kan lese og skrive data på nøkkeladresser. Følgende Firebase biblioteker må da legges til:

```
com.google.android.gms:play-services-auth:11.6.2  
com.google.firebase:firebase-auth:11.6.2  
com.google.firebase:firebase-core:11.6.2  
com.google.firebase:firebase-database:11.6.2  
com.google.firebase:firebase-storage:11.6.2  
com.google.firebase:firebase-messaging:11.6.2  
com.firebaseui:firebase-ui-storage:3.1.0
```

Prosjektdeltaker ble også tipset om å holde dato og tid til en satt standard. Dette skal lagres som en tekst streng i databasen og konverteres til en *datetime* datatype i appen. Formatet ble satt til *yyyy-MM-dd HH:mm:ss* som ble anbefalt av en stackoverflow-artikkel[21]. Det ble også tipset om å benytte JodaTime til håndtering av dato og tid. Følgende bibliotek må da inkluderes:

```
joda-time:joda-time:2.9.4
```

Tidlig i prosjektet ble det også oppdaget at Android ikke har enkle løsninger på sirkulære bilder. Det ble derfor bestemt å benytte Henning Dodenhof sin CircleImageView[22]. Følgende bibliotek må da inkluderes:

```
de.hdodenhof:circleimageview:1.3.0
```

## 2.7 Klasser

Her foreslås ulike klasser som kan måtte trenge i dette prosjektet.

Klasse for bildeopplasting, hvor klassen tar inn flere bildetyper, og konverter og komprimerer de slik at alle bilder i databasen blir lagret som jpg og ikke tar for mye plass. Det kan også lages flere størrelser av bilde for at appen skal kunne sende og motta bilder raskere fra databasen.

MatbitDatabase klasse for å etablere tilkobling til database og sende og motta data på en generalisert og strukturert måte, med feilhåndtering.

Timer klasse som skal ta tiden i bakgrunn (som en service), selv om appen lukkes.

DateUtililty klasse som håndterer datoer å passer på at disse er på samme format.

NewsFeed klasse som konstruerer nyheter som “Dagens oppskrift”, “Mest populær”, “Nyeste oppskrift”, “Nye følgere!” og så videre. Klassen leser av data fra databasen.

## 2.8 Usikkerhet

Dette er første gang prosjektdeltaker lager en android app. Hvor mye tid enkelte implementeringer tar er ukjent, og det må tas avgjørelser underveis hvilke funksjoner som skal prioriteres.

## Kapittel 3: Gjennomføring

I dette kapittelet beskrives produksjonsfasen av prosjektet. Prosjektdeltaker har lagd en prioritetsliste på hva som bør implementeres. Kapittelet utdyper hvilke metoder og verktøy som ble benyttet under utviklingen, samt hvilke valg som ble tatt på veien.

### 3.1 Oppsett

Android prosjektet opprettes og det første som må gjøres er å legge til Google Firebase. Matbit-appen registreres på Google Firebase nettsiden. I Android Studio kobles appen til Firebase via Tools > Firebase > Connect to Firebase. Denne setter opp *build.gradle* og inkluderer Firebase Core. Videre legges til resten av bibliotekene nevnt tidligere.

Så genereres en Android Keystore[24] *matbit.jks* som legges til prosjektet slik at appen kan benytte seg av bruker autorisering og login. Denne blir også benyttet til å signere APK filer som blir generert.

Så legges til en unik *SHA1 fingerprint* til innstillinger > *SHA certificate fingerprint* i Google Firebase som hentes fra *gradle signing report* loggen.

### 3.2 Database

Google Firebase databasen var viktig å få på plass først. Database strukturen ble først laget i en JSON-fil, og så lastet opp til Google Firebase. Videre ble de første linjene i oppskrift klassen skrevet, og det ble laget funksjoner for lesing og skriving av data fra oppskrift-klassen.

Dette viste seg til å være en dårlig fremgangsmåte. Flere problemer oppstod og data ble hentet ut feil og skrevet inn igjen feil. Firebase Database var fortsatt ganske nytt og uvant å jobbe med, og dette hjalp ikke prosjektet. Etter å ha studert *Realtime Database Guides*[25] på Firebase-siden ble det oppdaget en lettere måte å gå frem på. Istedenfor å definere Java-koden etter databasen var det heller anbefalt å definere databasen etter Java-koden. Det vil si at man lager en ren data klasse som representerer en JSON-blokk i databasen, og som bare inneholder datatyper tillatt av Google Firebase, samt *get* og *set* metoder til disse dataene.

#### 3.2.1 Skrive og lese data

I prosjektet ble det laget en data-klasse til alle tabellene beskrevet tidligere i planleggingsfasen: *Recipe*, *Steps*, *Comments*, *Ingredients*, *Ratings* og *Users*. Data-klassene ble benyttet direkte i skriving og lesing av data til databasen. Uten klassene ville koden ha sett ut som noe lignende:

```
databaseSnapshot.child("user").child("user_id").child("nickname").getValue(String.java);  
databaseSnapshot.child("user").child("user_id").child("bio").getValue(String.java);  
databaseSnapshot.child("user").child("user_id").child("birthday").getValue(String.java);  
(...)
```

Eksempelet over henter bruker data fra databasen. Antall linjer kode er lik antall variabler i databasen, og dette kan fort bli rotete og uoversiktlig. Med data-klassene kan koden over bli redusert til en linje kode, som vist under:

```
UserData user = databaseSnapshot.child("user").child("user_id").getValue(UserData.java);
```

Firestore fungerer slik at når en ønsker å lese data fra databasen, spesifiserer man en `DatabaseReference`[27] som er adressen til dataene man ønsker å hente ut. Den dataen som ligger under databasereferansen blir snappet opp og lagret som en `DataSnapshot`[26] (databaseavbildning). I eksempelet over ser man at brukeren blir hentet fra en databaseavbildningen med nøkkelen *user*, som er nøkkelen til alle brukere. Videre med nøkkelen *bruker\_id*, som er brukerens unike nøkkel. Et eksempel på en unik nøkkel generert av Firestore er for eksempel *k7wRLHSaRHU34Jw67mNy82*. Så benyttes funksjonen `getValue(UserData.java)` hvor `UserData.java` er data-klassen til en bruker. Her er det viktig at variabelnavn i data-klassen stemmer overens med de i databasen.

Det er verdt å nevne at eksempelet ikke representerer hvordan spesifikk data blir lest i dette prosjektet. Det er ikke nødvendig å ta databaseavbildning av hele databasen bare for å få ut data på en kjent nøkkel. En bedre måte som reduserer datastørrelsen og lastetiden er å ta databaseavbildning av den spesifikke databasereferansen. I dette prosjektet har det blitt laget ferdige funksjoner for dette i *MatbitDatabase.java*.

```
19 public class UserData {
20     private String nickname;
21     private String gender;
22     private String birthday;
23     private String signUpDate;
24     private String lastLoginDate;
25     private String bio;
26     private int exp;
27     private int num_followers;
28     private int num_recipes;
29     private Map<String, String> following;
30     private Map<String, String> followers;
31     private Map<String, String> recipes;
32     private Map<String, String> favorites;
33
34     public UserData() {
35         nickname = "";
36         gender = "";
37         birthday = "";
38         signUpDate = "";
39         lastLoginDate = "";
40         bio = "";
41         exp = -1;
42         num_followers = -1;
43         num_recipes = -1;
44         following = new HashMap<>();
45         followers = new HashMap<>();
46         recipes = new HashMap<>();
47         favorites = new HashMap<>();
48     }
```

*Figur 14*

Å lagre og sammenligne unike nøkler/ID er nødvendig for at Matbit skal fungere. Brukere, oppskrifter, kommentarer og graderinger bruker alle unike nøkler for å differensiere seg fra hverandre. I eksempelet over blir ikke den unike nøkkelen til brukeren lagret. Samtidig har ikke data-klassene andre funksjoner enn get- og set-funksjoner. Det er derfor laget en overordnet klasse for alle data-klassene som holder på data-objekter og ID til dataene.

```

19 public class User {
20     private static final String TAG = "User";
21     private String id;
22     private UserData data;
23
24     /**
25      * Default Constructor
26      */
27     public User() {}
28
29     /**
30      * Constructor
31      * @param DATA_SNAPSHOT data snapshot of specific user in Matbit database
32      */
33     public User(final DataSnapshot DATA_SNAPSHOT) { downloadData(DATA_SNAPSHOT); }
34
35
36
37     /**
38      * Download user data with data snapshot of specific user in Matbit database
39      * @param DATA_SNAPSHOT data snapshot of specific user in Matbit database
40      */
41     public void downloadData(final DataSnapshot DATA_SNAPSHOT) {;
42         this.id = DATA_SNAPSHOT.getKey();
43         this.data = DATA_SNAPSHOT.getValue(UserData.class);
44     }
45

```

Figur 15

*User.java* holder på ID/nøkkel til bruker på variabelen *id* og dataen til bruker på variabelen *userData*. Denne klassen har likhetstrekk med oppskrift-klassen *Recipe.java* i den grad at begge holder på data og knytter det mot dens respektive nøkkel i databasen. Begge har en konstruktør som tar inn en *DataSnapshot* av dataene. I figur 15 ser man funksjonen *downloadData(final DataSnapshot DATA\_SNAPSHOT)* som benytter databaseavbildningen til å initialisere *userData* variabelen i klassen med *setValue()*-funksjonen. Slik initialiseres en bruker i prosjektet.

User- og Recipe-klassen har flere dedikerte funksjoner, både eksklusive og statiske. I appen er det gjort slik at hver gang noen besøker en oppskrift, legges det til et besøkstall til det totale besøkstallet *views*. Figur 16 viser nøyaktig denne funksjonen.

```

229 /**
230  * Increment recipe views by one and update the database accordingly
231  */
232 public void addView() {
233     if (!hasViews()) {
234         Log.e(TAG, msg: "addView: Can't add views to uninitialized views");
235         return;
236     }
237     data.setViews(data.getViews() + 1);
238     uploadViews();
239 }

```

Figur 16

Her ser man at det først sjekkes om oppskriften har initialisert views. I RecipeData-klassen initialiseres alle variabler til spesifikke verdier med en standard konstruktør. Integers blir satt lik -1, Strings blir satt til tomme strenger og lister blir initialisert, men med en lengde lik 0. *hasViews()*-funksjonen sjekker om views er satt til noe annet enn -1. Hvis den inneholder en verdi lik -1, er den ikke satt, og returnerer *false*. Dette valideringsprinsippet blir benyttet for alle variabler i data-klassene slik at tom data ikke skrives over data i databasen. Dette forebygger også mot *NullPointerException* [28].

Hvis *hasViews()* returnerer *true*, økes views med 1 og blir lastet opp til databasen med funksjonen *uploadViews()*. Tilsvarende funksjoner er laget for alle data-variabler. Slik er det ikke nødvendig å skrive all oppskrift-dataen til databasen når bare en eller noen få verdier forandres. Dette gjør at appen laster data raskere, hvor det skrives bare den dataen nødvendig.

### 3.2.2 MatbitDatabase-klassen

For å samle og kontrollere spørringer mot databasen, ble det laget en MatbitDatabase-klasse som allerede har blitt nevnt noen par ganger. Klassen ble laget relativt sent i prosjektet, men var nødvendig ettersom prosjektet ble veldig stort. Dette er en statisk klasse som inneholder flere statiske variabler av typen databasereferanser, lagringsreferanser og autoriseringsreferanser, samt statiske funksjoner.

```

44 public final class MatbitDatabase {
45     private static String TAG = "MatbitDatabase";
46     private static FirebaseAuth AUTH = FirebaseAuth.getInstance();
47     private static FirebaseUser USER = AUTH.getCurrentUser();
48     private static FirebaseStorage STORAGE = FirebaseStorage.getInstance();
49     private static StorageReference RECIPE_PHOTOS = STORAGE.getReference( s: "recipe_photos");
50     private static StorageReference USER_PHOTOS = STORAGE.getReference( s: "user_photos");
51     private static FirebaseDatabase DATABASE = FirebaseDatabase.getInstance();
52     private static DatabaseReference DATABASE_ROOT = DATABASE.getReference();
53     private static DatabaseReference RECIPE_DATA = DATABASE_ROOT.child("recipes");
54     private static DatabaseReference USER_DATA = DATABASE_ROOT.child("users");
55
56     /**
57      * Refresh auth, user, database and storage references.
58      */
59     public static void refresh() {

```

Figur 17

Klassen sørger for å returnere instansene av de tre ulike Firebase-referansene. Klassen har en *refresh()*-funksjon som laster alle referansene på nytt og kjøres ved app-start og innlogging/utlogging av bruker. Uten *refresh()*-funksjonen ville ikke innlogging og utlogging av bruker fungert ettersom *FirebaseUser*[29] instansen ikke ble byttet ut med nye instanser.



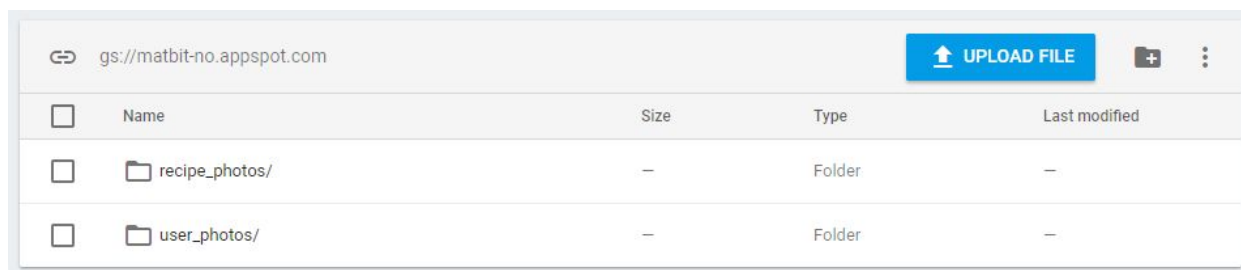
```

115 // STORAGE -----
116
117 /**
118  * Get storage reference of requested recipe photo by providing it's ID/key from the database
119  * @param RECIPE_ID ID/key of recipe from database
120  * @return storage reference of requested recipe photo
121  */
122 @ public static StorageReference getRecipePhoto(final String RECIPE_ID) {
123     return RECIPE_PHOTOS.child(RECIPE_ID + ".jpg");
124 }
125
126 /**
127  * Get storage reference of requested user photo by providing it's ID/key from the database
128  * @param USER_ID ID/key of user from database
129  * @return storage reference of requested user photo
130  */
131 @ public static StorageReference getUserPhoto(final String USER_ID) {
132     return USER_PHOTOS.child(USER_ID + ".jpg");
133 }

```

Figur 18

Dette prosjektet benytter seg av *FirebaseStorage*[30] til lagring av oppskrift bilder og profilbilder. Oppskriftbilder blir lagt en i mappe kalt *recipe\_photos* og profilbilder blir lagt i en mappe kalt *user\_photos*. Referansene til disse mappene lagres som statiske variabler til *MatbitDatabase*-klassen. Bildene blir lagret med den samme nøkkelverdien som bildet tilhører. Det vil si at hvis en bruker har nøkkelen “1234” vil bilde lagres som “1234.jpg”.



Figur 19

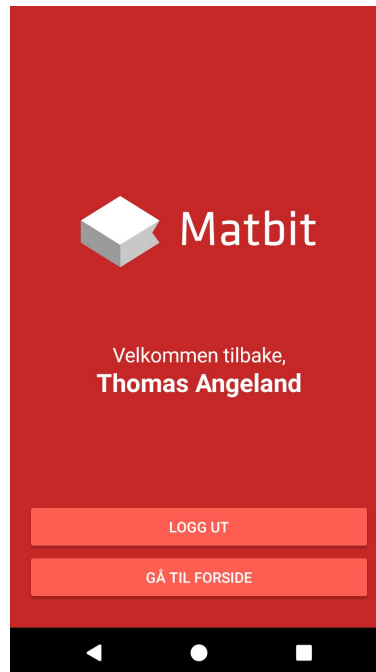
Lasting av *FirebaseStorage* bilder gjøres i egne funksjoner i *MatbitDatabase*-klassen, som for eksempel *recipePictureToImageView()*-funksjonen. Disse funksjonene benytter seg av en klasse som heter *Glide*[31] som laster bilder inn i et spesifisert *ImageView*. Så hver gang et bilde trengs å lastes fra databasen benyttes disse funksjonene.

### 3.3 Aktiviteter

Her listes hovedaktivitetene i prosjektet. Det beskrives hvordan de forskjellige aktivitetene er bygd opp og aktivitetens funksjon.

### 3.3.1 SignInActivity

*SignInActivity.java* er den første aktiviteten som møter brukerne når de starter appen. Her ser brukeren en fullskjerm med Matbit logoen, en velkomsthilsen med navnet til brukeren hvis han/hun er innlogget og to knapper. Her kan brukeren logge inn, logge ut og navigere videre til MainActivity.



Figur 20

Denne aktiviteten er laget ved å følge Google's hjelpeside for Google Logg Inn[32]. Prosjektet har begrenset innlogging til kun Google innlogging. Først konfigureres *GoogleSignInOptions* med Matbits backendserver OAuth 2.0 klient ID, hentet fra Firebase *Credentials Page*. Deretter knyttes dette til en *GoogleSignInClient* variabel. Når brukeren trykker på logg inn knappen, startes *GoogleSignInClient*-intenten og brukeren blir vist en login dialog.

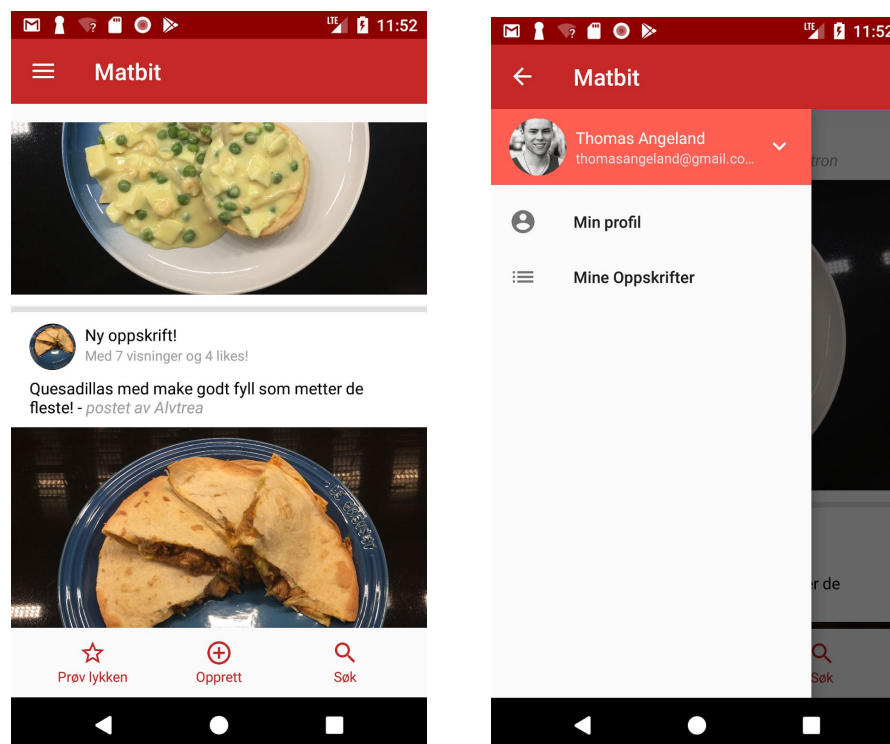
Aktiviteten vil da motta svar på funksjonen *onActivityResult()* og hente ut brukeren hvis han/hun logget inn. Deretter autentiseres brukeren til Firebase med *firebaseAuthWithGoogle()* og hvis alt går bra, er brukeren logget inn. MatbitDatabase kjører sin *refresh()*-metode som oppdaterer Firebase-instansene. Deretter oppdateres knappene slik at "Logg inn"-knappen blir gjort om til "Logg ut". Brukeren kan hentes ut ved å kalle på *MatbitDatabase.getCurrentUser()*.

Hvis det er første gang brukeren logger inn, kjøres funksjonen *MatbitDatabase.handleNewUserIfNew()*. Denne lager en ny bruker i databasen på brukerens ID. Profilbilde til brukeren lastes opp med *UploadUserPhoto.java* som er en asynkron klasse som laster ned Google profilbildet til brukeren med adressen som hentes med *MatbitDatabase.getCurrentUserPhotoURL()*. Dette profilbilde lagres til FirebaseStorage og kan endres senere. Brukeren blir tatt videre til *UserEditActivity.java* hvor han/hun kan skrive inn et kallenavn.

Etter det kan brukeren navigere videre.

### 3.3.2 MainActivity

*MainActivity.java* er hovedaktiviteten til appen. Her vil brukeren se flere oppskrifter listet under spesielle titler som “Denne ukas oppskrift”, “Mest populære oppskrift”, “Ny oppskrift”, osv. For navigering kan brukeren benytte sidemenyen (*navigation drawer*) som har tre valg: tilbake til innlogging/utlogging, min profil og mine oppskrifter. På bunnen er det ytterligere tre knapper: Last en tilfeldig oppskrift, legg til ny oppskrift og søk etter oppskrift.



Figur 21

Listen med oppskrifter er en RecyclerView[33] med en *NewsFeedAdapter* som laster inn *activity\_main\_feed\_item.xml* layouter til hver *NewsFeed* objekt som legges til i listen.

NewsFeed-objektet blir laget av en *NewsFeedBuilder*-klasse som laster inn databasens oppskrifter og finner ut hvilke oppskrifter som passer de ulike titlene:

*Denne ukas oppskrift!*

*Mest likte oppskrift!*

*Ny oppskrift!*

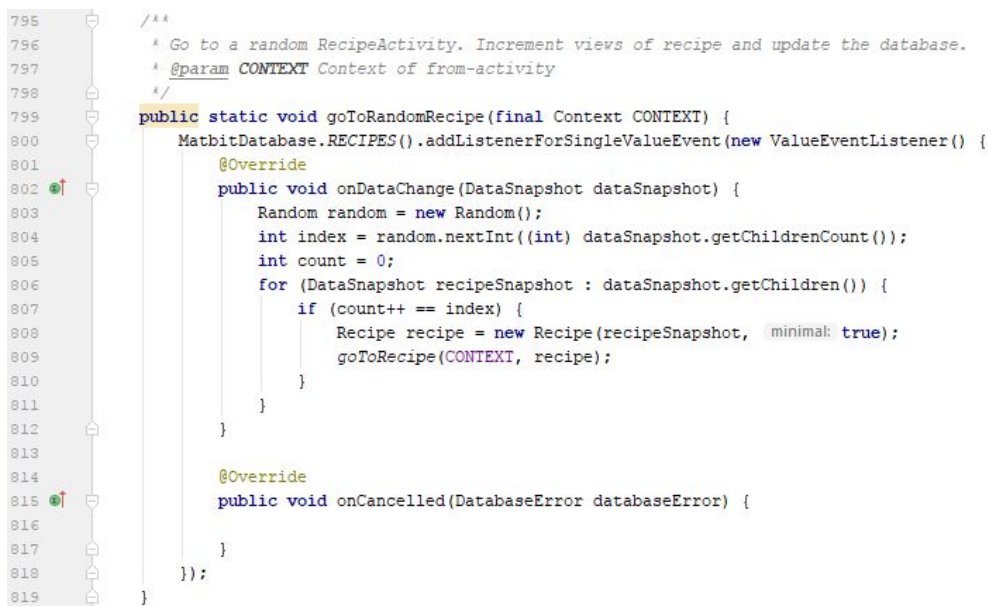
*Mest populære oppskrift!*

Når NewsFeed-objektene blir lagt til, sorteres de automatisk på dato/tid i adapteren: Adapteren lagrer NewsFeed-objektene i en *SortedList*[34] og tar inn en *Comparator*[35] som bestemmer hvordan NewsFeed-objektene skal sorteres.

Listen med NewsFeed-objektene ligger i en *SwipeRefreshLayout*[36]. Når brukeren swiper nedover på listen, lastes NewsFeed-objektene fra databasen på nytt og listen byttes ut.

Sidemenyen åpnes ved å klikke på ikonet øverst til venstre. I sidemenyen lastes brukerens Google informasjon: navn, e-post og profilbilde øverst i en *header*. Hvis brukeren klikker på denne, åpner *SignInActivity* på nytt. Hvis brukeren ikke er innlogget, vises en “Logg inn” tekst i stedet. Under finner du Min Profil så lenge brukeren er innlogget. Denne tar brukeren til *UserActivity.java*, brukerprofilen til den innloggede brukeren. Det siste i sidemenyen er Mine Oppskrifter, hvis brukeren er innlogget. Denne tar brukeren til *UserRecipeListActivity.java*, en aktivitet som lister alle oppskriftene til brukeren. Mer om brukerprofilen og mine oppskrifter kommer senere.

Nederst i appen finnes tre knapper. Knappen til venstre laster en tilfeldig oppskrift fra databasen med funksjonen *MatbitDatabase.goToRandomRecipe()*. Ettersom nøkler ikke kan hentes på nummer genereres det et tilfeldig tall mellom 0 og antall oppskrifter i databasen. Antallet hentes med funksjonen *dataSnapshot.getChildrenCount()*. Deretter itererer funksjonen gjennom oppskriftene og teller for hver oppskrift. Når telleren er lik det tilfeldige tallet, stopper iterasjonen og oppskriften hentes ut. Deretter starter *RecipeActivity.java* og laster inn denne oppskriften. Se figur 22.



```
795  /**
796   * Go to a random RecipeActivity. Increment views of recipe and update the database.
797   * @param CONTEXT Context of from-activity
798   */
799  public static void goToRandomRecipe(final Context CONTEXT) {
800      MatbitDatabase.RECIPES().addListenerForSingleValueEvent(new ValueEventListener() {
801          @Override
802          public void onDataChange(DataSnapshot dataSnapshot) {
803              Random random = new Random();
804              int index = random.nextInt((int) dataSnapshot.getChildrenCount());
805              int count = 0;
806              for (DataSnapshot recipeSnapshot : dataSnapshot.getChildren()) {
807                  if (count++ == index) {
808                      Recipe recipe = new Recipe(recipeSnapshot, minimal: true);
809                      goToRecipe(CONTEXT, recipe);
810                  }
811              }
812          }
813          @Override
814          public void onCancelled(DatabaseError databaseError) {
815          }
816      });
817  }
818  }
819  }
```

Figur 22

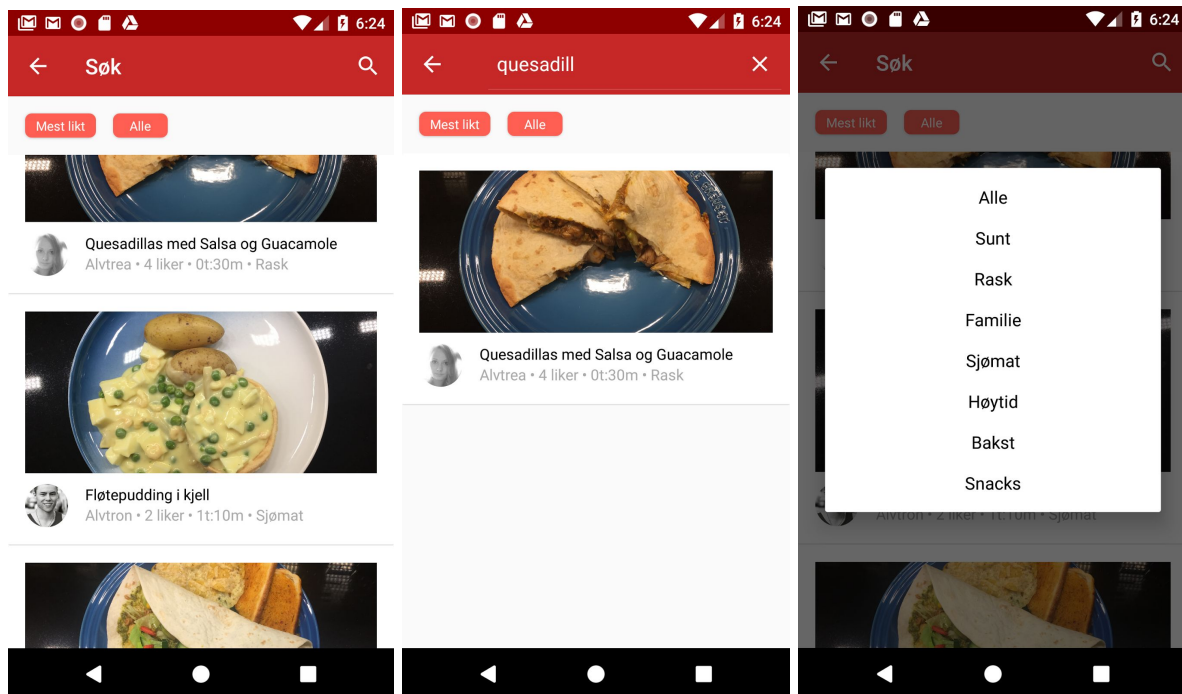
Den midterste knappen starter *AddRecipeActivity.java* så lenge brukeren er innlogget. Hvis brukeren ikke er innlogget, vises en dialog som forteller brukeren at han/hun må logge inn for å legge til oppskrifter.

Knappen til høyre starter *SearchRecipe.java*.

### 3.3.3 SearchActivity

*SearchRecipe.java* er søkesiden til aktiviteten. Her listes alle oppskrifter i databasen. Øverst kan brukeren velge filter og kategori, og søke med søkestreng.

Listen med oppskrifter er en *RecyclerView* med en *RecipeAdapter* som tar inn en *Comparator* og *Recipe*-objekter å legge de i en *SortedList*. Her benyttes også *SwipeRefreshLayout* slik at brukeren kan laste inn oppskriftene på nytt hvis ønskelig. Oppskriftene vises i hvert sitt *search\_item.xml* layout som viser oppskriftbildet, brukerprofilbildet, tittelen på oppskriften og kjapp informasjon under. Se figur 23.

*Figur 23*

Hvis brukeren klikker på profilbildet, åpner profilen til forfatteren av denne oppskriften. Hvis brukeren klikker noe annet sted, åpner oppskriften i `RecipeActivity.java`.

Filtrene øverst er begge Spinnere med en egendefinert layout. Som standard er filteret satt til “Nyeste” og kategorien satt til “Alle”. Hvis brukeren velger et annet filter vil adapteren lages på nytt, men med en annen Comparator som matcher valget til brukeren. De forskjellige Comparatorene ligger nederst i `Recipe.java`. Se figur 24. Hvis brukeren velger en kategori, blir alle oppskrifter i adapteren fjernet, og det legges til oppskrifter på nytt fra en lokal liste, hvor bare de oppskriftene med den kategorien legges til.

```

999      /**
1000       * Comparator for comparing recipe to recipe alphabetically by their title, A-Å,
1001       */
1002      public static final Comparator<Recipe> ALPHABETICAL_COMPARATOR_ASC = (a, b) -> {
1003          Locale noLocale = new Locale( language: "no", country: "NO");
1004          Collator noCollator = Collator.getInstance(noLocale);
1005          noCollator.setStrength(Collator.PRIMARY);
1006          return noCollator.compare(a.getData().getTitle(), b.getData().getTitle());
1007      };
1008
1009      /**
1010       * Comparator for comparing recipe to recipe alphabetically by their title, Å-A,
1011       */
1012      public static final Comparator<Recipe> ALPHABETICAL_COMPARATOR_DESC = (a, b) -> {
1013          Locale noLocale = new Locale( language: "no", country: "NO");
1014          Collator noCollator = Collator.getInstance(noLocale);
1015          noCollator.setStrength(Collator.PRIMARY);
1016          return noCollator.compare(b.getData().getTitle(), a.getData().getTitle());
1017      };
1018
1019      /**
1020       * Comparator for comparing recipe to recipe by their views, low to high,
1021       */
1022      public static final Comparator<Recipe> VIEWS_COMPARATOR_ASC = (a, b) -> {
1023          return Integer.compare(a.getData().getViews(), b.getData().getViews());
1024      };
1025
1026      /**
1027       * Comparator for comparing recipe to recipe by their views, high to low,
1028       */
1029      public static final Comparator<Recipe> VIEWS_COMPARATOR_DESC = (a, b) -> {
1030          return Integer.compare(b.getData().getViews(), a.getData().getViews());
1031      };
1032
1033      /**
1034       * Comparator for comparing recipe to recipe by their views, high to low,
1035       */
1036      public static final Comparator<Recipe> VIEWS_COMPARATOR_ASC = (a, b) -> {
1037          return Integer.compare(a.getData().getViews(), b.getData().getViews());
1038      };

```

Figur 24

Søkefunksjonen integreres i toolbaren til søkesiden. Her sjekkes og oppdateres oppskriftene i adapteren for hvert tastetrykk brukeren gjør. Søkealgoritmer, som ligger i *StringTools.java* er veldig enkel, men også laget selv. Se figur 25.

```

14      public final class StringUtility {
15          private static final String TAG = "StringUtility";
16
17          /**
18           * Search algorithm for searching. Very simple made and with flaws, but at least I made it myself.
19           * @param key search string
20           * @param source source string
21           * @return search string matches source string
22           */
23          public static boolean search(String key, String source) {
24              key.toLowerCase(); source.toLowerCase();
25
26              if (key.trim().length() < 4 || key == null || key.isEmpty())
27                  return true;
28
29              for (int i = 0; i < key.length() - 3; i++){
30                  if (source.contains(key.substring(i, i + 4)))
31                      return true;
32              }
33              return false;
34          }

```

Figur 25



### 3.3.4 RecipeActivity

RecipeActivity.java viser en spesifikk oppskrift, oppgitt av en nøkkel som kommer med aktivitetens Bundle[37]. Hvis nøkkelen er ugyldig, avsluttes aktiviteten. Oppskriften lastes ned med nøkkelen fra databasen. Øverst er det fire tabs som kan benyttes til å navigere de ulike sidene. Brukeren kan også swipe med fingeren til venstre og høyre. Hvis brukeren er forfatteren av oppskriften, vil et redigeringsymbol vises i toppen til høyre (se figur 28). Hvis brukeren klikker denne vil *AddRecipeActivity*-aktiviteten starte med oppskrift-nøkkelen som Bundle.

#### 3.3.4.1 RecipeFragmentInfo.java

Øverst er bilde av oppskriften. Lagt over bilde er tittelen på oppskriften. Under listes litt informasjon om oppskriften: besøkstall, lik- eller mislik knapp (kun for innlogget bruker, deaktivert ellers) og tiden. Til høyre er en lag rett knapp som åpner *CreateRecipeActivity.java*. Under er forfatteren av retten og til høyre under lag-rett-knappen er en følge-knapp. Nederst er en beskrivelse av oppskriften. Dette er den mest innholdsrike aktiviteten i appen.



Figur 26

Fire Fragments[38] initialiseres i en *RecipeTabManager.java* som er en utvidelse av *PagerAdapter*[40]. Fragmentene vises i en *ViewPager*[39] layout. Den første fragmentet, *RecipeFragmentInfo*, ses i figur 26.



*RecipeFragmentInfo.java*  
*RecipeFragmentIngredients.java*  
*RecipeFragmentSteps.java*  
*RecipeFragmentComments.java*

Lik- og misslik-knappene samsvarer med hverandre. Knappene lyser opp etter hvordan brukeren har gradert appen. Når brukeren benytter seg av disse knappene, er det en del som må oppdateres. La oss se på et eksempel hvor det er en bruker som ikke har gradert oppskriften ennå. Hvis brukeren trykker liker, vil like-knappen bli rød og databasen oppdateres. Hvis brukeren trykker liker igjen vil knappen gå tilbake til grå og databasen oppdateres. Hvis brukeren trykker på misslik-knappen, vil den bli rød og databasen oppdateres. Hvis brukeren trykker på liker nå, vil like knappen bli rød, og misslik-knappen bli grå igjen, og databasen oppdateres. Dette ble en del å holde styr på.

```

155 // If user is logged in, add onclick listeners to thumbs up, thumbs down and follow button.
156 if (MatbitDatabase.hasUser()) {
157     icon_thumbs_up.setOnClickListener(v -> {
158         // User has already rated thumbs up --> remove it.
159         if (recipe.getCurrentUserRating() == Recipe.THUMB.UP) {
160             recipe.removeUserRating();
161             txt_thumbs_up.setText(Integer.toString(recipe.getData().getThumbs_up()));
162             icon_thumbs_up.setColorFilter(ContextCompat.getColor(context, R.color.grey_500), android.graphics.PorterDuff.Mode.SRC_IN);
163         }
164         // User has already rated thumbs down --> remove it and add thumbs up.
165         else if (recipe.getCurrentUserRating() == Recipe.THUMB.DOWN) {
166             recipe.addRating(Recipe.THUMB.UP);
167             txt_thumbs_up.setText(Integer.toString(recipe.getData().getThumbs_up()));
168             txt_thumbs_down.setText(Integer.toString(recipe.getData().getThumbs_down()));
169             icon_thumbs_up.setColorFilter(ContextCompat.getColor(context, R.color.colorPrimary), android.graphics.PorterDuff.Mode.SRC_IN);
170             icon_thumbs_down.setColorFilter(ContextCompat.getColor(context, R.color.grey_500), android.graphics.PorterDuff.Mode.SRC_IN);
171         }
172         // User has not rated --> add rating.
173         else if (recipe.getCurrentUserRating() == Recipe.THUMB.NOTHING) {
174             recipe.addRating(Recipe.THUMB.UP);
175             txt_thumbs_up.setText(Integer.toString(recipe.getData().getThumbs_up()));
176             icon_thumbs_up.setColorFilter(ContextCompat.getColor(context, R.color.colorPrimary), android.graphics.PorterDuff.Mode.SRC_IN);
177         }
178     });
179
180     icon_thumbs_down.setOnClickListener(v -> {
181         // User has already rated thumbs down --> remove it.
182         if (recipe.getCurrentUserRating() == Recipe.THUMB.DOWN) {
183             recipe.removeUserRating();
184             txt_thumbs_down.setText(Integer.toString(recipe.getData().getThumbs_down()));
185             icon_thumbs_down.setColorFilter(ContextCompat.getColor(context, R.color.grey_500), android.graphics.PorterDuff.Mode.SRC_IN);
186         }
187         // User has already rated thumbs up --> remove it and add thumbs down.
188         else if (recipe.getCurrentUserRating() == Recipe.THUMB.UP) {
189             recipe.addRating(Recipe.THUMB.DOWN);
190             txt_thumbs_down.setText(Integer.toString(recipe.getData().getThumbs_down()));
191             txt_thumbs_up.setText(Integer.toString(recipe.getData().getThumbs_up()));
192             icon_thumbs_down.setColorFilter(ContextCompat.getColor(context, R.color.colorPrimary), android.graphics.PorterDuff.Mode.SRC_IN);
193             icon_thumbs_up.setColorFilter(ContextCompat.getColor(context, R.color.grey_500), android.graphics.PorterDuff.Mode.SRC_IN);
194         }
195         // User has not rated --> add rating.
196         else if (recipe.getCurrentUserRating() == Recipe.THUMB.NOTHING) {
197             recipe.addRating(Recipe.THUMB.DOWN);
198             txt_thumbs_down.setText(Integer.toString(recipe.getData().getThumbs_down()));
199             icon_thumbs_down.setColorFilter(ContextCompat.getColor(context, R.color.colorPrimary), android.graphics.PorterDuff.Mode.SRC_IN);
200         }
201     });
202 }
203
204
205
206

```

Figur 27

Til å begynne med var alle graderingene lagret i en liste i oppskriften som vist i planleggingen. Dette har ikke endret, men for å slippe å gå gjennom alle graderingene lagret i *ratings*-listen på oppskriften, ble det lagt til en teller for antall *thumbs\_up* og *thumbs\_down*. Dette gjør systemet raskere, men det må tas høyde for å oppdateres med ratings-listen.

Hvis brukeren trykker på følge-knappen, blir brukeren og forfatterens data oppdatert. Brukeren blir lagt til *follower*-listen til forfatteren og forfatteren blir lagt til *following*-listen til brukeren i databasen. Følge-knappen forandrer seg slik at den indikerer at brukeren følger forfatteren. Denne informasjonen er ikke blitt benyttet til noe interaktivt, men meningen var å gi brukeren notifikasjoner hver gang forfatteren la ut noe nytt. Dessverre var det ikke nok tid til å implementere dette.



Figur 30

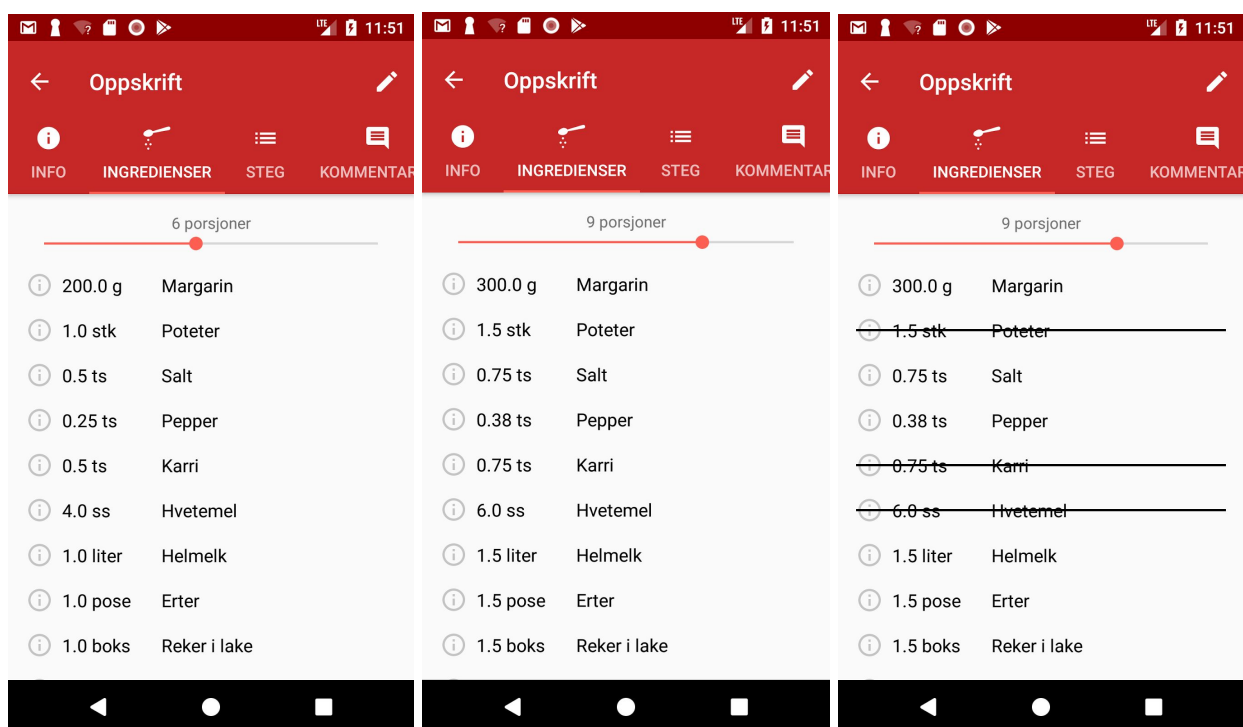
Hvis brukeren trykker på “Lag oppskrift”-knappen, vil *CreateRecipeActivity* åpnes som vist i figur 30. Denne aktiviteten er en del av *RecipeActivity* og viser hvert steg i sitt eget *Fragment*. Når aktiviteten starter vil *TimerService* ta tiden i bakgrunnen. Brukeren navigerer hvert steg med den nederste knappen.

Det er også lagt til funksjonalitet som gjør knappen i bunn om til en timer hvis steget har tid, men prosjektdeltaker rakk ikke å legge funksjonalitet for å legge til tid på steg.

Når brukeren er ferdig, vil en dialog vise den totale tiden som ble tatt i bakgrunnen. Brukeren kan velge å lagre denne tiden til databasen, men bare hvis brukeren er innlogget. Tiden vil slås sammen med tiden på oppskriften, og det vil regnes ut et gjennomsnitt. Hvis tiden er for stor eller for liten, kastes den.

### 3.3.4.2 RecipeFragmentIngredients.java

Øverst er et tekst som viser antall porsjoner. Under er en SeekBar som justerer antall porsjoner. Under er en liste over ingredienser.

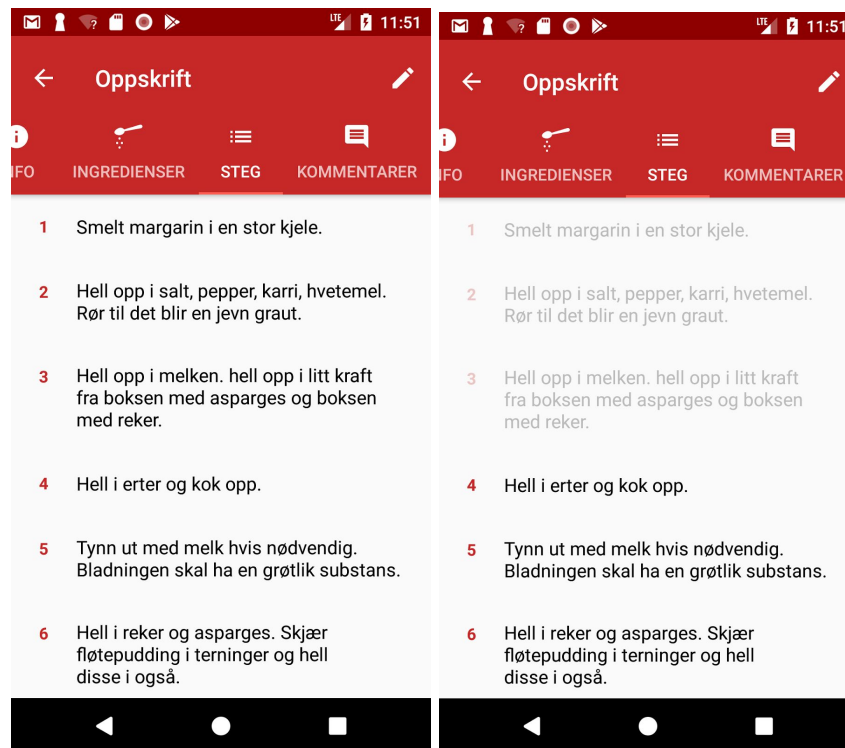


Figur 28

Ingrediensene vises i hvert sin layout som legges til i av en *IngredientAdapter.java* som settes til et ListView. Hvis brukeren justerer antall porsjoner, blir alle mengder justert automatisk. Hvis brukeren klikker på en av ingrediensene, vil den ingrediensen bli overstrøket. Hvis brukeren klikker igjen, fjernes streken.

### 3.3.4.3 RecipeFragmentSteps.java

I RecipeFragmentSteps vises en liste med alle stegene til oppskriften. Stegene er nummeret.

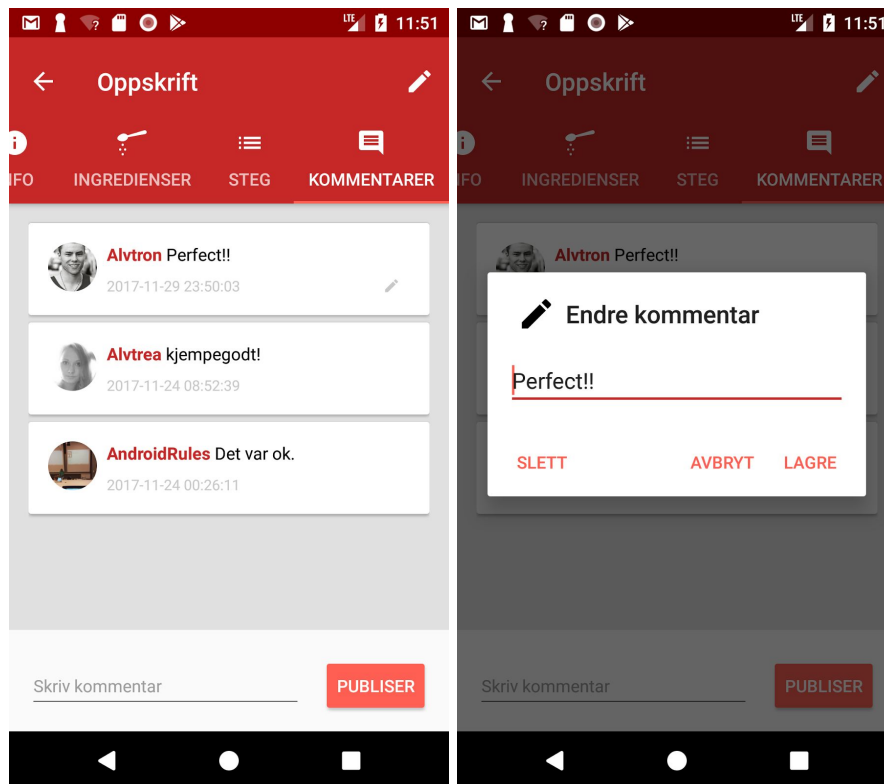


Figur 29

Stegene vises i hvert sin layout som legges til i av en *StepAdapter.java* som settes til et *ListView*. Hvis brukeren klikker på en av stegene, vil det steget bli delvis fjernet. Dette kan benyttes når brukeren ønsker å vite hvor han/hun er på steg-listen.

#### 3.3.4.4 RecipeFragmentComments.java

RecipeFragmentComments viser kommentarer lagt til oppskriften av brukere. Hver kommentar har et profilbilde, brukernavn, en kommentar tekst, publiseringsdato og et redigeringsikon hvis kommentaren er publisert av brukeren. Nederst vises skrive-feltet. Hvis brukeren ikke er innlogget, kan han/hun ikke kommentere og skrive-feltet er gjernet og det står igjen en beskjed om at brukeren må logge inn for å kommentere.



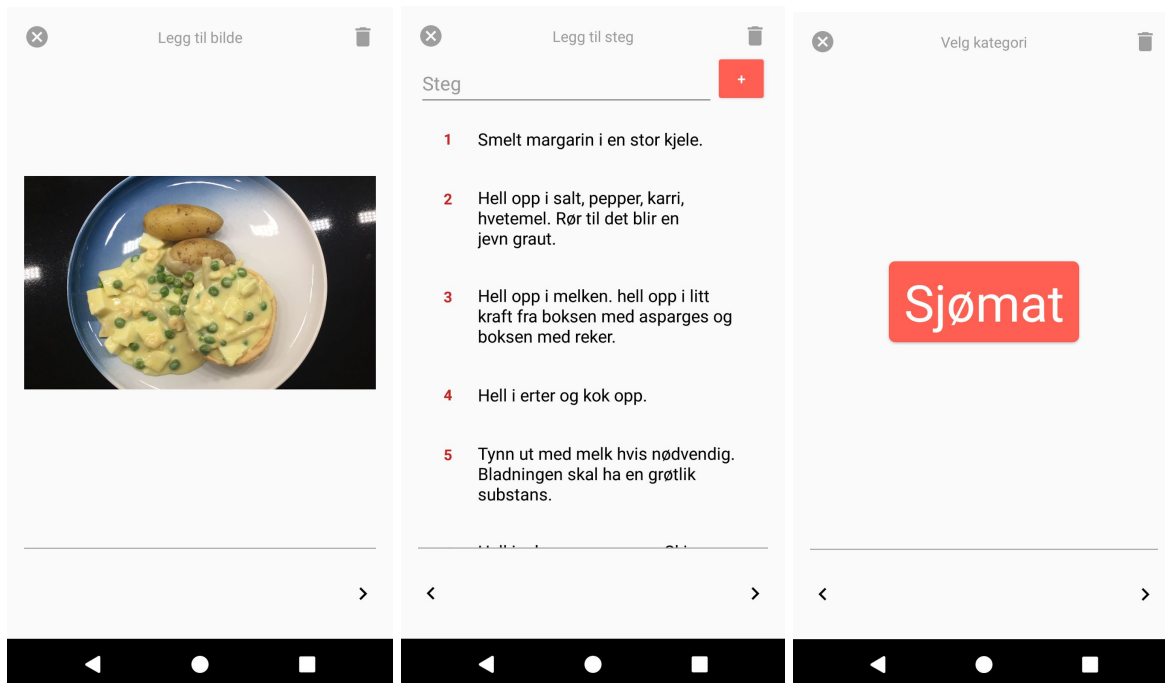
Figur 31

Kommentarene er lagret i et RecyclerView med en *CommentAdapter*. CommentAdapteren sorterer kommentarene på dato med en Comparator og en SortedList slik at den nyeste kommentaren kommer øverst. Hvis en kommentar blir lagt til, endret eller fjernet, oppdateres listen automatisk.

Hvis brukeren er forfatter av kommentaren, vises redigeringssymbolet. Når brukeren klikker denne, vil en dialog med egendefinert layout dukket opp og forfatteren kan endre kommentaren sin. Brukeren kan også slette kommentaren.

### 3.3.5 AddRecipeActivity

*AddRecipeActivity.java* er aktivitet som legger til eller endrer oppskrifter. Hvis det ligger med en Bundle med oppskrift-nøkkel betyr det at en oppskrift skal endres, ikke legges til ny oppskrift.



Figur 32

Aktiviten består av en `AddRecipePagerAdapter` som utvider `FragmentPagerAdapter`. Denne adapteren holder på 8 forskjellige Fragments som vises i en `ViewPager` layout. Hver fragment kan swipes mellom hverandre.

*AddRecipeFragmentPhoto.java*  
*AddRecipeFragmentTitle.java*  
*AddRecipeFragmentInfo.java*  
*AddRecipeFragmentIngredients.java*  
*AddRecipeFragmentSteps.java*  
*AddRecipeFragmentCategory.java*  
*AddRecipeFragmentTime.java*  
*AddRecipeFragmentPortions.java*

Den første fragmenten er `AddRecipeFragmentPhoto.java`. Den legger til eller endrer oppskriftsbilde. Når brukeren klikker på rammen i midten, vil en dialog dukke opp med tre valg: Ta bilde, velg bilde eller avbryt. Hvis bruker velger å ta bilde eller legge til fra galleriet, blir brukeren bedt om tilgang til kamera og filene til brukeren. Denne tillatelse blir forespurt via en `PermissionUtility`-klasse[41]. Denne klassen er laget av Shreya Kotak.

Når brukeren velger bilde eller tar nytt bilde, blir bildet komprimert til maks 1920x1080 piksler og kvaliteten blir halvert. Derettes lagres bilde som en byte[] array og lagres i adapteren til fragmenten.

AddRecipeFragmentTitle legger til tittel og AddRecipeFragmentInfo legger til informasjon. Begge tar inn tekst fra bruker, sjekker den og lagrer teksten til adapteren.

AddRecipeFragmentIngredients legger til oppskrifter og AddRecipeFragmentSteps legger til steg hvor begge lagres i hver sin liste som i adapteren. Listene blir vist på samme måte som i RecipeFragmentIngredients og RecipeFragmentSteps , men her kan brukeren slette oppskrifter/steg hvis han/hun klikker på de. En dialog vil dukke opp og spørre om brukeren vil slette eller avbryte.

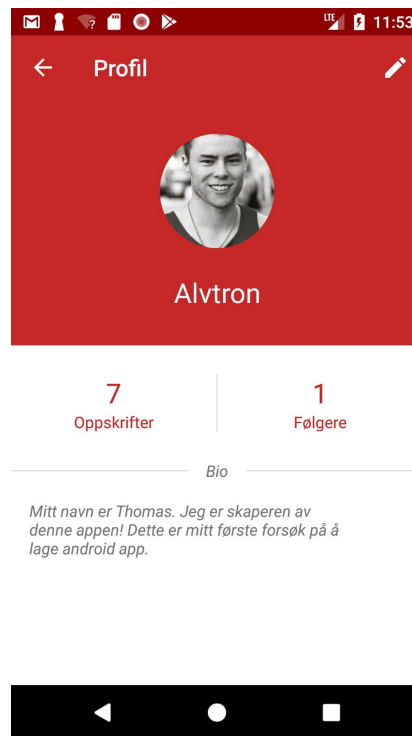
AddRecipeFragmentCategory legger til kategori til adapteren. Siden viser en stor spinner med egendefinert layout. Når brukeren velger en kategori, blir den lagret til adapteren.

AddRecipeFragmentTime legget til oppskrift tid til adapteren. Tiden blir vist i et TextView layout og hvis brukeren trykker på tiden, vil en TimePickerDialog[42] vise seg. Når brukeren velger tid, vil tiden automatisk regnes om til minutter og sendes til adapteren.

Tislutt kommer AddRecipeFragmentPortions og etter at brukeren har tastet inn antall porsjoner, kan han/hun lagre oppskriften. All input blir sjekket i adapteren og hvis noe er feil, blir brukeren meldt i fra om dette. Hvis alt er greit, genereres en ny oppskrift ID. Bildet lastes opp til FirebaseStorage og oppskriftdata til FirebaseDatabase. Brukeren blir vist en dialog med valg om å gå tilbake til hovedsiden eller til den nye oppskriften.

### 3.3.6 Brukerprofil

*UserActivity.java* er aktiviteten som viser brukerenprofiler. I midten ser man bilde av brukeren med brukerens kallenavn under. Det vises hvor mange oppskrifter brukeren har lagt ut og hvor mange følgere brukeren har. Nederst vises brukerens bio.

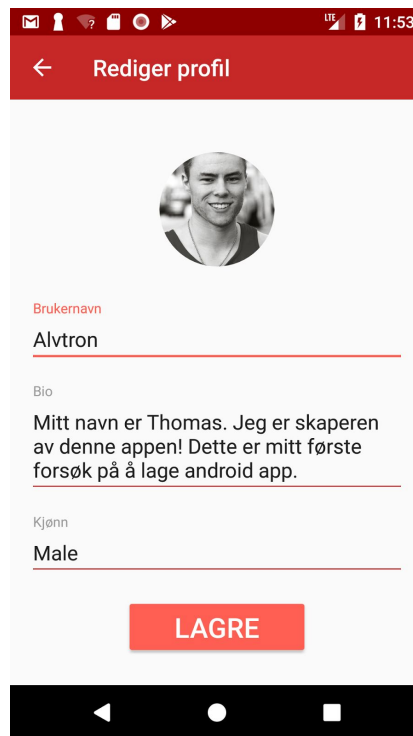
*Figur 33*

Hvis brukeren besøker sin egen profil, vil et redigeringssymbol vises i Toolbaren til høyre. Når brukeren klikker på den, vil *UserEditActivity.java* aktiviten åpne.

### 3.3.6.1 Rediger brukerprofil

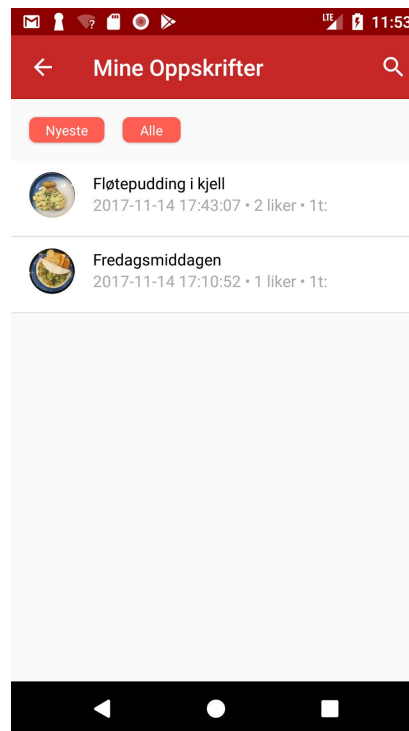
*UserEditActivity.java* gir brukeren mulighet til å endre sitt profilbilde, *nickname*, bio og kjønn. Denne aktiviten er ikke ferdig, men det ble valgt å ikke prioritere denne.



*Figur 35*

### 3.3.6.2 Mine Oppskrifter

Hvis brukeren trykker på “Oppskrifter” på profilsiden eller på Mine Oppskrifter fra sidemenyen i MainActivity, åpner UserRecipeListActivity.java aktiviten. Denne aktiviteten er veldig lik SearchActivity.class, bortsett fra at det kun vises oppskrifter som er lagt ut av brukeren. Oppskriftene vises i et egendefinert layout, *recipe\_item.xml*.



Figur 34

Lik SearchActivity kan denne også filtreres, kategoriseres og søkes. Funksjonaliteten er lik. Det som skiller denne appen er at hvis brukeren er forfatteren av disse oppskriftene, så kan han/hun slette oppskriftene direkte fra listen med en langt fingertrykk (*long click*). En dialog vil spørre brukeren om han/hun er sikker på om han/hun vil slette oppskriften. Ellers vil et vanlig trykk ta brukeren til oppskriftsiden.

### 3.4 Firebase Notifications

Matbit har mulighet for å motta notifikasjoner, uavhengig om appen er startet på brukernes android enheter eller ikke. Dette er gjort mulig ved å benytte `FirebaseMessagingService`[44] som er integrert i Google Firebase. Etter å ha fulgt en guide[43] lagt ut av Yan Zhang på `codementor.io` gikk dette bra. Det ble laget en `MatbitMessagingService`-service som utvider `FirebaseMessagingService` og en `FirebaseIDService`-service som utvider `FirebaseInstanceIdService`. Slik kan prosjektdeltakeren benytte *Notification* verktøyet i Firebase nettsiden til å sende alle brukere som har Matbit installert en notifikasjon.

### 3.5 Firebase Persistence Mode

Firebase har en funksjon[45] som gjør det slik at all data fra databasen blir lagret lokalt hos brukeren. Dette gjør det mulig å fortsette å benytte appen uten internett, så lenge brukeren ser på innhold som er lastet inn tidligere. Denne funksjon var svært attraktiv for dette prosjektet, men det kreves noe jobb og integrere den slik at den virker slik den burde, uten rare resultater. Et problem som oppstod med denne var at selv om data ble fjernet fra databasen, ble det ikke fjernet lokalt.

Prosjektet har mulighet til å skru på denne modusen, men den kommer skrudd av som standard. Hvis du ønsker å teste dette i Matbit, gå til *MatbitApplication.java* og sett

```
FirebaseDatabase.getInstance().setPersistenceEnabled(false);
```

til

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

### 3.6 Testing

Appen ble testen på flere punkter, og det ble passet på at appen ikke krasjet under forskjellige omstendigheter.

#### 3.6.1 Anonyme brukere

En viktig del av denne appen var at brukere som ikke ønsket å logge inn skal fortsatt ha lov til å lese data fra databasen og finne oppskrifter. Derfor ble appen programmert slik at man ikke trenger å være logget inn for å kunne benytte appen. Først måtte database-reglene oppdateres slik at alle kan lese fra databasen:

```
{  
  "rules": {  
    ".read": true,  
    ".write": "auth != null"  
  }  
}
```

Deretter måtte navigasjonssekvensen som ble laget i planleggingsfasen studeres slik at ikke anonyme brukere kan gjøre ting de ikke burde. Legg til oppskrift ble blokkert, og kommentering og gradering ble deaktivert for anonyme brukere.

### 3.6.2 Internettavbrudd

Ettersom dette er en app som er avhengig av internett for å vise data fra databasen, ble det testet hva som skjer når brukeren mister internett. Det ble gjort endringer i hvordan appen skriver og leser data fra databasen, samt hvordan bilder lastes. Slik appen er nå skal det gå helt fin å benytte den uten internett. Skruv du på *persistence mode* som nevnt i kapittel 3.5 skal appen også laste data lagret lokalt ved internett-avbrudd.

### 3.6.3 Test på fysiske enheter

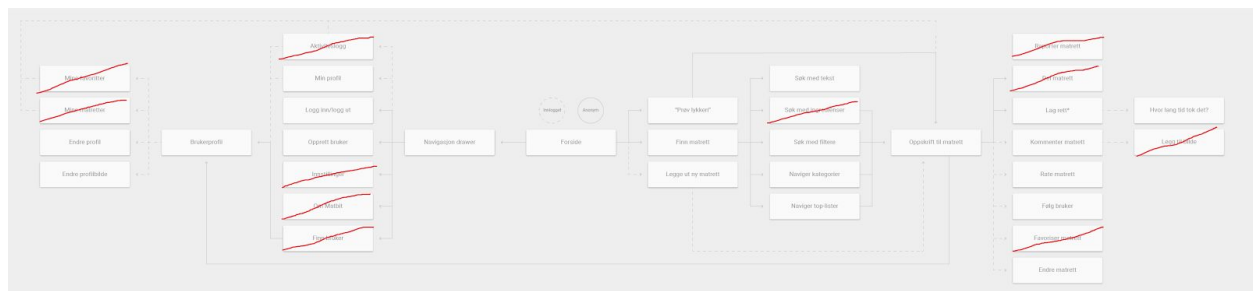
Alle appfunksjoner ble testet grundig på en fysisk enhet, altså ikke en emulator. Kamera, galleri, internett og portrett/landskap modus ble testet. Alt så ut til å fungere likt på enheten sammenlignet med emulatoren på datamaskinen.

Hvis vi ser bort fra det tekniske, var det en del designmessige avgjørelser som ble tatt på grunnlag av disse testene. Noen knapper var for små, og vanskelig å trykke med fingeren. Ulike aktiviteter skalerte ikke riktig på større skjermer. Å teste appen på en fysisk enhet hjalp å ta tak i disse spesielle tilfellene.

## Kapittel 4: Refleksjon

Det prosjektet var mer omfattende enn forventet, men også svært lærerikt. Prosjektet har utfordret prosjektdeltakeren på mange fronter, og mye av det som ble lært her kan tas med videre. Men prosjektperioden er over og det er tid til å se tilbake på prosjektet.

En rask titt på navigasjonssekvensen og kravspesifikasjonen forteller hva som ble lagt til og hva som ble droppet.



Figur 36

Det er synd at det ble aldri tid til å legge til *Mine favoritter*. Dette ville ikke ha vært så vanskelig å implementere, og det hadde vært mulighet for å benytte resirkulert funksjoner i appen. Uten denne må brukerne finne tilbake til oppskrifter de likte manuelt, og dette kan fort bli en umulig oppgave hvis databasen hadde vært mye større.

Det skulle vært lagt til innstillinger i appen som forandrer språk. Det ble lagt av litt tid til å passe på at tekst som vises i appen var hentet fra string ressursene i Android Prosjektet. Å implementere endring av språk ville vært en enkelt oppgave, hvor all tekst som vises i appen er hentet fra string ressursene i Android prosjektet.

Hvis det hadde vært tid, skulle det vært lagt til gode offline muligheter slik at appen kan brukes som normalt med lokal data hvis internettet skulle blitt avbrutt. Det ble gjort noe arbeid for å forsøke å legge til dette, men det ble nedprioritert når prosjektet begynte å slippe opp for tid.

Tilslutt hadde det vært mer attraktivt med flere oppskrifter i appen. Blant egne oppskrifter var det bare noen få som var interessante nok til å deles. Det hadde ikke vært riktig å legge ut andres oppskrifter uten deres samtykke.

Videre ville det blitt lagt tid på å legge til resten av funksjonene i kravspesifikasjonen nevnt tidligere. Før appen hadde vært klar for lansering måtte det ha blitt lagt til gode

rapporteringsmuligheter slik at brukere med dårlige hensikter blir tatt hånd om. Hovedsiden burde blitt forandret og det burde ha blitt lagt mer jobb i det sosiale aspektet med appen slik at appen føles mer levende.

## Kapittel 5: Konklusjon

Prosjektdeltakeren har med dette prosjektet produsert en komplett Android-app som kjører uten feil, og som støtter Android SDK 19 til 26. Gjennom prosjektperioden har prosjektdeltaker innhentet kunnskaper om hvordan å lage en komplett Android app, om bruk av Google Firebase biblioteker og design ved å følge Google Material Design retningslinjer.

Prosjektdeltakeren viser beherskelse av programmeringspråket Java og hvordan man lager brukergrensesnitt.

Prosjektet benytter flere eksterne biblioteker som gjør det mulig for at appen kan skrive og lese data fra database.

Det har blitt gjort flere tester for mot forskjellige situasjoner, da særlig internett tap.

Appen anses som en kreativ app på det grunnlag at appen følger sin egen ideologi inspirert av ikke-relevante referanser.

Appen benytter flere typer servicer og flere asynkrone funksjoner.

### **Fokusliste**

Av alle funksjonene i denne appen ønsker prosjektdeltakeren at sensor ser på følgende:

Kommentering: Legge til ny kommentar, slette kommentar og endre kommentar. Ha gjerne to emulatorer eller enheter oppe samtidig. Kommentaren skal oppdateres automatisk.

Legg til oppskrift / endre oppskrift: AddRecipeActivity, med alle fragmentene og adapterne som inngår, er en av aktivitetene som har tatt lengst tid og som har flest funksjoner. Test gjerne både å legge til oppskrift, endre oppskrift og slette oppskrift.

Ta gjerne en titt på RecipeActivity-klassen og MatbitDatabase-klassen. Disse klassene ble svært omfattende

Prøv å trykk på “Lag oppskrift”-knappen og se at appen tar tiden i bakgrunnen som vises mot slutten.



### **Installasjonsveiledning**

Appen krever at brukeren benytter en oppdatert versjon av Google Play Services og at brukeren har internett tilkobling. Appen fungerer uten internett, men det blir ikke mye å se på.

APK-filen inkludert i dette prosjektet installeres som normal, enten direkte ved nedlastning eller at man drar APK-filen inn i emulator-vinduet. Appen skal også installeres uten problemer på fysiske enheter og det skal ikke være nødvendig å foreta forandringer for at installasjonen skal bli vellykket og appen kjøre normalt

Android prosjektmappen inkludert som ZIP-fil åpner i Android Studio 3.0.1 på operativsystemet Windows 10. Prosjektet ble testet på 3 forskjellig datamaskiner. Det er viktig å understreke at prosjektet vil ikke kjøre som normalt i Android Studio på datamaskiner som ikke sertifiserte, altså maskiner som ikke har lagt til sin unike SHA-1 fingerprint til Matbit Firebase prosjektet. Dette er en sikkerhet krevd av Google og kan ikke skrus av.

Ta gjerne kontakt på +4741854806 hvis det er noen spørsmål.

### Referanser

- [1] Google Inc. Material Design. <https://material.io>
- [2] Google Inc (2015). Layout templates. Hentet 04.10.17 fra <https://material.io/guidelines/resources/layout-templates.html>.
- [3] MatPrat. *MatPrat*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=no.smartphones.matprat&hl=no>
- [4] FancyPants Productions. *TVNorge mat*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=com.fancypants.tvnorgemat&hl=no>
- [5] Facebook. *Facebook*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=com.facebook.katana&hl=en>
- [6] Google. *Youtube*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=com.google.android.youtube&hl=en>
- [7] Twitter. *Twitter*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=com.twitter.android&hl=no>
- [8] LinkedIn. *LinkedIn*. Hentet 30.11.17 fra <https://play.google.com/store/apps/details?id=com.linkedin.android&hl=no>
- [9] Tine. *Wrap med krydret kylling*. Bilde hentet den 04.10.17 fra <https://www.tine.no/oppskrifter/middag-og-hovedretter/kylling-og-fjarkre/wrap-med-krydret-kylling>
- [10] videogamedunkey. *Pizza*. Bilde hentet den 04.10.17 fra <https://www.youtube.com/watch?v=1X6OAucentE>
- [11] Bedre Trent. *Kyllingwrap*. Bilde hentet den 04.10.17 fra <http://www.bedretrent.no/lunsj-eller-middagstips-som-bygger-muskler/>
- [12] Bama. *Grove wraps med kylling og mango*. Bilde hentet den 04.10.17 fra <http://www.bama.no/lisesblogg/skolestart/oppskrift-grove-wraps-med-kylling-og-mango>

- [13] Klidfaster.dk. *Kålwrap med kylling og guacamole*. Bilde hentet den 04.10.17 fra <http://www.klidfaster.dk/2014/02/kalwraps-med-kylling-og-guacamole.html>
- [14] Google. *Layout templates*. Maler hentet 04.10.17 fra <https://material.io/guidelines/resources/layout-templates.html>
- [15] Care2. *How Color Affects Your Appetite*. Hentet 30.11.17 fra <http://www.care2.com/greenliving/how-color-affects-your-appetite.html>
- [16] Adobe. *Illustrator*. Hentet 30.11.17 fra <http://www.adobe.com/products/illustrator.html>
- [17] Google Developer. *Grid View*. Henter 30.11.17 fra <https://developer.android.com/guide/topics/ui/layout/gridview.html>
- [18] Apache. *Apache License Version 2*, hentet 30.11.17 fra <http://www.apache.org/licenses/LICENSE-2.0.txt>
- [19] Oracle. *MySQL*. Hentet 30.11.17 fra <https://www.mysql.com/>
- [20] Google Inc. *Firebase*. Hentet 30.11.17 fra <https://firebase.google.com/>
- [21] StackOverflow. *How to convert firebase timestamp into date and time*. Hentet 30.11.17 fra <https://stackoverflow.com/questions/38016168/how-to-convert-firebase-timestamp-into-date-and-time>
- [22] Henning Dodenhof. *CircleImageView*. Hentet 30.11.17 fra <https://github.com/hdodenhof/CircleImageView>
- [23] Google Inc. *Material Design Icons*. Hentet 30.11.17 fra <https://material.io/icons/>
- [24] Google Inc. *Android Keystore System*. Hentet 30.11.17 fra <https://developer.android.com/training/articles/keystore.html>
- [25] Google Inc. *Firebase Realtime Database Guide*. Hentet 30.11.17 fra <https://firebase.google.com/docs/database/android/start/>  
<https://firebase.google.com/docs/database/android/structure-data>  
<https://firebase.google.com/docs/database/android/read-and-write>

<https://firebase.google.com/docs/database/android/lists-of-data>

<https://firebase.google.com/docs/database/android/offline-capabilities>

- [26] Google Inc. *DataSnapshot*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/reference/android/com/google/firebase/database/DataSnapshot>
- [27] Google Inc. *DatabaseReference*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference>
- [28] Google Inc. *NullPointerException*. Hentet 30.11.17 fra  
<https://developer.android.com/reference/java/lang/NullPointerException.html>
- [29] Google Inc. *FirebaseUser*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseUser>
- [30] Google Inc. *FirebaseStorage*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/reference/android/com/google/firebase/storage/FirebaseStorage>
- [31] Google Inc. *Firebase Guides: Download Files*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/storage/android/download-files>
- [32] Google Inc. *Firebase Guides: Google Sign-In*. Hentet 30.11.17 fra  
<https://firebase.google.com/docs/auth/android/google-signin>
- [33] Google Inc. *RecyclerView*. Hentet 30.11.17 fra  
<https://developer.android.com/guide/topics/ui/layout/recyclerview.html>
- [34] Google Inc. *SortedList*. Hentet 30.11.17 fra  
<https://developer.android.com/reference/android/support/v7/util/SortedList.html>
- [35] Google Inc. *Comparator*. Hentet 30.11.17 fra  
<https://developer.android.com/reference/java/util/Comparator.html>
- [36] Google Inc. *SwipeRefreshLayout*. Hentet 30.11.17 fra

<https://developer.android.com/reference/android/support/v4/widget/SwipeRefreshLayout.html>

- [37] Google Inc. *Bundle*. Hentet 30.11.17 fra <https://developer.android.com/reference/android/os/Bundle.html>
- [38] Google Inc. *Fragments*. Hentet 30.11.17 fra <https://developer.android.com/guide/components/fragments.html>
- [39] Google Inc. *ViewPager*. Hentet 30.11.17 fra <https://developer.android.com/reference/android/support/v4/view/ViewPager.html>
- [40] Google Inc. *FragmentManager*. Hentet 30.11.17 fra <https://developer.android.com/reference/android/support/v4/app/FragmentManager.html>
- [41] Tejas Jasani. *Android Take Photo From Camera and Gallery - Code Sample*. Kodene fra artikkelen er laget av Shreya Kotak. Hentet 30.11.17 fra <http://www.theappguruz.com/blog/android-take-photo-camera-gallery-code-sample>
- [42] Google Inc. *DatePickerDialog*. Hentet 30.11.17 fra <https://developer.android.com/reference/android/app/DatePickerDialog.html>
- [43] Yan Zhang. *Sending Push Notifications to Android with Firebase*. Hentet 30.11.17 fra <https://www.codementor.io/flame3/send-push-notifications-to-android-with-firebase-du10860kb>
- [44] Google Inc. *FirebaseMessagingService*. Hentet 30.11.17 fra <https://firebase.google.com/docs/reference/android/com/google/firebase/messaging/FirebaseMessagingService>
- [45] Google Inc. *Enabling Offline Capabilities on Android*. Hentet 30.11.17 fra <https://firebase.google.com/docs/database/android/offline-capabilities>

## Figurer

- [1] Navigasjonssekvens laget av prosjektdeltakeren.
- [2] Utkast til hovedsiden til Matbit. Bildene i utkastet er hentet fra referanse 9 og 10.
- [3] Utkast til hovedsiden til Matbit. Navigasjons drawer er åpen. Bildene i utkastet er hentet fra referanse 9 og 10.
- [4] Utkast til søkesiden til Matbit. Bildene i utkastet er hentet fra referanse 9, 11, 12 og 13.
- [5] Utkast til oppskriftsiden til Matbit. Bildene i utkastet er hentet fra referanse 9.
- [6] Fargepalett til Matbit. Laget av prosjektdeltaker.
- [7] Første og andre utkast til Matbit logo. Laget av prosjektdeltaker.
- [8] Tabell av oppskrift-data i databasen. Laget av prosjektdeltaker.
- [9] Tabell av gradering-data i databasen. Laget av prosjektdeltaker.
- [10] Tabell av kommentar-data i databasen. Laget av prosjektdeltaker.
- [11] Tabell av steg-data i databasen. Laget av prosjektdeltaker.
- [12] Tabell av ingrediens-data i databasen. Laget av prosjektdeltaker.
- [13] Tabell av bruker-data i databasen. Laget av prosjektdeltaker.
- [14] Programkode fra UserData.java. Laget av prosjektdeltaker.
- [15] Programkode fra User.java. Viser variabler og konstruktør. Laget av prosjektdeltaker.
- [16] Programkode fra Recipe.java. Viser addView()-funksjonen. Laget av prosjektdeltaker.
- [17] Programkode fra MatbitDatabase.java. Viser statiske variabler. Laget av prosjektdeltaker.

- [18] Programkode fra MatbitDatabase.java. Viser get-funksjoner for lagringsreferanser. Laget av prosjektdeltaker.
- [19] Bilde av Firebase Storage til Matbit i nettleservinduet
- [20] Bilde av SignInActivity.java fra emulator.
- [21] Bilde av MainActivity.java fra emulator.
- [22] Programkode fra MatbitDatabase.java, goToRandomRecipe(). Laget av prosjektdeltaker.
- [23] Bilde av SearchActivity.java fra emulator.
- [24] Bilde av Recipe.java, ulike Comparatorer. Laget av prosjektdeltaker.
- [25] Bilde av StringTools.java, search()-funksjonen. Laget av prosjektdeltaker.
- [26] Bilde av RecipeActivity.java fra emulator.
- [27] Programkode fra RecipeActivity.java, laget av prosjektdeltaker.
- [28] Bilde av RecipeFragmentIngredients.java i emulator.
- [29] Bilde av RecipeFragmentSteps.java i emulator.
- [30] Bilde av CreateRecipeActivity.java i emulator.
- [31] Bilde av RecipeFragmentComments.java i emulator.
- [32] Bilde av AddRecipeActivity.java i emulator.
- [33] Bilde av UserActivity.java i emulator.
- [34] Bilde av UserRecipeListActivity.java i emulator.
- [35] Bilde av UserEditActivity.java i emulator.

- [36] Oppdatert navigasjonssekvens som viser hva som har blitt laget i appen. Laget av prosjektdeltaker.



### Vedlegg

- Vedlegg 1 Signert APK-fil av Matbit appen, laget av prosjektdeltaker:  
ITF21013\_Thomsa\_Angeland\_Matbit.apk
- Vedlegg 2 Signert APK-fil av Matbit appen, laget av prosjektdeltaker (backup):  
ITF21013\_Thomsa\_Angeland\_Matbit\_backup.apk
- Vedlegg 3 Android Studio 3.0.1 komprimert projektmappe:  
ITF21013\_Thomsa\_Angeland\_Matbit.zip
- Vedlegg 4 JSON-fil av databasen, slik den så ut 30.10.17 kl. 22:37:  
ITF21013\_Thomsa\_Angeland\_Matbit\_database\_301117
- Vedlegg 5 PDF av Matbit logo, laget av prosjektdeltaker:  
ITF21013\_Thomsa\_Angeland\_Matbit\_logo
- Vedlegg 6 PDF av navigasjonssekvens modell laget av prosjektdeltaker:  
ITF21013\_Thomsa\_Angeland\_Matbit\_Navigation\_Sequence
- Vedlegg 7 Utkast til design som ble laget i planleggingsfasen:  
ITF21013\_Thomsa\_Angeland\_Matbit\_Design\_Utkast

Tine. Wrap med krydret kylling. Bilde hentet den 04.10.17 fra

<https://www.tine.no/oppskrifter/middag-og-hovedretter/kylling-og-fjarkre/wrap-med-krydret-kylling>

videogamedunkey. Pizza. Bilde hentet den 04.10.17 fra

<https://www.youtube.com/watch?v=1X6OAucentE>

Bedre Trent. Kyllingwrap. Bilde hentet den 04.10.17 fra

<http://www.bedretrent.no/lunsj-eller-middagstips-som-bygger-muskler/>

Bama. Grove wraps med kylling og mango. Bilde hentet den 04.10.17 fra

<http://www.bama.no/lisesblogg/skolestart/oppskrift-grove-wraps-med-kylling-og-mango>

Klidfaster.dk. Kålwrap med kylling og guacamole. Bilde hentet den 04.10.17 fra

<http://www.klidfaster.dk/2014/02/kalwraps-med-kylling-og-guacamole.html>