Kollektiv Intelligens i Digitale Medier

Jeppe Lind Andreasen - s083456
&
Ulrik Uhre Brink - s081830

Kongens Lyngby 2012
IMM-B.Eng-2012-17

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-B.Eng-2012-17

Forord

Denne afhandling er udarbejdet ved Institut for Informatik og Matematisk Modellering på Danmarks Tekniske Universitet i opfyldelse af kravene for erhvervelse af en B.Sc. i Teknologi og økonomi - internetteknologi og økonomi-sporet.

Afhandlingen beskæftiger sig med musik anbefalinger, herunder kontekstbaserede anbefalinger samt design og implementering.

Afhandlingen består af et leveret produkt, i form af en prototype, samt en rapport til dokumentation af dette.

Vi vil gerne takke vores vejleder på DTU, Michael Kai Petersen, for tid og vejledning under projektforløbet.

Lyngby, 21-Juni-2012

Jeppe Lind Andreasen & Ulrik Uhre Brink

Resumé

Markedet for digitale medier har været stærkt stigende de seneste år og aldrig har der været så mange forskellige typer af medier at vælge imellem. Blot på markedet for digital musik, findes der millioner af sange og indholdet er stadig stigende. Dette skaber uoverskuelighed, og det kan være svært for kunderne at finde og vælge relevante sange.

Der vil i projektet blive udviklet en prototype i form af en mobil applikation med det formål, at foretage kontekstbaserede anbefalinger på nye sange, som er relevante for kunderne i given kontekst, således der bliver skabt overblik på et ellers uoverskueligt marked.

Personalisering af nye anbefalede sange er en længere proces, som kræver mange forskellige typer af delelementer, der alle er afhængige af hinanden. I den bagvedliggende logik i applikationen er der lagt fokus på at udlede en brugers kontekst ud fra implicit feedback fra en smartphones indbyggede sensorer. Ved at knytte kontekstuelle parametre til den musik en bruger lytter til såsom; bevægelse i form af: stilstand, gang, løb eller transport, støjniveau angivet som højt eller lavt samt lokation ud fra omgivelsernes WiFi Access Points, vil det forsøges at foretage mere specifikke og kontekstbaserede anbefalinger på ny musik ud fra en given kontekst.

En brugers bevægelse vil blive udledt vha. accelerometeret, støjniveauet vha. mikrofonen og lokation udledt af den indbyggede WiFi scanner. Prototypen vil udnytte musiktjenesten Spotify som musikbibliotek og danne anbefalinger ud fra de afspillede sange, som brugeren herefter kan afspille gennem Spotify.

De afspillede sange fra Spotify vil danne grundlag for anbefalinger, som i prototypen vil blive genereret via Echo Nest, der er en intelligent musikplatform, der indeholder information på 30 millioner sange og 1,5 millioner kunstnere verden over.

Prototypen vil som udgangspunkt blive udviklet til Android-platformen, men mulighederne for andre platforme såsom iOS og Windows Phone er til stede og er gjort mulige i prototypen.

En prototype af applikationen er blevet publiceret på Google Play og er gennem udviklingsprocessen blevet testet af fem personer. Prototypen kan med stor succes knytte kontekstuelle parametrer til afspillede sange og præsentere nye anbefalede sange, som er relevante for brugeren.

Indhold

Kapitel 1

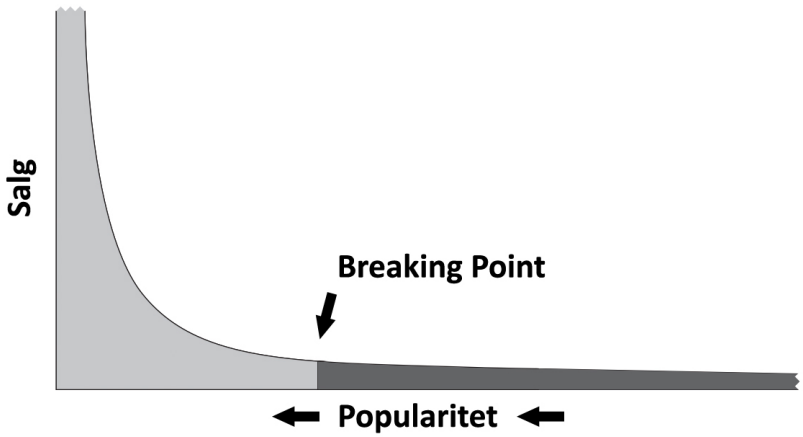
Indledning

Særligt to teknologier har været med til at ændre den måde hvorpå musik tilgås i dag, internettet samt digitaliseringen af medier. Internettet har gjort det muligt, nemt og hurtigt at finde relevant information omkring alverdens emner. Internettet har derfor været en vigtigt faktor i skiftet fra det fysiske til det digitale medie, hvor lagring på fysiske medier er blevet erstattet med datacentre og computere verden over. Digitaliseringen af medier har fjernet de omkostninger, der tidligere var forbundet med producering af medier. Omkostninger forbundet med den fysiske "hyldeplads" er blevet minimeret eller helt elimineret. De første online musikbutikker, heriblandt iTunes Music Store[23], kom frem i 2003 og i dag eksisterer et væld af forskellige online musikbutikker.

Fysiske butikker har fysiske rammer for, hvor meget hyldeplads der er tilgængelig. Dette resulterer i, at butikker kun hjemtager musik, som aktivt bidrager til butikkens omsætning - hitlisterne. Online musikbutikkerne kan tilbyde brugeren et langt større udvalg end de fysiske butikker, da hyldepladsen er ubegrænset og de dermed også har plads til niche-produkterne.

Salget har ifølge Chris Anderson[1] fulgt 80/20 reglen, hvor 20% af musikken stod for 80% af omsætningen. Dette er efter online musikbutikkernes samt streaming tjenesternes fremtræden ikke længere gældende, hvor mønsteret i stedet følger The Long Tail begrebet, der er illustreret på figur 1.1. Omsætningen i de fysiske butikker stopper ved breaking point grundet manglende hyldeplads, hvilket er resultatet af de manglende niche-produkter. Derimod har online musikbutikker og streaming tjenester skabt et stort for salg af niche-produkterne indenfor musik, som selv med få køb bidrager positivt til omsætning grundet de lave omkostninger.

På baggrund af den massive stigning i datamængden findes der nu så meget musik, at markedet er blevet uoverskueligt for den enkelte bruger, hvor det er



Figur 1.1: Illustration af The Long Tail.

blevet vanskeligt at finde relevant musik blandt det enorme udbud, der idag er tilgængeligt. Alene streaming tjenesten Spotify har et udbud på mere end 15 millioner sange. Denne udfordring vil blive forsøgt afhjulpet med en mobilapplikation, hvis hovedfokus er at foretage personaliserede anbefalinger på ny musik, som er relevant for den enkelte bruger. Anbefalinger vil ske på baggrund af den enkelte brugers kontekst, da dette antages at have stor indflydelse på valg af musik. Der vil endvidere blive fokuseret på at udlede konteksten vha. implicit feedback, for dermed at gøre det nemmere for den enkelte bruger, at finde relevant musik.

Kapitel 2

Markedsanalyse

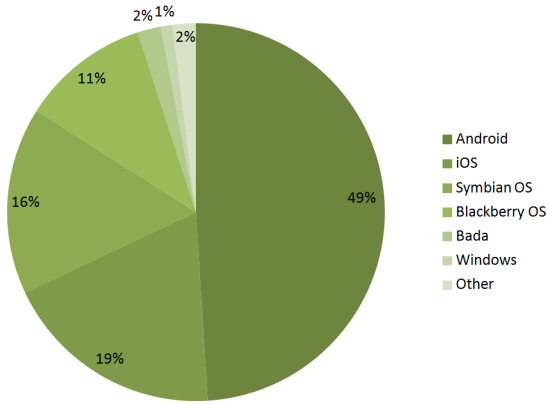
Inden udviklingen af et nyt produkt påbegyndes er det vigtigt at forstå markedet samt dets forskellige platforme og konkurrenter. Følgende vil derfor være en analyse af markedet og dets konkurrenter samt en evaluering herpå, således at der skabes overblik over, hvordan et nyt produkt vil kunne differentiere sig.

2.1 Mobile platforme

Der findes i dag flere forskellige markeder for mobile applikationer, idet der findes forskellige mobile platforme heriblandt iOS (iPhone), Android, Windows Phone, Symbian OS og Blackberry. Der vil i dette projekt kun blive fokuseret på iOS og Android, da dette er de to største platforme på markedet.

2.1.1 1 ud af 2 smartphones er en Android

Figur 2.1 viser, at der i 2011 fandtes flere Android smartphones på markedet fordelt over hele verdenen. Selvom Android med 49% har en større markedsandel end Apple's iOS på 19%, findes der dog stadig flere applikationer til iOS. Apple's App Store indeholder i skrivende stund over 540.000 mod Androids 350.000. Herudover tyder det på, at den enkelte iOS bruger er mere købestærk, da artiklen fra CNN [8] netop viser, at fortjenesten pr. applikation er højere på iOS end den er på Android, idet hele 13,5% af alle hentede applikationerne på iOS er betalte applikationer, hvor det på Android kun er 1,3%. Det vil derfor alt andet lige være mere lukrativt for udviklere, at udvikle til iOS-platformen.



Figur 2.1: Figuren illustrerer de forskellige platforme og deres markedsandele. Kilde: http://www.visionmobile.com/blog/2012/02/infographic-100-million-club-top-smartphone-facts-and-figures-in-2011/

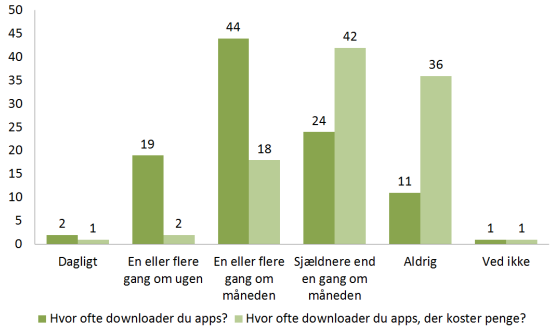
2.1.2 Forretningsmodeller for mobile applikationer

Der findes helt overordnet to typer af applikationer og herunder forskellige forretningsmodeller for applikationer.

2.1.2.1 Betalingsapplikationen

Betalingsapplikationer, er selvsagt, at den enkelte bruger betaler for at hente applikationen i App Store eller via Google Play. Prisen for de udbudte applikationer ligger, på begge platforme, i intervallet kr. 6-1200. Fælles for dem er ligeledes, at de beregner sig en afgift på 30%. En tredjedel lyder umiddelbart ikke af meget set i forhold til applikationspriser på eksempelvis kr. 6. Bliver de bestsellere, vil det være en betydelig sum penge, som både Apple og Google får ved blot at distribuere applikationerne.

Selvom 30% af salget går direkte til Apple og Google, er fordelen ved betalingsapplikationen dog stadig, at den enkelte bruger betaler for selve applikationen. Dette vil alt andet lige give en større fortjeneste på selve applikationen. Derudover bliver betalingsapplikationer i App Store favoriseret fremfor gratisapplikationerne. Dette ses tydeligt på figur 2.3, hvor betalingsapplikationer bliver præsenteret ved opstart. Hitraten/klikraten er dermed også højere end ved gratis-applikationerne, hvor brugeren aktivt skal vælge listevisningen med gratisapplikationerne.



Figur 2.2: Figuren illustrerer fordelingen af downloadede apps fordelt på gratis/betaling.
Kilde: http://fdb.dk/analyse/vi-har-smartphones-%E2%80%93-nu-vil-vi-have-tablets

Den grundlæggende tanke bag Apple's favorisering af betalingsapplikationer, må findes i den indtjening de får ved salg af betalingsapplikationerne.

2.1.2.2 Gratisapplikationen

Gratisapplikationer kan brugeren hente kvit og frit, og dermed bliver disse også hentet oftere end betalingsapplikationer, hvilket analysen fra FDB illustrerer på figur 2.2. Her ses det, at kun 2% af danskerne henter betalingsapplikationer en eller et par gange om ugen, 18% henter en eller flere gange om måneden, mens 42% kun henter en gang om måneden. Hvorimod fordelingen for hentede gratis-applikationer er helt anderledes med henholdsvis 19%, 44% og 24%. Brugerne er dermed glade for at hente gratisapplikationer.

Selvom en applikation udbydes som en gratis applikation kan denne dog sagtens indtjene penge. Grundlæggende er der tre mulige metoder, som kan anvendes ved indtjening på en gratis applikation.

- Anvendelsen af reklamer i applikationen.
- Sælge de data applikationen indhenter over tid.
- Tilbyde en "freemium" applikation, hvor grundlæggende funktionalitet er gratis og ekstra features kan tilkøbes.

Den mest benyttede metode er sandsynligvis anvendelsen af reklamer, da dette

Figur 2.3: Figuren viser en gratis appplikation med reklame indbygget (tv.) samt oversigt over betalingsapplikationer i Apple's App Store (th.).

er en god og sikker forretning. Dybest set indsættes reklamerne således, at de konstant præsenteres for brugeren eller i nogle tilfælde direkte overlapper applikationen. Indtjeningen sker ud fra genererede klik - uanset om disse måtte være oprigtige eller tilfældige, hvilket er illustreret på figur 2.3. Fordelen ved reklamerne er, at enhver bruger er en potentiel indtjeningskilde. Der kan altså tjenes penge, hver gang brugeren benytter applikationen, hvorimod indtjeningen ved betalingsapplikationer kun sker ved selve købet. Sandsynligheden for at gratisapplikationen bliver hentet er desuden større, som tidligere illustreret på figur 2.2.

Ulemperne ved reklamerne er dog, at brugeren ofte bliver irriteret og ser reklamerne som et forstyrrende element. Ved forkert eller overdreven brug af reklamer, kan det derfor få brugeren til at afinstallere applikationen og brugeren vil dermed være en tabt indtjeningskilde.

Alternativt kan reklamerne helt udelades og indtjeningens muligheden vil bestå af de indsamlede data. Dette kræver naturligvis, at den indsamlede data er relevant for en tredjepart, samt at datagrundlaget er i en sådan mængde, at dette kan analyseres og danne grundlag for ordentlige antagelser af eventuelle trends og mønstre. Twitter, et socialt informationsnetværk, er et glimrende eksempel på videresælgelse af data. Ifølge artiklen [9] har Twitter i Storbritannien omkring 7 millioner brugere og undersøgelsen viser, at britiske brugere tror deres tweets[1], efter en uge, ikke længere er tilgængelige grundet den manglende visning ved fremsøgning på hjemmesiden. Virkeligheden er en anden, da Twitter gemmer enhver af de ca. 250 millioner tweets, som bliver skrevet hver dag. Twitter har

[1] Beskeder der sendes gennem Twitter, som er offentlige

indgået en aftale med virksomheden Datasift, der har fået adgang til enhver tweet siden 2010. Datasift kan bruge disse tweets til fx at finde aldersmæssige og demografiske trends og mønstre, hvilket de benytter til at målrette marketings kampagner og reklamer for andre betalende virksomheder.

Ulempen ved at videresælge brugerdata er dog, at virksomheden kan blive yderst upopulær, blandt de selvsamme brugere, som danner forretningsgrundlaget. Nick Pickles, direktør i Big Brother Watch, udtaler i artiklen [9], at brugerne ofte ser deres tweets, som værende personlige. Efter aftalen mellem Twitter og Datasift lader det dog ikke til, at brugernes tweets er privat ejendom og en god pointe er udtagelsen - når en bruger anvender et gratis produkt, såsom Twitter, er brugeren ikke længere kunden, men selve produktet, der bliver solgt videre.

Der kan i stedet vælges en anden, og mindre upopulær, tilgang til videresælgelsen af brugerdata, hvori tredjeparts selskaber ikke får direkte adgang til specifik brugerdata, men blot opsummerede data. Dette kunne eksempelvis være data eller fakta om brugerens vaner, trends der rører sig indenfor musikken, samt hvordan brugerens musikadfærd ændre sig over tid mm. Ligeledes kunne en forretningsmodel som Facebooks benyttes, hvor data ikke bliver solgt, men virksomheden blot tilbyder målrettet marketing til andre virksomheder, som en service.

2.2 Konkurrenter

Det er vigtigt med et godt kendskab til eventuelle konkurrenter, når et nyt produkt skal udvikles. Nedenfor følger derfor en gennemgang af de konkurrenter der anses, som de største i forhold til det nye produkt.

Moodagent[16] (iOS/Android) giver mulighed for, at finde anbefalede sange blandt en brugers købte musik, og danne afspilningslister på musik, der passer til en brugers humør.

SpotON Radio[21] (iOS) giver mulighed for, at få dannet afspilningslister på anbefalede kunstnere ud fra en valgt kunstner, hvor musikken bliver streamet fra en brugers Spotify-konto.

Soundrop[19] (iOS) er den sociale jukebox, hvor den enkelte bruger aktivt deltager i valget af det musik, som bliver afspillet. Brugeren har mulighed for at afgive sin stemme på sange og dermed få dem højere op på listen. Musikken bliver streamet fra en bruger Spotify-konto.

Jog.fm[11] (iOS) analyserer tempoet i en brugers købte musik og afspiller musik der passer til en brugers tempo under fx. en løbetur.

Last.fm[14] (iOS/Android giver anbefalinger på ny musik baseret på tidligere afspillede sange. Musikken bliver streamet fra en brugers Last.fm-konto.

Nike+ iPod[4] (iOS) giver mulighed for, at afspille musik fra afspilningslister der passer til en brugers aktivitet og tempo. Musikken kan købes i iTunes, hvor forskellige afspilningslister til forskellige aktiviteter er nøje udvalgt af professionelle sportsudøvere.

Slacker Radio[18] (iOS/Android) giver mulighed for, at lytte til forskellige musik-kanaler, hvor musikken er sammensat af eksperter. Musikken bliver streamet fra en brugers Slacker Radio-konto.

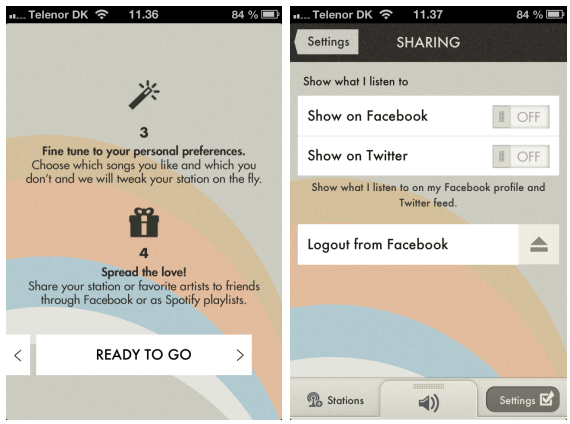
2.3 Viral markedsføring

Der findes mange typer af modeller og løsninger indenfor markedsføring af et produkt. I dette projekt er der lagt fokus på anvendelsen af viral markedsføring, hvor der gøres brug af Web 2.0.

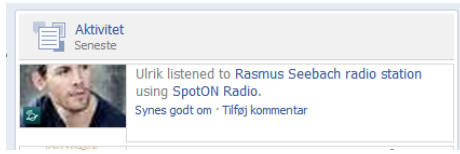
Målet med denne type markedsføring er, at drage nytte af en brugers sociale netværk til at sprede budskabet omkring applikationen. Personer er efterhånden blevet immune overfor almindelige reklamer, der vises overalt i applikationer, på hjemmeside etc., og en bruger vil derfor alt andet lige være mere modtagelig overfor en reklame, hvis den kommer fra en ven i det sociale netværk. Som artiklen [24] også beskriver det, så er der tale om en ny anvendelse af den gamle mund-til-mund kommunikation, hvor internettet anvendes som kommunikationsled til at få videreført og spredt budskabet.

Anvendelsen af denne type markedsføring går igen i mange mobile applikationer, hvor det er gjort nemt for brugeren at dele enten applikationen eller informationer fra applikationen direkte på forskellige sociale medier, herunder Facebook og Twitter. På figur 2.4 ses et godt eksempel på, hvordan viral markedsføring kan anvendes i en mobil applikation. SpotON applikationen giver mulighed for, at en bruger kan aktivere og dermed også dele, hvad brugeren afspiller på henholdsvis Facebook og Twitter. Fordelen ved dette er, at brugeren nu reklamerer for applikationen til alle brugerens venner på de sociale netværk. Et eksempel på hvordan reklamen fra SpotON vises på Facebook er illustreret på 2.5.

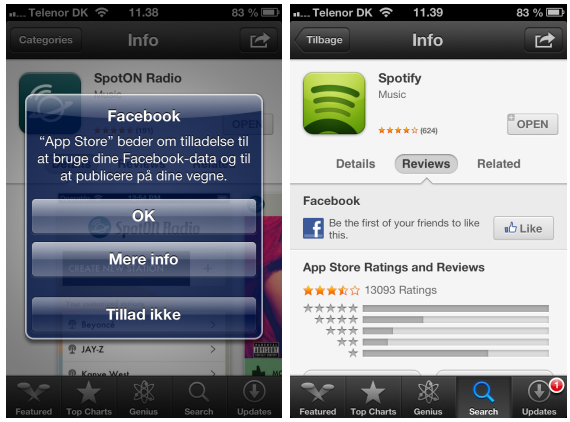
Ikke kun udviklere af mobile applikationer har set mulighederne indenfor anvendelsen af denne type markedsføring, hvor funktionen nu er blevet bygget ind i Apple's nye mobil styresystem, iOS 6. iOS 6 har blandt andet gjort det muligt, gennem App Store at kunne "synes godt om" den enkelte applikation, som er



Figur 2.4: Figuren illustrerer anvendelsen af viral markedsføring i SpotON Radio.



Figur 2.5: Figuren illustrerer resultatet af viral markedsføring af SpotON Radio på Facebook



Figur 2.6: Figuren illustrerer tilladelse til publicering på Facebook

Figur 2.7: Figuren illustrerer en eventuel anvendelse af viral markedsføring i den kommende applikation.

illustreret på figur 2.6, hvor en bruger skal give App Store tilladelse til at publicere en status opdatering på Facebook. Ved godkendelse kan App Store nu foretage publiceringer på vegne af brugeren og tilkendegive, at brugeren "synes godt om" eksempelvis Spotify.

Der kan med fordel anvendes viral markedsføring i applikationen og et eksempel på Twitter er illustreret på figur 2.7, hvor en bruger er blevet anbefalet en sang og den herefter er blevet tweetet[2] ud til alle der følger brugeren. I tweeten autogenerer applikationen en tekst "Jeg er blevet anbefalet" af den ene årsag, at gøre tweeten mere personlig og dermed minde mindre om en reklame. Herefter bliver der udskrevet, hvilken sang der er blevet anbefalet samt et tag[3], og til slut en opfordring til andre personer samt et link til applikationen på Google Play.

[2] Tweet: En besked på 140 tegn på det sociale netværk Twitter
[3] Enkelt ord, der beskriver beskeden

2.4 Streaming kontra download

Ved afspilning af musik på en smartphone er der grundlæggende tre måder at tilgå musikken på.

Den første måde er, når en bruger fysisk lægger købte/downloadede sange over på smartphonen, hvilket ofte findes besværligt. Derudover er denne løsning utrolig statisk, da det kræver tilkobling til computeren for at få nye sange overført til en smartphone. Dog findes der applikationer til Android markedet, som tilbyder download af sange, men disse linker ofte til ulovlige og ikke særligt opdaterede musikbiblioteker.

Den anden er via adskillige cloud services såsom Dropbox, SkyDrive, Google Drive, Apple iCloud samt Amazon Cloud. Fælles for dem alle er, at de tilbyder fildeling i skyen[4], der gør delingen af musik på diverse enheder betydeligt nemmere. Alle undtagen Google Drive tilbyder desuden direkte streaming fra skyen hvilket betyder, at musikken ikke behøver at blive downloadet til en smartphone før afspilning. Det er derfor blevet nemt for en bruger at tilgå det downloadede musik i skyen fra sin smartphone, hvor der kan oprettes store musikbiblioteker, uden at tænke på enhedens fysiske lagerplads. Løsningen er dog forholdsvis statisk for flere af de tilgængelige services, herunder Dropbox, SkyDrive og Google Drive, da musikken stadigvæk skal downloades og lægges i skyen andetsteds fra. Google har dog lavet en applikation, Google Play Music, der på samme måde som Apple iCloud og Amazon Cloud tilbyder, at der købes musik direkte til en placering i skyen. Dette gør musikbiblioteket mere dynamisk, da en bruger straks vil kunne downloade en given sang til sit musikbibliotek, som herefter direkte kan afspilles på en smartphone. Downloads har dog den ulempe, at det i længden vil blive dyrt for brugere, der lytter til mange forskellige musiktyper og dermed sange. Både grundet prisen for køb af de enkelte sange, men samtidigt for betaling af gigabytes hos de enkelte cloud services. Fordelen ved downloads er imidlertidig, at en bruger har ejerskab over det købte musik.

Alle de nævnte cloud services er stort set cross-platform løsninger, hvor de alle dækker en bred række af platforme med undtagelse af nogle få.

Køb og download af musik er efterhånden blevet erstattet af streaming services, der giver direkte adgang til uanede mængder af musik. Af de store kan nævnes Deezer, Rdio, MOG og Spotify, der giver adgang til flere millioner af sange. Dette betyder, at en bruger for et relativt lille beløb om måneden kan "leje" og derved tilgå alle de sange, brugeren kan forestille sig. Det er desuden utrolig nemt for en bruger at blive oprettet, hvilket har betydet en kraftig fremgang på

[4]Cloud service der indeholder data

markedet indenfor streaming de seneste år.

Fordelene ved streaming er, at musikken kan tages med overalt og afspilles på henholdsvis. en smartphone, tablet, computer etc., således at uanset hvilken enhed der anvendes, er der tilgang til de uanede mængder af musik. Fordele kommer sjældent uden ulemper og streaming services har den ulempe, at uden forbindelse til internettet, vil det ikke være muligt at afspille musikken. Dette er i skrivende stund blevet løst ved, at den musik der ønskes afspillet i offline-tilstand, kan hentes ned på telefonen og dermed ikke kræver adgang til internettet.

2.5 Evaluering af markedsanalyse

Ved valg af marked vil det optimale være en lancering på samtlige platforme, da det alt andet lige vil ramme flest mulige brugere. Dette kræver dog et helt hold af programmører, der hver især har spidskompetencer indenfor de forskellige kodesprog, som de forskellige platforme anvender. Apple anvender deres eget Objective C i iOS, hvor Google benytter Java, som grundlag for Android SDK'et. Derudover er der utrolig lave adgangsbarrierer ved udvikling til Android i forhold til iOS, da prisen for en udviklingskonto består af et engangsbeløb på $25 mod $99 pr. år til iOS. Udgivelse til iOS kræver desuden en særlig godkendelse af den udviklede applikationen, hvorimod Android som udgangspunkt tillader enhver applikation.

Ved et kig på konkurrenterne på musik anbefalings området for henholdsvis Android og iOS, ses det tydeligt, at udbudet er betydeligt lavere på Android platformen. Der må derfor antages at være mindre konkurrence på denne platform. Dette sammenholdt med vores eksisterende kompetencer indenfor Java, er det valgt at anvende Android som platform til applikationen.

Efter en undersøgelse af andre konkurrerende applikationer på det mobile applikationsmarked indenfor musik anbefalinger ses det tydeligt, at der grundlæggende er tale om applikationer målrettet til

- at modtage personaliseret musik til sportsudøvere.
- at udforske ny musik blandt mange millioner af sange.

Det gør sig gældende for samtlige applikationer, at ingen kombinerer disse to måder og det kan derfor med fordel anvendes i applikationen. Alt andet lige må

det antages, at der lyttes til forskellige typer af musik ved forskellige typer af aktiviteter, og ved bestemmelsen af en brugers kontekst, vil det være muligt at give brugeren mere præcise anbefalinger.

Grundet den stigende vækst indenfor streaming services, vil den kommende applikation tage udgangspunkt i benyttelsen af disse. Dette vil endvidere give applikationen mulighed for, at anbefale helt ny musik til en bruger som denne ikke kendte til, og samtidigt give brugeren adgang til afspilning af anbefalingerne direkte, uden at sangene først skal downloades.

Der kan med fordel gøres brug af viral markedsføring, da det herved vil være muligt, at få spredt budskabet omkring applikationen uden direkte at bruge penge på markedsføring. Det må antages at ikke alle brugere vil give en applikation tilladelse til at publicere indhold på eksempelvis Facebook eller Twitter uden videre. Der kan derfor gives en gevinst i form af særlige funktioner, hvis en bruger koblede sine sociale netværk til applikationen.

Al begyndelse er svær og før applikationen kan lanceres på Google Play og skabe indtjening, er der behov for at vælge netop den forretningsmodel, der passer bedst til applikationen. Det er derfor valgt, at udgive en gratis applikation, da dette vil betyde mere aktivitet og flere brugere der kan danne grundlag for data indsamlingen. Indsamlingen af data vil med tiden opnå en værdi omkring en brugers præferencer indenfor musik. Over tid vil det derfor være muligt, at få et afkast på applikationen uden brug af reklamer, som eksempelvis Twitter anvender.

Kapitel 3

Modeller for musikanbefalinger

For at foretage anbefalinger kan der benyttes mange forskellige typer af anbefalingssystemer og metoder. Følgende vil være en beskrivelse af generelle samt dynamiske anbefalingssystemer, hvor processen samt valg af metode bag de genererede anbefalinger vil blive beskrevet.

3.1 Anbefalingssystemer

Der findes i dag forskellige anbefalingssystemer og efter internettets stigende datamængde ses systemer, som forsøger at foretage gode anbefalinger til brugeren på mange typer af produkter, herunder musik, film, bøger men også alt fra makeup-artikler til computerdele. Produkterne har det til fælles, at de indeholder en beskrivelse. Denne beskrivelse er nødvendig for et anbefalingssystem, før det kan foretage anbefalinger af andre produkter, som ligner dette på forskellige parametre, herunder en produktkategori, kunstner, genre etc.

Helt generelt er et anbefalingssystem et system, som har til hensigt at danne præferencer omkring et produkt i systemet – der kunne eventuelt være tale om en sang. Systemet vil kunne gøre brug af både beskrivelsen af sangen samt den bedømmelse, som andre brugere har givet sangen, for herefter at bruge disse bedømmelser til at præsentere andre brugere for andre sange, der også kunne være interessante at købe.

Anbefalingssystemer behøver to delelementer for at foretage anbefalinger. Den første del er brugerens profil. Her har systemet brug for alle de informationer om brugeren som muligt. Informationer om brugeren kunne eksempelvis være

brugerens profil, herunder alder, land/landsdel, køn etc. Der kunne også være brug for brugerens historik for tidligere viste/købte produkter samt brugerens bedømmelser af de tidligere købte produkter. Den anden del er mekanismen i systemet, som gør det muligt at foretage anbefalinger, finde sammenligneligheder mellem flere forskellige produkter eller en mekanisme, der kan finde ud af, om et produkt vil være særligt interessant for en bruger eller ej.

Det beskrives i afsnit 3.2, hvordan processen for foretagelse af anbefalinger sker, men forinden, er det nødvendigt at beskrive de forskellige del-elementer, et anbefalingssystem bør indeholde. Hugo Siles har i sin afhandling[17] beskrevet, hvad et sådan system bør indeholde, hvad disse del-elementer betyder, hvilket der er taget udgangspunkt i.

Som Hugo Siles beskriver det, så er det ikke strengt nødvendigt, at dette anbefalingssystem er et stykke software, da der sagtens kan være tale om en person. Dette kan være en person fra et netværk, der anbefaler en sang, da personen tror, at sangen vil falde i god smag.

Et anbefalingssystem har behov for nogle produkter, som systemet kan sammenligne og foretage anbefalinger blandt. Produkterne i systemet indeholder værdifuld information, som senere hen vil skabe grundlag for et godt anbefalingssystem, ved sammensætning med andre del-elementer. Som tidligere beskrevet kan produkter være musik, film og bøger. Eksempelvis kan en sangs information være kunstneren, der har produceret den, albummet hvorpå sangen er udgivet samt hvilken genre den tilhører. Det kan også være mere komplekse forhold, såsom rytmen og tempoet i sangen. En films information kan være produceren, som har produceret filmen, hvilke skuespillere der er med i filmen, og som musik kan film også være af en særlig genre. Derudover ligner bøger på mange måder musik og film, idet der er tale om en forfatter, der har skrevet bogen og at den ligeledes kan være af en eller flere genrer.

De sociale medier er blevet utrolig populære de seneste år og er kommet for at blive. Disse indeholder også anbefalinger til nye venner, bekendtskaber samt information omkring brugerne. Det kan fx være civil status, placering i landet, favoritfilm etc.

Uden brugere er et anbefalingssystem ikke særlig anvendeligt, idet brugerne i systemet spiller en central rolle i foretagelsen af anbefalinger blandt systemets produkter. Brugernes profiler og deres præferencer blandt produkterne er med til at give en repræsentativ ide om, at en bruger synes godt om et produkt, er neutral omkring et produkt eller slet ikke kan lide produktet. Når en bruger bedømmer et produkt, bliver denne bedømmelse gemt i systemets database over produktet samt på brugerens profil. Vha. bedømmelserne kan systemet sammenligne produkter mellem hinanden. Derudover gør de det muligt for systemet at

sammenligne brugerne indbyrdes, hvilket gør det muligt at anbefale produkter til en bruger, der ligner de produkter som tidligere er bedømt. Herudover giver det mulighed for at anbefale nye produkter til en bruger, der ligner andre brugere på deres bedømmelser. Brugere med samme type bedømmelser kan betegnes som værende naboer, idet de ligner hinanden – jo mere de ligner hinanden på de afgivne bedømmelser desto tættere naboer er der tale om. Systemet kan herfra gruppere disse naboer og dermed skelne blandt dem ved eksempelvis; denne gruppe naboer kan lide rock, mens en anden gruppe naboer kan lide jazz. Det gør det muligt for systemet at returnere nogle produkter, som matcher deres præferencer.

Anbefalingerne som systemet viser for dets brugere kan også betegnes som systemets output. Dette er nærmere beskrevet i afsnit 3.2.

Anbefalinger kan være sammensat af enkelte anbefalede produkter som eksempelvis, at andre brugere har lyttet til denne sang og kan lide den. Der kan også være tale om en liste af anbefalede produkter, som eksempelvis en liste over de fem mest relaterede sange ud fra de afspillede. Formålet med det generelle anbefalingssystem er at præsentere brugeren for andre relevante produkter.

Anbefalingssystemer kan klassificeres under forskellige grupper, som hver især er grundlagt ud fra processen for, hvordan anbefalinger foretages. Der er tale om to generelle typer af anbefalingssystemer, som herunder er beskrevet.

- Memory-based
- Model-based

Memory-based foretager anbefalinger ud fra hele databasen, hvor model-based danner en model (uddrag fra databasen). Disse er beskrevet i de følgende afsnit.

3.1.1 Memory-based filtrering

Memory-based[6] tager udgangspunkt i alle de informationer som anbefalingssystemet indeholder i databasen, og foretager anbefalinger herfra. Metoden anvender brugernes bedømmelser af sange til at udarbejde, hvilke brugere der er ens eller tilnærmelsesvis ens. Systemet kan dermed finde sammenlignelige sange, da det tager udgangspunkt i, hvordan brugerne har bedømt de forskellige sange.

Ideen bag memory-based anbefalinger er at anbefale nye sange baseret på bedømmelser fra andre ligesindede brugere, som tidligere har bedømt andre sange

ens. Baseret på systemets brugere vil der blive dannet såkaldte naboer, der grundlæggende set er brugere, som ligner hinanden baseret på deres præferencer blandt de bedømte sange. En brugers tætteste naboer vil herefter blive anvendt som en samlet gruppe, der skal være med til at danne grundlag for de anbefalinger, en given bruger modtager fra systemet.

Ved anvendelse af memory-based giver det systemet en række fordele. Nogle af disse fordele kunne eksempelvis være.

- Det er nemt at implementere, idet der vil blive taget udgangspunkt i bedømmelserne fra brugerne på de afspillede sange.
- Det er nemt at foretage opdateringer i databasen, idet hele databasen vil blive anvendt for hver gang, der skal foretages en ny anbefaling til en bruger.
- Kvaliteten af anbefalingerne til brugerne vil være gode, idet brugerne er med til at gøre anbefalinger bedre. Jo flere og mere præcise bedømmelser brugerne giver, desto bedre bliver de anbefalinger de modtager.

Memory-based besidder dog også ulemper.

- Anbefalinger foretages på baggrund af hele databasen og ventetiden vil derfor stige proportionalt med størrelsen af databasen.
- Der kan ikke dannes anbefalinger til en bruger, der ikke tidligere har bedømt andre sange, da systemet ikke har mulighed for at kende til brugerens præferencer.
- Da der tages udgangspunkt i brugerens bedømmelser, så vil anbefalingerne kun være ligeså gode, som dem brugeren give retur til systemet. Dvs. hvis bedømmelserne varierer, så vil kvaliteten af de anbefalinger brugeren får retur også varierer i kvalitet.

3.1.2 Model-based filtrering

Til forskel for memory-based anbefalinger, udarbejder model-based[7] anbefalinger ligheder mellem brugere og sange, der bliver gemt i en slags model, som samtlige nye anbefalinger vil blive beregnet fra. Dette gør det muligt ikke at skulle gennemgå hele databasen for bedømmelser, hver gang der skal udarbejdes en ny anbefalin, hvilket gør metoden hurtigere end memory-based.

Anvendelsen af model-based anbefalinger giver systemet følgende fordele.

- Yderst skalérbar da den model, der bliver dannet fra databasen alt andet lige vil være mindre end databasen. Der kunne fx. udtages 10.000 bedømmelser ud af en database indeholdende 100.000, og systemet ville stadig kunne foretage anbefalinger til brugerne.
- Da modellen allerede indeholder alle ligheder mellem brugere/sange, så vil anbefalinger kunne foretages meget hurtigt, da systemet ikke skal køre samtlige bedømmelser fra databasen igennem for at få anbefalet en ny sang til en bruger.

Model-based har, som memory-based, nogle ulemper.

- Da alle anbefalinger vil blive foretaget ud fra en model, bliver den hurtigt meget statisk. Statisk fordi tilføjelsen af nyt data, kræver udarbejdelse af en ny model. Sker dette ikke, vil modellen være upræcis.
- Når anbefalingerne bliver foretaget ud fra en model, bliver kvaliteten af den information systemet har til at foretage anbefalinger også det ringere, hvilket kan resultere i mere upræcise anbefalinger til brugeren.

3.2 Anbefalingsprocessen

De grundlæggende elementer, der er nødvendige i et anbefalingssystem, kan sættes sammen for at beskrive processen for, hvordan de bliver anvendt i systemet. Selvom der findes mange forskellige typer af anbefalingssystemer, så er metodikken samt processen for dem alle den samme. Følgende vil beskrive den mest simple form for proces som et anbefalingssystem kan anvende, samt beskrive den bedst mulige proces et system bør sættes op efter, som på sigt gør det muligt for systemet at blive klogere på dets brugere, baseret på deres bedømmelser. På den måde kan brugeren modtage bedre og mere personaliserede anbefalinger.

3.2.1 Generelt anbefalingssystem

Den mest simple proces, der ifølge Hugo Siles, beskrives som en generel proces for anbefalingssystemer - uanset hvilken type anbefalinger det drejer sig om, indeholder tre dele og er illustreret på figur 3.1.

Figur 3.1: Illustration af et generelt anbefalingssystem.

De tre dele er:

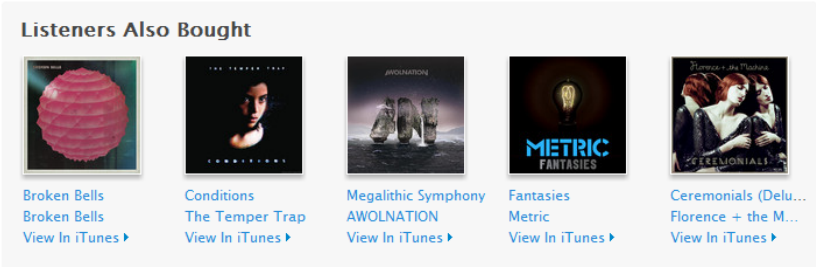
- Input, som indeholder brugerprofiler og systemets produkter, information omkring produkterne, samt hvorledes produkterne er bedømt blandt brugerne.
- Anbefalingssystemet, der sørger for at indhente information fra brugerprofilerne samt produkterne.
- Output, som er de anbefalinger, brugeren modtager.

Som det fremgår af figur 3.1 har anbefalingssystemer behov for information (input) for at kunne foretage anbefalinger til brugeren (output). Systemet indhenter information fra to kilder – sine brugeren og produkterne.

Når der skal indhentes information fra brugeren, kan det ske eksplicit eller implicit. Eksplicitte informationer er, når brugeren manuelt har indtastet generelle informationer om dem selv og eventuelt om deres musikvaner i forskellige situationer. Dvs. hver bruger har aktivt givet information til systemet. Implicitte informationer kan systemet eksempelvis få fra brugerens Facebook-konto, hvis brugeren aktivt har givet godkendelse til, at systemet må indhente data fra brugerens eksterne kilder. Det kan også være i form af brugerens afspilningsmønster, hvor systemet lagrer, hvilke sange brugeren enten har købt, kigget på eller afspillet. Systemet har dermed uden brugerens interaktion indhentet information om brugerens musikvaner.

Information fra produkterne kan være metadata til en sang såsom titel, kunstner, året for udgivelse, genren, længden på sangen samt hvilket eller hvilke album sangen optræder på. Der kan også kigges nærmere på sangens mønster og herved komme ind på sangens BPM[1], rytmen og endda humøret på sangen – er der tale

[1] Beats per minute



Figur 3.2: Illustration af anbefalede produkter

om en glad, trist eller aggressiv sang. Systemet kan også hente information om en sang fra eksterne kilder. Eksempelvis biografien fra kunstneren, anmeldelser fra blogs eller andre kilder på internettet, samt hvor hot eller populær en kunstner er blandt andre kunstnere etc.

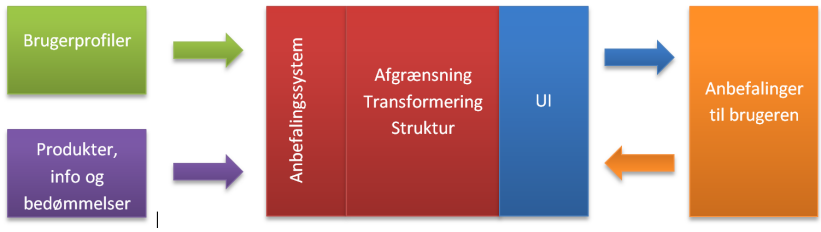
Systemet kan herefter bearbejde de forskellige informationer, som bliver hentet fra brugeren og produkterne. Ved anvendelse af forskellige anbefalingsmekanismer kan systemet foretage en række forskellige typer af anbefalinger og præsentere disse for brugeren. Alt efter hvilken type af anbefalingssystem der er tale om, kan der være forskel på, hvorledes de anbefalede produkter, bliver præsenteret overfor brugeren. For som Hugo Siles også beskriver det, kan det variere fra en liste, en liste med billeder, som vist på figur 3.2 til hele udsnit af produkterne.

3.2.2 Dynamisk anbefalingssystem

Den generelle proces for anbefalingssystemer er ikke tilstrækkelig til det mere dynamiske anbefalingssystem, som med tiden skal blive bedre til at kende sine brugere samt til at foretage gode anbefalinger. Systemet beskrevet tidligere mangler del-elementer, som giver systemet mulighed for at blive mere dynamisk. Baseret på Hugo Siles erfaringer kan del-elementerne, som det generelle anbefalingssystem mangler, opstilles som vist på figur 3.3. Figuren viser procesen for et mere dynamisk anbefalingssystem.

3.2.2.1 Afgrænsning af data

Systemet indeholder allerede en større mængde data, både omkring brugeren og eksempelvis dennes musikvaner. Hver gang brugeren lytter til en sang, bliver



Figur 3.3: Illustration af et dynamisk anbefalingssystem, der genererer bedre anbefalinger ud fra bruger feedback.

kunstneren og titlen på sangen registreret i systemet. Systemet har brug for de del-elementer, som kan undersøge hvilke produkter, der vil være interessante eller relevante for en given bruger og hermed fjerne alle andre produkter, som er irrelevante. På denne måde vil brugeren altid få anbefalet netop de produkter, som er interessante. Systemet skal derfor have en metode, som ikke blot skal vælge data og afgrænse dem, men også en metode der, vha. forskellige paramtre, kan definere hvor sammenlignelige produkter og brugere er. Afgrænsningen af data afhænger udelukkende af anbefalingsmetoderne, som anbefalingssystemet anvender. De forskellige typer af metoder er beskrevet senere i afsnit 3.3.

3.2.2.2 Transformering

Anbefalingssystemet kan i nogen grad hente sine anbefalinger fra eksterne kilder eksempelvis hente anbefalinger til sange fra blandt andet Echo Nest. Systemet får en lang række sange retur fra Echo Nest, men det er ikke sikkert, at alle sange er relevante for brugeren. Anbefalingssystemet skal derfor transformere de modtagne sange til sange, som brugeren vil finde interessante. Transformationen kunne eksempelvis være at filtrere karaoke udgaver af sange eller sange med anstødeligt indhold fra. Ideen bag transformationen er at oprette et filter og filtrere uønsket indhold væk, så kun relevant indhold for brugeren bliver præsenteret.

3.2.2.3 Strukturering og præsentation af anbefalinger

Brugeren skal navigere gennem de forskellige anbefalede sange, og for at opretholde interessen er strukturen, for hvordan de anbefalede sange bliver vist, vigtig. Der findes forskellige måder at strukturere de forskellige sange på. Der kan fx laves en liste over samtlige sange og arrangere denne efter de bedst anbefalede ud fra forskellige parametre – eksempelvis kunne den mest interessante og

relevante sang ligge øverst i listen. Sangene kunne også arrangeres i grupper ud fra, hvor tætte naboer eller ens de forskellige sange er, igen baseret på parametre som fx sangenes metadata. Det er vigtigt at have fokus på, hvordan det bliver præsenteret for brugeren. Hvis det anvendte UI/UX[2] omkring de anbefalede sange ikke falder i brugerens smag eller er langsommeligt og svært at gennemskue, vil det resultere i utilfredse brugere og i værste fald brugere, som vil se sig om efter andre steder at få gode anbefalinger. Derfor skal der tages højde for, hvordan brugeren opfatter og oplever de modtagne anbefalinger. Dermed skal brugeren være i fokus, når et godt UI skal udvikles.

3.2.2.4 Feedback

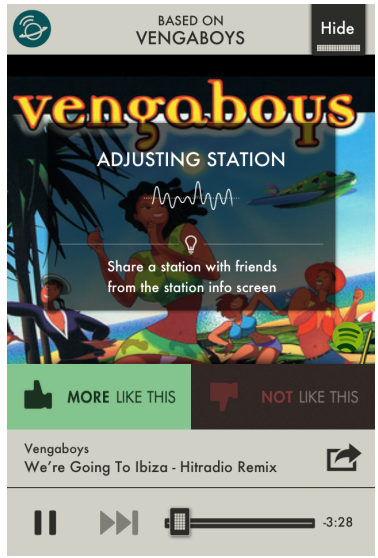
Et anbefalingssystem er udviklet for at kunne give systemets brugere anbefalinger på nye sange baseret på forskellige parametre. Der findes gode og dårlige anbefalinger, men indeholder systemet ikke en funktion, som kan bestemme om de anbefalinger brugeren modtager er gode eller dårlige, vil anbefalingssystemet fejle på sigt. Det er derfor nødvendigt, at et anbefalingssystem hele tiden opdaterer brugerens musikprofil. For bedre at kunne forstå brugerens musik adfærd er det nødvendigt, at implementeret en funktion, der gør det muligt for brugeren at give feedback på de anbefalede sange, de modtager.

Der findes, som tidligere beskrevet, to forskellige metoder at indhente feedback fra brugerne på, eksplicit eller implicit feedback.

Eksplicit feedback
Den første mulighed er at spørge brugeren direkte, for at få relevant feedback på de sange, som anbefalingssystemet har anbefalet. Eksplicit feedback fra brugeren bliver som udgangspunkt altid målt i en bedømmelse fra brugeren. Et eksempel på dette kunne være, at brugeren i applikationen SpotON, kan give en anbefalet sang tommelfingeren op for at give sangen positiv feedback, eller tommelfingeren ned for negativ feedback, som illustreret på figur 3.4.

En anden type bedømmelse kunne være, at brugeren havde mulighed for, at bedømme en anbefalet sang i en skala fra fx 0 til 5, hvor 0 er den laveste værdi og 5 den højeste. Dette giver brugeren mulighed for selv at placere en anbefalet sang på den givne bedømmelsesskala, som både kan bidrage positivt eller negativt til anbefalingssystemet. Ved at give brugeren muligheden for at vælge en specifik bedømmelse til en anbefalet sang, kan der opstå uoverensstemmelser om, hvordan en bruger opfatter en sang og bedømmer. Systemet har derfor behov for at have forskellige metoder, som udelukkende skal sikre, at variabiliteten og afvigelserne blandt brugernes bedømmelser bliver holdt på et minimalt niveau.

[2] User Interface/User Experience



Figur 3.4: Eksempel på eksplicit feedback, hvor der kan vælges *More like this* eller *Not like this*.

En tredje mulighed er, at lade brugeren kommentere på de anbefalede sange og give sangene forskellige tags, for dermed at komme med mere præcis feedback. Denne ekstra information vil imidlertid kræve, at systemet skal kunne aflæse rå-teksten, for at se om der er tale om negativ eller positiv feedback.

Implicit feedback
Anbefalingssystemet kan også indhente feedback implicit fra brugeren ved passivt at følge brugerens aktivitet. Eksempelvis ved at registrere og analysere de senest afspillede sange, hvor mange gange en bruger har afspillet en specifik sang, for at kunne registrere om sangen falder i brugerens smag. Hvis en sang er blevet spillet mere end én gang vil systemet antage, at sangen falder i brugerens smag og bliver derfor registreret som værende en positiv anbefaling.

Er der tale om sange, som ikke falder i brugerens smag, er den implicitte feedback fra brugeren ikke nær så pålidelig og præcis som den eksplicitte, der modtager en klar melding fra brugeren. Systemet skal derfor observere brugerens handlinger, når der bliver afspillet en sang, og hvis brugeren skifter sang ualmindeligt hurtig efter en ny sang er startet, må det antages, at sangen ikke er faldet i brugerens smag og vil blive registreret negativt.

	Track 1	*Track 2*	*Track 3*	*Track 4*	*Track 5*	*Track 6*
UID_1	1	0		1	1	0
UID_2	0		1	1	1	0
UID_..	..	..	..	..	..	..
UID_N	0	1	0	0	0	1

Tabel 3.1: Illustration på et system, der anvender sociologisk filtrering.

3.3 Anbefalingsmetoder

Der findes mange forskellige metoder at lave anbefalinger på. Nedenstående vil være en gennemgang af de mest anvendte anbefalingsmetoder, deres fordele samt ulemper, iht. Andrius Butkus' afhandling [5].

3.3.1 Sociologisk filtrering (Collaborative filtering)

En af det mest anvendte metoder idag er Collaborative Filtering (herefter omtalt som CF), som efterhånden også er almindelig kendt som den sociologiske anbefalingsmetode, der tager udgangspunkt i brugerens bedømmelser af sange. Tabel 3.1, illustrerer systemets brugere samt de sange brugerne har afspillet og afgivet deres bedømmelse på, og som danner grundlag for samtlige anbefalinger.

Den hyppigst anvendte CF er idag den brugerbaserede CF, hvor der bliver opbygget et netværk på brugere, som ligner en given bruger mest muligt, baseret på tidligere afgivne bedømmelser. En anden metode er den sangbaserede CF, hvor der bliver skabt et netværk over sange, som ligner hinanden mest muligt ud fra de bedømmelser, brugeren har givet dem. Som det fremgår, er den brugerbaserede og den sangbaserede CF meget ens og processen foregår også på samme måde, hvor der først bliver opbygget et netværk over enten brugere eller sange. Netværket danner grundlag for anbefalinger til nye brugere, som en given bruger ligner. Netværket kan ligeledes danne grundlag for nye sange.

3.3.1.1 Brugerbaseret CF

Som beskrevet, er den brugerbaserede CF den mest anvendte. Derfor bliver den også gerne omtalt som den traditionelle CF. Grundideen bag brugerbaseret CF

er, at brugeren vil kunne lide sange, som andre ligesindede brugere tidligere har bedømt godt. Ligesindede brugere er brugere, som har afspillet de samme sange og har bedømt dem nogenlunde ens. Dette er illustreret i Tabel 3.1, hvor brugere (UID) har afgivet deres bedømmelser til sangene (Track). En positiv bedømmelser af angivet som 1, hvor en negativ er angivet som 0. En sang, som en bruger ikke har afspillet, er angivet uden indikator.

Jo større systemet er desto flere ligesindede vil der være. Netværket af ligesindede brugere, som bliver dannet, kan afgrænses på to forskellige måder. Enten kan man afgrænse netværket ved at definere en maksimal grænse for, hvor mange ligesindede brugere, et netværk må indeholde - eksempelvis seks brugere. Problemet ved at sætte en øvre grænse for, hvor mange ligesindede brugere der danner netværket er, at grænsen normalt bliver sat for lavt og dermed har for lidt informationer til, at basere nye anbefalinger på. Alternativt kan top-N af ligesindede brugere vælges, således at brugere bliver sat i rækkefølge efter hvor ligesindede de er. På denne måde vil der som udgangspunkt være tilstrækkelig med ligesindede brugere til at foretage anbefalinger ud fra. Problemet med denne måde er, at top-N brugere ikke altid er ligesindede brugere, eller ligesindede nok til at foretage gode anbefalinger ud fra.

Ethvert anbefalingssystem afhænger af, om de anbefalinger der bliver foretaget er gode og dermed kan give brugerne gode anbefalinger. Disse anbefalinger bliver beregnet ved fx. anvendelse af CF og det har derfor nogle fordele og ulemper. I den brugerbaserede CF er der behov for at systemet hele tiden er online, idet systemet skal gennemgå og finde brugere som ligner hinanden. Dette kræver ressourcer og bliver det meget populært, vil antallet af brugere stige og ressourcerne for at finde ligesindede brugere stige tilsvarende.

3.3.1.2 Sangbaseret CF

Sangbaseret CF svarer, som tidligere beskrevet, meget til den brugerbaserede CF, men grundideen bag sangbaseret CF tager udgangspunkt i, at brugeren vil være interesseret i sange, som ligner sange brugeren tidligere har lyttet til. Selve processen foregår principielt som i brugerbaserede CF, men der bliver skabt et netværk over lignende sange istedet for et netværk over ligesindede brugere. Sangbaseret CF vil gennemgå de sange som er indeholdt i anbefalingssystemet, for så at søge efter sange der ligner hinanden. Den store fordel ved at anvende sangbaseret CF er, at beregningerne af hvilke sange der passer sammen med hvilke sange kan ske offline, hvor brugerbaseret CF sker online. Den er derfor mere skalérbarhed og dermed vil kunne give hurtigere anbefalinger, da de sammenlignelige sange er beregnet på forhånd, hvorimod brugerbaseret CF skal beregnes i det øjeblik brugeren har behov for det.

3.3.1.3 Ulemper ved CF

De teknologiske fremskridt, herunder internettet har givet os uanede mængder af data til rådgihed og efterspørgslen på relevant data stiger i takt med den stigende datamængde. Der er derfor behov for metoder, der kan udvælge relevant information for brugeren - herunder brugen af CF. Nogle af ulemperne ved brug af CF er imidlertid.

- Kold start - manglende informationer
- Sparsomme bedømmelser fra brugerne
- Skævheder i brugernes bedømmelser
- Ukorrekte bedømmelser fra brugerne
- Popularitetsfaktorer

Anbefalingssystemer arbejder på forskellige komplicerede platforme, herunder Amazon.com, Apple iTunes, Spotify og mange andre. Anbefalingssystemet er til for at hjælpe brugeren med at finde relevante produkter herunder kunstnere, albums og sange, men systemerne skal også sørge for at fastholde interessen hos brugeren for at sikre salg i samtlige sange, som de forskellige platforme tilbyder. For at kunne tilbyde både gode anbefalinger og forudsige hvilke sange, som brugerne vil lytte til, er det nødvendigt, at anbefalingssystemet kan håndtere de forskellige udfordringer, de står overfor.

Kold start – manglende informationer
Tages der udgangspunkt i et anbefalingssystem, der bliver anvendt til at bearbejde et meget stort antal sange fx. Spotify med over 15 millioner må det antages, at der også er flere millioner aktive brugere. Problemet er imidlertid, hvis brugeren ikke bedømmer de sange der afspilles, så vil det være svært for systemet at koble sangene sammen med brugeren. Problemet opstår, når en ny bruger bliver oprettet i systemet, eller når en ny sang bliver tilføjet, da det vil være svært for systemet at foretage gode anbefalinger til en ny bruger, når systemet ikke har kendskab til brugerens musikpræferencer. Nye sange vil ligeledes være vanskelig for systemet at anbefale idet sangen ikke tidligere har været afspillet eller bedømt af brugerne.

Sparsomme bedømmelser fra brugerne
Hvis kun få brugere bedømmer sangene, vil det medføre at systemet vil have svært ved at give gode anbefalinger til brugeren. Dette gælder både nye og eksisterende sange.

Skævheder i brugernes bedømmelser
Brugerne afgiver deres bedømmelser ud fra sange, de har afspillet. Det ses, at nogle brugere afgiver ukorrekte bedømmelser på sange, som de har afspillet. Eksempelvis ved at bedømme en sang som værende god, men hvor brugeren istedet "burde"have bedømt den anderledes. Forkerte bedømmelser kan resultere i forkerte anbefalinger. Der kan være tale om enten en menneskelig fejl eller en overlagt handling, hvor brugeren bevidst bedømmer en sang højere end den burde. Baggrunden for en ukorrekt bedømmelse kan være, at der ønskes en tilsigtet promovering af en sang idet antallet af gode bedømmelser er proportional med sandsynligheden for at sangen vil blive anbefalet.

Skævheder i brugernes bedømmelser
Mennesker er alle forskellige af natur og har derfor ikke samme musiksmag. Musikvaner er derfor også vidt forskellige. Det må derfor antages, at brugerne vil bedømme en given sang forskelligt alt efter hvem, der har afspillet den og en sang vil derfor naturligt have både gode og dårlige bedømmelser. Skævheder kan opstå, hvis brugeren foretager ukorrekte bedømmelser, hvorfor det er nødvendigt at systemet skal være i stand til at håndtere dette, ved at normalisere eller endda helt at kunne skelne blandt bedømmelserne.

3.3.2 Indholdsbaseret filtrering (Content-based filtering)

Ved anvendelsen af CF, vil værdien af anbefalingerne afhænge af, hvordan brugerne har bedømt tidligere sange, men der bliver ikke taget højde for det indhold en sang kan have - hvilket indholdsbaseret filtrering kan afhjælpe. Udgangspunktet for at foretage anbefalinger er den samme som CF, hvor “en bruger vil være interesseret i sange som ligner sange, brugeren tidligere har bedømt højt”, men forskellen er, at der ved CBF tages udgangspunkt i det indhold, sangene har til fælles.

Ligesom med almindelig CF kan anbefalingerne foretages enten memory-based eller model-based, som er uddybet i afsnit 3.1. Dvs. at anbefalinger enten kan foretages ved at gennemgå hele databasen for sangenes indhold for hver gang der skal foretages en ny anbefalinger (memory-based), eller ved at udarbejde en model med ligheder blandt et uddrag af sangene fra hele databasen.

Før CBF kan foretage anbefalinger er det nødvendigt at kende sangenes indhold. Jo mere detaljeret indholdet er, desto nemmere vil det være for systemet at finde sange med fælles indhold. Det vil også ligeledes give bedre anbefalinger til brugeren, idet kvaliteten af anbefalinger vil være skabt ud fra flere forskellige typer indhold. Systemet skal derfor kunne indhente indhold fra sange, og ud fra dette foretage anbefalinger på nye sange til brugeren. Eksempelvis kan der

hentes indhold fra brugerens tidligere afspillede sange. Ligesom med sangbaseret CF har CBF også en matrice over de forskellige sange, hvor bedømmelserne blot er skiftet ud med sangenes forskellige typer af indhold. Matricen vil således indeholde lignende sange, hvor hovedfokus vil være flyttet fra brugernes bedømmelser til selve indholdet. Systemet vil være i stand til at tegne et billede af brugerens musikpræferencer udelukkende på baggrund af de sange, som brugeren har bedømt.

Sangenes indhold kan eksempelvis være repræsenteret helt generelt som kunstnerens eller gruppens navn, sangtitel, genre, året den blev lanceret, hvilket album den tilhører etc.. Alle disse informationer vil kunne hentes direkte fra en sangs metadata, også kaldet ID3-tag. ID3-tag indeholder metadata omkring en sang, som allerede er gemt i selve musikfilen. Systemet skal dermed blot "læse" sangens ID3-tag igennem og på den måde indhente indholdet af hver sang.

3.3.2.1 Ulemper ved CBF

Det er med stor interesse for både brugere, men også anbefalingssystemet at brugerne får gode anbefalinger. Selvom CBF løser nogle af de problemstillinger som CF ikke kan håndtere, står systemet der anvender CBF overfor nogle andre udfordringer:

- Kold start - tilføjelse af ny bruger
- Begrænset indhold og informationer i sangene
- Specialiserede anbefalinger ud fra begrænset indhold
- Synonymer

Kold start - tilføjelse af ny bruger
Ligesom ved CF er der udfordringer, når en ny bruger bliver tilføjet til anbefalingssystemet, idet CBF afhænger af brugerens præferencer. Før brugeren kan modtage anbefalinger er det derfor nødvendigt, at brugeren aktivt lytter til forskellige sange og afgiver sin bedømmelse til hver af disse. Bedømmelserne kan gøres enten implicit eller eksplicit, og er nærmere beskreveti afsnit 3.2.2.4. Først når systemet har fået opbygget en brugerprofil med en brugers præferencer, kan brugeren modtage anbefalinger.

Begrænset indhold og informationer i sangene
Anbefalinger ud fra anvendelsen af CBF er direkte relateret til indholdet af de sange, som brugeren afspiller. CBF er derfor begrænset af sangenes indhold.

Det er derfor nødvendigt, at der er et vis niveau af informationer eller indhold i sangene, før systemet kan foretage anbefalinger. Er indholdet begrænset, er informationerne til systemet ikke tilstrækkelige og det vil derfor ikke være muligt for systemet af kunne genkende sangen og/eller kunne finde lignende sange, som kan anbefales til brugeren.

Specialiserede anbefalinger ud fra begrænset indhold
Et resultat fra sange med begrænset indhold kan medføre, at brugeren modtager både specialiserede samt et ringe udvalg af anbefalinger. Hvis en bruger eksempelvis kun har hørt musik af Michael Jackson vil systemet være begrænset til at anbefale sange af Michael Jackson, men som brugeren ikke har afspillet. Dvs. brugeren modtager anbefalinger til nye sange af Michael Jackson. Problemet kan afhjælpes ved at hæve kvaliteten og mængden af sangenes indhold.

Forskellige sange men med samme indhold
Idet der alene bliver lagt vægt på indholdet af sangene, mister systemet muligheden for at kunne skelne imellem dem og dermed fortælle, om der er en eller flere forskellige sange. To forskellige sange kan meget vel indeholde de samme informationer og dermed have samme indhold, og da CBF anbefaler ud fra indholdet, vil brugeren modtage forkerte anbefalinger.

3.3.3 Kontekst-baseret filtrering

Hvor sociologisk filtrering er den mest almindelige og dermed også den mest anvendte i forskellige anbefalingssystemer, er kontekst baseret filtrering som udgangspunkt ikke en generelt anvendt anbefalingsmetode. Dette gør den til en vigtigt faktor i applikationen, som skal differentiere sig fra de andre konkurrenter på det mobile marked.

Denne anbefalingsmetode er på mange måder forskellig fra blandt andre CF, CBF etc., da anbefalinger hverken tager udgangspunkt i brugerens mening og bedømmelse af forskellige sange, ligheder blandt ligesindede brugere eller i indholdet af de forskellige sange, de har til fælles. Kontekst-baseret filtrering tager udgangspunkt i brugerens kontekst, som er sammenhængen mellem brugeren og den kontekst, brugeren befinder sig i. Det kan eksemplificeres ved, "Brugeren har lyttet til denne sang i denne kontekst". Konteksten vil da blive gemt i databasen for senere hen, at bruges til at gøre anbefalingerne til brugeren mere præcise.

Anbefalingerne for kontekst-baseret filtrering foregår på den måde, at anbefalinger sker ud fra sange, som brugeren tidligere har lyttet til i en given kontekst. Det er derfor nødvendigt for systemet at modtage den rette kontekst, hvori en

sang er blevet afspillet. Der findes forskellige typer af bruger-kontekst og nogle af dem er herunder beskrevet.

- En sang er blevet lyttet til ved denne type af aktivitet
- En sang er blevet lyttet til på denne lokation
- En sang er blevet lyttet til sammen med denne eller disse personer
- Der er blevet lyttet til en sang under særlige omgivelser såsom lys -og støjforhold.

Aktivitet
En bruger kan eksempelvis, over en periode, have lyttet til en særlig type af musik under løbeture. Brugeren vil derfor blive anbefalet nye sange ud fra sange, der tidligere er blevet afspillet i den givne kontekst.

Lokation
Ligesom musik bliver lyttet til under mange former for aktivitet, så kan lokationen også have indflydelse på valg af musik. Eksempelvis vil valg af musik være forskellige fra om brugeren er hjemme eller på arbejde, hvor der er andre at tage hensyn til. Ligesom ved en given aktivitet, bliver der foretaget anbefalinger til brugeren baseret på det brugeren tidligere har lyttet til på en given lokation.

Venner og bekendte (socialt)
Ligesom forskellige aktiviteter og lokationer præger en brugers valg af musik, så vil valget også være præget af, hvem brugeren er sammen med. Eksempelvis kan brugeren have en særlig type musik tilfælles med en gruppe af venner, som de typisk lytter til, når de er sammen. Næste gang systemet ser, at brugeren er sammen med denne gruppe, vil brugeren blive anbefalet musik ud fra tidligere afspillet musik.

Særlige omgivelse
Der afspilles musik på forskellige lokationer og som regel vil der ikke blive afspillet samme type musik på samtlige lokationer. En lokation kan danne omgivelser for en særlig type aktivitet. Eksempelvis kan hjemmet danne omgivelser for både arbejde og fritid, tillige med social aktivitet blandt venner og familie, hvor der givetvis bliver lyttet til forskellige typer af musik på samme lokation under særlige omgivelser. Der kan derfor være behov for at tage højde for fx. lys og støjforhold, når en sang skal registreres under en given aktivitet. Der kunne eksempelvis være tale om en stille middag med dæmpet lys med kæresten eller en larmende fest med nogle tidligere kollegaer. Under disse to vidt forskellige situationer, vil der sandsynligvis blive afspillet vidt forskellige typer af musik.

Fælles for de forskellige typer kontekst er, at anbefalingssystemet kan basere sit gæt på anbefalede nye sange til brugeren ud fra den givne type af kontekst brugeren befinder sig i, uanset om der er tale om at brugeren er ude at løbe, går en tur med hunden, er på arbejde, eller nyder weekenden blandt venner og bekendte i forskellige omgivelser.

3.3.3.1 Ulemper ved kontekst-baseret filtrering

Selvom kontekst-baseret filtrering er med til at beskrive, hvilken kontekst brugeren befinder sig i og dermed gør personalisering af de anbefalede sange mere præcis, er der også ulemper forbundet med den.

- Kold start, uden bedømmelser fra brugeren
- Forkert bestemmelse af kontekst

Kold start - manglende informationer
Problemet er den samme som det tidligere er set ved anvendelsen af CF. Ved anvendelsen af kontekst-baseret filtrering er det derimod ikke bedømmelser fra brugerne der mangler, men blot hvilken kontekst en given sang er blevet afspillet i. Første gang brugeren anvender applikationen under en løbetur, vil brugeren ikke kunne modtage anbefalinger på nye sange, da systemet ikke tidligere har registreret sange ved den givne kontekst. Det vil derfor også være svært for systemet at foretage anbefalinger til en ny bruger, da systemet ikke har kendskab til brugerens musikpræferencer, ud fra de forskellige typer af kontekst brugeren kan befinde sig i. Jo flere gange en bruger anvender applikationen i forskellige typer af kontekst, desto bedre bliver kvaliteten af de anbefalinger brugeren modtager.

Forkert bestemmelse af kontekst
Anbefalingerne, som brugeren modtager fra systemet, vil være bestemt ud fra deres nuværende kontekst hvorefter, systemet vil se på, hvad brugeren tidligere har hørt i denne kontekst og hermed præsentere anbefalede sange. Det hele afhænger af at kunne bestemme brugerens givne kontekst. Hvis systemet ikke kan bestemme brugerens kontekst eller bestemmer den forkert, vil brugeren modtage forkerte anbefalinger og fremtidige anbefalinger vil også blive upræcise, da de sange brugeren lytter til, vil være registreret med en forkert type kontekst. Det er derfor yderst vigtigt, at systemet kan tage højde for forskellige afvigelser, når der skal bestemmes en brugers kontekst.

3.3.4 Hybrid-baseret filtrering

Det er tidligere blevet vist, at ved anvendelsen af enten CF, CBF eller kontekstbaseret filtrering er der forskellige typer af ulemper, som hverken gavner brugeren eller anbefalingssystemet. Hybrid-baseret filtrering har til formål at kombinere de forskellige typer af anbefalingsmetoder. Hver for sig har anbefalingsmetoderne både fordele og ulemper, men ved anvendelse af hybrid-baseret filtrering, vil det være muligt at sammensætte eksempelvis to anbefalingsmetoder, som sammen minimerer de ulemper de nu måtte have hver for sig.

Der findes mange forskellige kombinationsmuligheder og herunder er nogle af dem listet op. Der er blevet taget udgangspunkt i anvendelsen af de tre anbefalingsmetoder, CF - som er den sociologiske anbefalingsmetoder, CBF - som er den indholdsbaserede anbefalingsmetode og kontekst-baseret filtrering, som foretager anbefalinger ud fra en kontekst en given bruger kan have.

Nogle af de forskellige kombinationer er

- Sociologisk -og indholds-baseret filtrering (CF og CBF)
- Sociologisk (CF) -og kontekst-baseret filtrering
- Indholdbaseret (CBF) -og kontekst-baseret filtrering

Sociologisk -og indholdsbaseret filtrering (CF og CBF)
Den sociologiske anbefalingsmetode er, som tidligere nævnt, den mest anvendte og dermed også den anbefalingsmetode, som mange personer refererer til, så snart der bliver talt om at foretage anbefalinger. Disse anbefalinger sker ud fra bedømmelserne på de sange, som brugeren af et anbefalingssystem har lyttet og afgivet deres bedømmelse til. Problemet ved at foretage anbefalinger ud fra brugernes bedømmelser er mange og er yderligere uddybet i afsnit 3.1.

For at komme disse forskellige ulemper til livs, skal der indhentes informationer andre steder fra. Bedømmelserne bliver byttet ud med features omkring sangene, som yderligere er uddybet i afsnit 3.3.2, hvor anbefalinger foretages ud fra sange, som ligner andre sange på deres features i stedet for at se på, hvordan brugerne har anbefalet sangene. Anbefalinger foretaget ud fra forskellige features i sange er ikke fejlfri, hvilke er beskrevet i afsnit 3.3.2.

Eksempelvis mangler der en sammenhæng i de anbefalinger, der bliver foretaget ud fra brugerens bedømmelse, idet der ikke tages højde for afvigelser i disse. Denne ulempe blive minimeret ved anvendelsen af indholdsbaseret filtrering, da

anbefalinger i stedet bliver foretaget ud fra sangenes feature, som de har til fælles.

Sociologisk (CF) -og kontekst-baseret filtrering
Alene står CF overfor en række problematikker, herunder menneskelige faktorer hvor brugeren afgiver sin personlige bedømmelse til de afspillede sange. Disse er yderligere beskrevet i afsnit 3.3.1

Den kontekst-baserede filtrering kan minimere problematikkerne, hvis ikke helt løse dem. Den føromtalte kombination af CF og CBF sammensætter brugerens personlige bedømmelser med de forskellige features hver sang har. Disse features fra sangene bliver erstattet med en brugers givne kontekst, som er nærmere beskrevet i afsnit 3.3.3.

Det vil derfor være muligt, at koble brugerens bedømmelse sammen med konteksten, som brugeren har haft på det givne tidspunkt. Denne sammensætning giver mulighed for at bestemme hvilke sange der passer i given kontekst.

Indholds-baseret (CBF) -og kontekst-baseret filtrering
Selvom det er vigtig at tage højde for det personlige aspekt mht. brugerens bedømmelser, kan disse afvige og afspejler ikke altid det sande billede for en given sang. Ved at tage udgangspunkt i indholdet af sangene og deres ligheder, er det muligt at danne mere præcise anbefalingerne til brugeren. De forskellige features samt CBF's fordele og ulemper er beskrevet i afsnit 3.3.2.

Ved alene at anvende indholdet af sangene, vil anbefalinger forblive snævre, idet anbefalingerne kun vil blive dannet ud fra eksempelvis kunstnernavn, album titel etc. Den manglende information kan hentes fra brugerens kontekst, og ved at sammensætte de forskellige features fra sangen med brugerens kontekst, vil det være muligt for systemet, at eliminere ulempen ved ensartede anbefalinger - uanset kontekst.

3.3.5 Demografisk filtrering

Demografisk filtrering kan anvendes til at identificere, hvilken type af brugere, der kan lide en bestemt type musik eller mere bestemt, hvilke sange de kan lide. Denne filtreringsmetode behøver personlige data fra systemets brugere og anvender denne information til at klassificere systemets brugere. Der kunne fx være behov for at kende personlige informationer såsom køn, alder, civil status, eller mere geografiske informationer såsom land, landsdel og postnummer, som vist i tabel 3.2

Bruger ID	*Køn*	*Alder*	*Civil status*	*Land*	*Landsdel*	*Postnr.*
UID_1	Kvinde	19	Single	Danmark	Nordjylland	9000
UID_2	Mand	24	Forhold	Danmark	Sjælland	2980
UID_..	..	..	..	..	..	..
UID_N	Mand	43	Gift	Danmark	Fyn	5000

Tabel 3.2: Illustration på et system, der anvender demografisk filtrering

Systemet indeholder nu informationerne om brugeren og hver gang en bruger lytter til en sang, vil denne blive registreret på brugerens profil i systemet. På sigt vil systemet kunne klassificere, hvilken type brugere, der afspiller en given sang. Eksempelvis kan der om sangene fra Burhan G siges, at de brugere der oftest afspiller hans musik er i alderen 13-28 år, single og kommer fra alle landsdele i Danmark. På baggrund af systemets iagttagelser, kan systemet dermed foretage anbefalinger til nye sange, som er af den samme klassifikation af brugere, som kunstneren Burhan G har.

Ulemperne ved anvendelse af demografisk filtrering i et anbefalingssystem er, at anbefalingsmetoden er alt for simpel. Anbefalingerne sker ud fra den generelle type af brugere og dermed bliver metodens største problem, at det anbefaler de samme sange til brugere med samme type klassifikation eller demografiske profil. Dette gør anbefalinger alt for generelle og brugeren derfor ikke specifikke anbefalinger baseret på sin unikke profil, men anbefalinger baseret på sin klassifikations type. En anden ulempe er, at brugeren aktivt (eksplicit) skal indtaste de personlige samt geografiske informationer, før brugeren kan modtage anbefalinger fra systemet.

3.3.6 Ekspert-baseret filtrering

Ved anvendelse af ekspertbaseret filtrering fås, ligesom ved de andre typer af anbefalingsmetoder, en række anbefalinger retur baseret på det input metoden har til rådighed. Den store forskel i den ekspertbaserede filtreringsmetode er dog, at der ikke bliver anvendt en algoritme eller en computer til at foretage beregningerne og anbefalingerne til brugeren. Istedet bliver der anvendt helt almindelige mennesker, som er eksperter indenfor musik.

Samtlige sange brugeren lytter til bliver registreret på dennes profil, som musikeksperterne har adgang til. I stedet for at en maskine gennemgår sangene, der er afspillet af brugeren og muligvis begår fejl, er det i stedet musikeksperter, som

kender de forskellige kunstnere samt musiktyper bedre end en maskine måske kan "læse" dem. På den måde, får brugerne en helt anden type af anbefalinger. Der kan argumenteres for, at brugerne får mere personlige og menneskelige anbefalinger og måske er mere tilbøjelig til at lytte til disse anbefalinger, end hvis der var tale om en maskine som foretog anbefalingerne for dem – brugerne kan relatere til andre mennesker, men ikke til maskiner.

Dog er denne type af anbefalinger ikke helt uden ulemper. Selvom brugeren får anbefalinger fra eksperter, er det stadig menneskeligt at fejle og eksperterne kan ikke vide alt omkring musik. Eksempelvis hvilke kunstnere der passer sammen, samt hvilke sange der passer sammen. Derudover har mennesker forskellige meninger og holdninger til, hvilke sange der passer sammen, samt hvorfor de netop passer sammen. Der er ej heller sikkerhed for, at en musikekspert forholder sig upartisk. Eksempelvis kanekspertenn have en forkærlighed for en bestemt kunstner eller type af musik.

3.4 Evaluering af anbefalingsmetoder

Anbefalingssystemer består af mange forskellige dele, og hver del af systemet arbejder sammen om at foretage de bedste og mest præcise anbefalinger til brugeren. Den mest nødvendige og mest anvendte del af systemet er den, som sørger for at foretage korrekte anbefalinger til brugeren. Uanset om der er tale om informationer fra sange, bedømmelser fra brugerne eller andre informationer, vil der findes en anbefalingsmetode, som kan hjælpe til bedre at forstå brugeren. På den måde vil systemet bedre kunne give anbefalinger på nye sange, som brugeren måske vil kunne lide, baseret på de tidligere afspillede sange.

Som nævnt findes der forskellige typer af metoder, og i visse situationer er nogle bedre end andre. Den som de fleste personer taler om, uden at kende den, er CF, hvis stærke side ligger i anbefalinger foretaget ud fra brugernes bedømmelser. Der vil blive foretaget anbefalinger ud fra en brugers nærmeste naboer, og disse vil være med til at ramme rigtigt med nye anbefalinger til sange. Problemet er imidlertid, at anvendelse af CF kræves utrolig mange ressourcer for, at kunne foretage anbefalinger. Jo flere brugere desto flere ressourcer, hvorfor denne anbefalingsmetode ikke vil være det første valg til implementering i applikationen.

For at minimere mængden af de krævede ressourcer ved anvendelsen af CF, blev CBF valgt, hvor bedømmelserne fra brugerne er skiftet ud med features og indholdet fra sangene. Alle anbefalinger til nye sange sker ud fra hvor mange ligheder en given sang har med andre sange i systemet. Alle disse beregninger kan ske i baggrunden og ligheder blandt sangen skal ikke beregnes, for hver gang der

kommer en ny sang i systemet. Tanken bag applikationen er, at anbefalingerne skal være så personaliseret som overhovedet muligt, hvorfor CBF alene ikke er tilstrækkelig, da der er risiko for, at anbefalinger bliver for ensformige og på den måde alt for generelle.

Ved implementeringen af en kontekst-baseret anbefalingsmetode vil det være muligt for systemet at bestemme, hvilken sammenhæng brugeren befinder sig i, ved afspilning af musik. Denne kontekst bestemmelse er en vigtig faktor til at foretage de mest personaliserede anbefalinger til brugeren. Kontekst-baseret filtrering vil dermed blive anvendt, da netop konteksten har stor indflydelse på brugerens valg af musik. Kontekst-baseret filtrering vil alene står overfor problematikken, at der genereres ensartede anbefalinger, da der ikke er mulighed for at anbefale nye sange ud fra tidligere afspillede sange.

Hverken de demografiske eller ekspertbaserede anbefalingsmetoder er blevet taget med i overvejelserne til implementering i applikationen. De anbefalinger, som den demografiske anbefalingsmetode kommer med er ganske enkelt for generelle og derfor på ingen måde unikke for brugeren. Anbefalinger sker ud fra parametre såsom postnummer og alder, og en bruger vil derfor blive anbefalet en sang baseret på andre brugere fra det givne postnummer samt alder. Problematikken er, at personer er vidt forskellige i smag og dermed også musik. De informationer anbefalingerne foretages ud fra, er derfor utilstrækkelige. Den ekspert-baserede anbefalingsmetode kræver ansættelse af eksperter til bestemmelsen af hvilke sange der passer samme. Mange brugere vil uden tvivl stole mere på anbefalinger fra en ekspert end en maskine, men personaliseringen og kvaliteten vil ikke være god nok.

Der er overvejende mange gode anbefalingsmetoder, men mange af dem begrænses med forskellige ulemper, som kræver mange ressourcer før de kan implementeres i applikationen. Dette kan løses ved at benytte hybrid-baseret filtrering, hvor en sammensætning af flere filtreringsmetoder anvendes. Kombinationen der vil blive benyttet i applikationen er en kombination af indholds-baseret filtrering og kontekst-baseret filtrering. Ved anvendelsen af denne kombination, vil det være muligt for systemet, at foretage anbefalinger på nye sange ud fra forskellige sammenhænge.

Kapitel 4

Designmønstre for kontekst bevidste applikationer

Smartphones bruges i dag i meget varierende og skiftende omgivelser, hvilket har stor betydning for en masse af brugerens beslutninger. Context awareness i mobile systemer er dermed en måde, hvorpå mobilen kan føle dens omgivelser samt forstå sammenhæng og dermed reagere eller tilpasse adfærd derefter. Ikke mindst i forhold til valg af musik har situationen/sammenhængen/konteksten brugeren befinder sig i afgørende betydning. Herunder spiller et væld af faktorer ind såsom; de omkringværende personer, en given aktivitet brugeren udfører, belysning, støjniveau og selv den sociale sammenhæng, fx om brugeren er sammen med en ven, sine kolleger eller sin familie. Disse er alle faktorer, som enten direkte eller indirekte har indflydelse på brugerens valg af musik. Ved at inddrage konteksten, må det antages, at anbefalingerne vil blive mere præcise samt målrettede mod brugerens præferencer og samtidig føre til en styrkelse af brugeroplevelsen.

For at kunne forstå hvilken sammenhæng brugeren befinder sig ud fra en given kontekst er det væsentligt at forstå, hvad kontekst er. I det følgende vil begrebet kontekst derfor blive defineret samt en række kontekstuelle informationer blive identificeret. Derudover følger en gennemgang af hvilke nye aspekter af kontekst, der kan tilføjes de eksisterende anbefalingsparametre, for bedre at kunne beskrive brugeradfærd og dermed danne bedre anbefalinger.

4.1 Definering af kontekst

Ifølge K. Dey, D. Abowd og D. Salbers rapport[12] kan kontekst beskrives som; Enhver information der kan anvendes til at beskrive en given situation, som

er relevant for interaktionen imellem brugeren og applikationen. Kontekst er typisk lokation, identitet, aktivitet, hvilke personer der er i nærheden samt andre fysiske omgivelser.

Selvom begrebet kontekst kan medføre yderst abstrakte fortolkninger, er tanken bag kontekst bevidste applikationer at; kunne benytte bottom-up tilgangen, hvor parametrene bliver tilføjet én af gangen, og hvor fokus er at kunne forstå og kunne håndtere den kontekst, som kan beskrive/føle brugerens sammenhæng. Disse kontekstuelle informationer/parametre, kan deles op i fire forskellige kategorier og er defineret herunder, samt yderligere beskrevet i afsnit 4.3.

Maskinel-kontekst indeholder netværksforbindelse samt ressourcer til fx. Wi-Fi, Bluetooth, SMS, indgående opkald etc.

Bruger-kontekst består af brugerprofiler/identitet, lokation samt sociale sammenhænge.

Fysisk-kontekst er information i form af belysning, støjniveau, bevægelse, vejr og vindforhold. Herunder også temperatur og luftfugtighed.

Tidskontekst er tid såsom; hvilken tid på dagen, ugen, måneden samt årstid.

Formålet med at bestemme hvilken situation brugeren befinder sig i, ud fra en given kontekst, er netop at kunne afgøre eller bestemme, hvilket mål brugeren forsøger at opnå. Det er dog vanskeligt, direkte at afgøre målet brugeren forsøger at opnå, og derfor kan de ovenstående parametre anvendes, til at gøre applikationen i stand til, bedre at udlede hvilket mål brugeren forsøger at opnå. For at kunne indhente disse oplysninger, er der behov for at hente feedback fra brugeren, og det kan ske ved enten eksplicit eller implicit-feedback. Applikationen kan fx. indhente eksplicit feedback, når brugeren indtaster hvilken bevægelsestilstand brugeren er i, eller hvilke personer brugeren er sammen med etc. Hvorimod hvis applikationen anvender implicit feedback, kan applikationen automatisk, uden brugerens interaktion, bestemme, med en vis afvigelse, hvilken bevægelsestilstand brugeren er i vha. data fra accelerometeret. I praksis kan de fleste sammenhænge dog ikke blive bestemt automatisk, og det kan derfor være nødvendigt for applikationen, at modtage eksplicit feedback fra brugeren. I denne rapport vil der blive fokuseret på anvendelsen af implicit feedback, som gør det muligt at bestemme brugerens kontekst. De forskellige typer af feedback er nærmere beskrevet i afsnit 3.2.2.4.

4.2 Brug af kontekst i eksisterende applikationer

Som det ses i afsnit 2.2, så er det de færreste eksisterende applikationer, som forsøger at forstå, hvilken situation brugeren befinder sig i. Der er herunder givet et indblik i, hvordan udvalgte applikationer gør brug af kontekst.

Moodagent benytter sig af bruger-kontekst. Applikationen tager udgangspunkt i brugerens humør, for på den måde at anbefale nye sange, som passer til brugerens nuværende humør. Brugeren angiver eksplicit sit humør i applikationen, og hvis humøret ændrer sig, skal brugeren aktivt angive sit humør igen, for ikke at få upræcise anbefalinger.

Jog.fm gør brug af fysisk-kontekst i form af bevægelse. Brugerens bevægelse bliver registreret implicit fra accelerometer data, som bliver omdannet til BPM[1], som applikationen så anbefaler nye sange ud fra. Det er også muligt for brugeren, at eksplicit kunne angive det ønskede humør, således applikationen afspiller musik i det ønskede tempo, uanset om brugerens tempo ændrer sig.

Nike+ iPod benytter sig ligeledes af fysisk-kontekst, dog danner den givne kontekst ikke grundlag for anbefalingerne. Der bliver i stedet anvendt kontekst til at coache brugeren for på den måde at motivere brugeren under løbeturene. Musikken der bliver afspillet ændrer sig ikke i forhold til konteksten, da den musik der bliver afspillet, bliver hentet fra brugerens valgte afspilningsliste.

Soundrop benytter sig af bruger-kontekst, idet brugerne i fællesskab danner rammerne for hvad man kan kalde en jukebox. Brugerne tilføjer sange til jukeboxen og hver bruger har én stemme til hver sang. Jo flere stemmer en sang får, jo højere op på afspilningslisten kommer sangen, og den sang med flest stemmer bliver afspillet først. Brugerne arbejder derfor sammen om, at få udarbejdet en afspilningsliste, som brugerne er enige om at lytte til.

Slacker Radio og SpotOn Radio benytter ingen form for kontekst i anbefalingerne til brugeren. Applikationerne foretager udelukkende anbefalinger baseret på sanges indholdsparametre. SpotOn Radio og Slacker Radio medregner dog historik over tidligere afspillede, men uden nogen form for tilknyttede kontekst-parametre såsom tid, sted osv.

Last.fm tager samme udgangspunkt som SpotON Radio, dog med den ene forskel, at Last.fm tager højde for tidligere afspillede sange og danner

[1] Beats per minut

anbefalinger på baggrund af afspilningshistorikken. Applikationen giver yderligere mulighed for, at brugerne kan få anbefalede sange fra andre brugere, som ligner brugeren selv (top-N naboer), dog bliver den givne kontekst ikke taget i betragtning.

Generelt set for de eksisterende applikationer på markedet, benyttes der hovedsageligt kun tre kontekst parametre; humør, bevægelse og social sammenhæng. Der er derfor store muligheder i markedet, for at kunne differentiere sig fra de nuværende applikationer set ud fra anvendelse af kontekst.

4.3 Nye kontekst-parametre til musikanbefaling

For at kunne differentiere sig fra de nuværende applikationer er der behov for anvendelsen af nye kontekst-parametre til musikanbefaling. Det vil derfor være muligt med større sandsynlighed, at kunne bestemme brugeres situation og dermed også anbefale mere præcist. Herunder er det beskrevet, hvordan implicit feedback fra sensorer i brugerens smartphone og hvordan brugerens konti på diverse sociale medier kan benyttes til at beskrive, hvilken situation brugeren er i.

Ved et kig på de fysiske kontekstuelle parametre er *bevægelse og aktivitetsniveau* en, der kan sige utrolig meget om brugerens situation. Denne kan som i Jog.fm applikationen udledes via mobilens accelerometer, hvor bevægelses mønsteret kan fastlægges i BPM eller som ingen bevægelse, nogen bevægelse, kraftig bevægelse eller, som i den implementerede bevægelsesdetektor i afsnit 7.4.3, som stilstand, gang, løb eller transport i bil. Herefter kan en applikation enten afspille sange ud fra tempoet eller blot ved at huske, hvad der blev afspillet i en lignende situation, da ikke alle brugere hører højt-tempo musik ved fx løb. Derudover ville belysning samt vejrforhold kan være to interessante parametre i forhold til musik, da disse begge kan være væsentlige faktorer. Belysningen kan udledes via mobilens indbyggede lyssensor og vejrforhold via mobilens internet samt position.

Støjniveauet som parameter kan selvfølgelig også give indblik i de omkringværende omgivelser via mikrofonen, men samtidigt kunne en sådan parameter være behjælpelig i mange andre funktioner. Eksempelvis kunne en intelligent applikation, der selv styrer volumen op/ned ud fra støjniveauet, men samtidigt slår lyden fra ved pludselig støj såsom sirener fra en ambulance eller et dyt fra en bil, forestilles.

Bruger kontekst kan ligeledes udledes implicit i en applikation fx ved adgang til

brugerens Facebook konto, hvor brugerens *profil* direkte kan aflæses mht. civil status mm. Derudover kan *sociale sammenhænge* samt *lokation* udledes, hver gang brugeren checker-in[2] eller nævner venner i en statusopdatering[3]. Lokation kan ligeledes udledes direkte via GPS eller lokationsbestemmelse ud fra sendemaster, men lokation behøver ikke nødvendigvis at være angivet som en direkte position. Det kan blot være i nærheden af noget som fx Zoologisk have og Tivoli eller i form af et WiFi Access Point.

Computing kontekst går på hvilke ressourcer der er til rådighed på mobilen samt, hvilke der er i brug. Eksempelvis kan det undersøges om hvorvidt brugeren bruger mobilen til at spille på, eller måske er et skriveprogram åbent. Er der mange tilgængelige Bluetooth forbindelser, kan det tyde på at brugeren står i en større forsamling.

Tidskontekst er en væsentlig faktor i og med at mange hører forskellige typer musik på forskellige tider af døgnet samt at folk ofte hører forskelligt musik, når det er vinter i forhold til sommer. Samtidigt kan brugeren skifte musiksmag over tid. Dvs. sange der blev afspillet for fx tre måneder siden behøver ikke længere at være interessante. Musik anbefalingerne kunne eksempelvis indeholde en masse julesange i december måned, hvorimod det ville være nogle klassiske sommerhits i juli måned.

4.4 Kontekst bevidst applikation

Som beskrevet i K. Dey, D. Abowd og D. Salbers rapport findes der tre kategorier af kontekst bevidste funktioner (1) dem der præsenterer information og services; (2) dem der automatisk udfører en handling/service; og (3) dem der tilknytter kontekst information til senere brug.

Tanken med musik anbefalingsapplikationen er på sigt; at gøre brug af samtlige beskrevede kontekstuelle parametre samt de tre typer af kontekst bevidste funktioner. Dvs. en applikation, der automatisk tilknytter kontekst parametrene til afspillede sange, således at anbefalingerne bliver genereret, hver gang konteksten skifter. Disse anbefalinger kan enten blive præsenteret til brugeren eller applikationen kan starte afspilning af de anbefalede sange automatisk ud fra, hvilken sammenhæng brugeren befinder sig i. I afsnit 6.2 er detaljerede scenarier beskrevet.

[2] Check-in er en statusopdatering på brugerens profil fx: Michael Kai Petersen - for 4 timer siden i nærheden af Lyngby: "Dejligt vejr" - sammen med Jeppe Andreasen og Ulrik Brink"

[3] Jeppe Andreasen: Griller i haven - sammen med Ulrik Brink

I prototypen vil der være fokus på *bevægelse*, *støjniveau* samt *lokation*, da disse antages at have størst indflydelse på brugerens valg af musik.

Kapitel 5

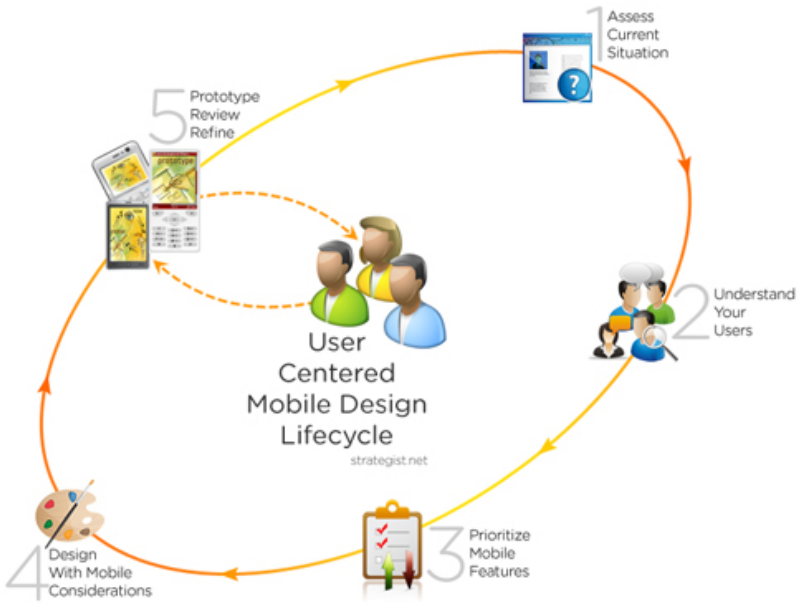
Udviklingsmodel & proces

Der er arbejdet efter feature-driven development[13] (herefter omtalt som FDD) modellerings-principperne. Idet denne udviklingsmetode er agil, vil kravene under hele processen blive revurderet mht. eventuelle ændringer til systemet (applikationen). Agile udviklingsmetoder vil derudover erfaringsmæssigt give en højere kvalitet af det udviklede produkt, og vil, i forhold til kvaliteten, sikre en relativt større produktivitet. Dokumentationen vil blive opdateret parallelt med udviklingen, og herved opnås – foruden en højere kvalitet i form af et mere gennemført produkt – forbedrede omstændigheder i forhold til senere videreudvikling samt vedligeholdelse.

FDD er desuden en utrolig brugerorienteret udviklingsmetode og interaktionsmodellen der ses på figur 5.1 er blevet fulgt. Modellen itererer over samme forløb gentagne gange for derved at opnå et optimalt resultat.

Som det ses på figur 5.1 er der fem essentielle trin i hver iteration.

1. Hvordan er den aktuelle situation, og er der fx behov for endnu et anbefalingssystem.
2. Forstå den enkelte bruger samt lytte til dennes krav og ønsker til en ny løsning.
3. Sammenlign brugerens krav med de forretningsmæssige krav til applikationen. Stemmer de ikke overens ses der på, hvilken værdi disse tilføjer brugeren. Alle krav/features prioriteres.
4. Design applikationen
5. Prototype udvikles, testes og evalueres. Herefter starter næste iteration.



Figur 5.1: Illustration af den agile udviklingsmetode.
Kilde: http://www.smashingmagazine.com/2011/05/02/a-user-centered-approach-to-mobile-design/

Formålet med processen er at udvikle en funktionel prototype, der illustrerer, hvordan applikationen er tiltænkt at virke. Fra idé til den funktionelle prototype er der foretaget tre iterationer, hvor hver iteration indeholder de fem ovenstående trin.

I **første iteration** vil der være fokus på eksisterende løsninger, samt idé generering der kan være med til at differentiere applikationen fra konkurrenterne. Derudover vil idéerne samt skitserne til prototypen blive omsat til brugergrænsefladen i applikationen (GUI) for at visualisere idéen til brugerne.

Under **anden iteration** vil applikationens væsentlige funktioner, der danner grundlag for anbefalingerne, blive implementeret, testet og evalueret.

I **tredje iteration** vil 2. iterations evalueringen blive benyttet til prioritering af de resterende krav/features på listen. Herunder vil anbefalingsdelen mht. dannelse og præsentation af anbefalinger blive implementeret således, at prototypen vil være funktionsdygtig. Derudover bliver applikationen lagt på markedet (Google Play), i form af en beta version.

Da idéen med den iterative arbejdsmodel er, at man direkte kan forsætte med udviklingen i næste iteration, vil der afslutningsvis i 3. iteration være et afsnit om fremtidigt arbejde i næste iteration. Derudover vil de fremtidige planer/muligheder for applikationen i et større perspektiv blive nævnt.

Kapitel 6

1. Iteration

Første iteration bliver brugt til at få overblik over hvilke løsninger, der eksisterer i forvejen, samt generere idéer, der kan være med til at differentiere dette produkt fra konkurrenterne. Derudover vil idéerne samt skitserne til prototypen blive omsat til brugergrænsefladen i applikationen (GUI) for at visualisere idéen til brugerne.

6.1 Overordnet idé med applikationen

Den overordnede idé med applikationen er, at give brugeren et bedre alternativ i forhold til de eksisterende musikanbefalingsapplikationer. Dette gøres ved at fokusere på brugerens givne kontekst og udarbejde anbefalinger ud fra denne.

I prototypen af applikationen vil der være fokus på de tre kontekstuelle parametrer, herunder bevægelse, støjniveau samt lokation indhentet via implicit feedback. Målet ved anvendelsen af brugerens kontekst ud fra de tre kontekstuelle parametrer er, at applikationen skal kunne anbefale nye sange, som passer endnu bedre til brugerens unikke musikprofil i den givne kontekst.

6.2 Scenarie & brugeranalyse

Der lyttes til musik i mange forskellige typer af scenarier heriblandt, på løbeture, cykelture, med bus eller i bil, på arbejdet, en tur på stranden i den blændende sol, indendøre mens regnen siler ned etc., og dette er kun et udpluk af mulige scenarier. Fremadrettet skal det være muligt for applikationen, at automatisk og

uden eksplicit feedback fra brugere, kunne bestemme hvilket scenarie brugeren anvender applikationen i vha. af de førnævnte konstekstuelle parametrer i afsnit 4.1.

Herunder er der beskrevet tre scenarier, hvor applikationen ud fra forskellige parametre har bestemt hvilken kontekst brugeren befinder sig i.

En kold og våd løbetur en onsdag morgen inden dagen rigtigt går igang. En person løber ned af trapperne i opgangen og ud på vejen mod sin sædvanlige tur rundt om søerne i det indre København. Undervejs på turen befinder brugeren sig blandt morgenens tætte trafik og larmen fra trafikken får personen til at skrue højere op for musikken. Tæt ved søerne findes der mange boligblokke og personen løber ofte den samme rute, hvor han indlægger korte og lange spurter blandt dem.

Om formiddagen **tager personen med bussen** ind til sit studie for at følge undervisningen blandt nogle af sine valgte kurser. I bussen om formiddagen er der som regel stadig megen larm i bussen, selvom de fleste folk alle er mødt på arbejde. Personen sidder stille, mens bussen kører gennem indre by og ud mod universitetet.

Sen eftermiddag sidder **personen i stille omgivelser derhjemme** og skriver det sidste på næste uges aflevering.

Baseret på de tre forskellige scenarier kan der foretages en analyse af de forskellige personer, som ville anvende applikationen.

Den aktive person, der anvender applikationen, kan være mange typer af personer, som nyder det aktive liv. Der kan her være tale om personer, som nyder naturen ved at gå en tur i skoven, imens der lyttes til afslappende musik, eller personer, som er idrætsudøvere, der anvender musikken til at få tempoet op og dermed presser dem selv til det yderste. Fælles for disse personer er, at de uanset formålet med aktiviteten, lytter til musik for enten at fremme deres effektivitet eller slappe mere af. Der er her tale om forskellige typer af musik, som der lyttes til alt efter typen af aktivitet.

Den rejsende person, der anvender applikationen kan være alle fra den studerende, som tager det offentlige (bus, tog eller metro) på vej til eller fra sit studie, til forretningsmanden som kører i sin bil til og fra møder ude i byen hos kunder eller til og fra sit arbejde i virksomheden, eller en helt almindelig gennemsnitsperson, som kører til og fra sit arbejde. Alle disse forskellige typer af personer har én ting til fælles og det er, at de alle vil lytte til musik undervejs på deres rejse til og fra forskellige lokationer,

uanset om der er tale om forretningsmanden, den studerende eller den helt almindelige gennemsnitperson

Den stillesiddende , der anvender applikationen kan ligelides som den rejsende, være alle typer af personer, som har et stillesiddende arbejde. Der kunne være tale om den studerende, som lytter til god musik, for at forbedre arbejdet, når der skrives opgave. En anden person kunne være programmøren, som sidder og knokler afsted for at afslutte dagens kode og derfor lytter til god musik, som øger effektiviteten. En tredje person kunne være en gennemsnitsperson, som lytter til musik derhjemme, blot for at koble af efter en lang arbejdsdag.

6.3 Målgruppe

På baggrund af scenarie- og brugeranalysen bestemmes målgruppen for applikationen. Prototypen vil blive udviklet til Android platformen og målgruppen er derfor personer af begge køn i alle aldre, der ejer og anvender mobile applikationer på deres Android smartphones, som samtidigt lytter til musik fra deres smartphones via Spotify-applikationen.

På sigt er målgruppen bredere, da der satses på at udvikle til alle platforme, som er beskrevet i afsnit 2, således at alle musik interesserede brugere med en smartphone vil kunne benytte applikationen.

6.4 Brugerkrav & idé generering

Da der arbejdes med brugerorienteret udvikling, har vi fra start afholdt en fokusgruppe bestående af de fem testpersoner, hvor de kommende testbrugere blev præsenteret for de oprindelige idéer. Her blev der kort fortalt om, hvordan applikationen vil kunne genkende hvilken kontekst brugeren befinder sig i ud fra mobilens hardware såsom: wifiscan, accelerometer samt mikrofon etc.

Fokusgruppen havde til formål at opbygge en indbyrdes dialog blandt deltagerne, hvori mulige idéer blev genereret og vurderet. Derudover blev deltagerne spurgt ind til hvad der skulle til, for at de ville benytte et produkt som dette. Ud af dette kom følgende krav:

- Det skal være nemt og hurtigt at oprette en bruger.

- Det skal være simpelt at navigere rundt og anbefalingerne skal være let tilgængelige.
- Brugerne vil ikke tage stilling til hvilken type musik de hører, da det ofte er blandet.
- Applikationen må ikke påvirke batteriet på mobilen væsentligt.

Samt følgende idéer til applikationen:

- Facebook integration, så de kan logge ind med deres Facebook konto, og dele anbefalinger med deres venner.
- Der skal være mulighed for at kunne få anbefalinger, selvom man skifter mobil, uden at starte forfra.

De fandt den oprindelige idé utrolig interessant, men alle viste dog skepsis over for at åbne mikrofonen, da dette eventuelt kunne bruges til en form for aflytning. Det vil derfor kræve stor tillid til applikationen, hvilket typisk først opstår, når mange tusind brugere har downloadet applikationen i markedet (Google Play). Der vil derfor være fokus på dette i en eventuel videre markedsføring for applikationen, således at det tydeligt fremgår, at ingen lyd optagelser vil blive gemt.

6.5 Kravspecifikation

Ud fra analysen af konkurrenter på markedet i afsnit 2.2 samt brugernes krav og idéer til applikationen er følgende krav blevet opsat i prioriteret rækkefælge:

1. Oprette en bruger.
2. Logge ind/ud
3. Applikationen skal automatisk kunne registrere afspillede sange fra Spotify.
4. Genkende menneskelig aktivitet såsom stilstand, gå, løb eller transport via accelerometer.
5. Bestemme lokation for en afspillet sang.

6. Definere omgivelser ud fra lyd i form af støjniveau lav eller høj.
7. Kunne generere anbefalinger til brugeren ud fra en given kontekst.
8. Præsentere anbefalede sange til brugeren.
9. Starte anbefalede sange i Spotify.
10. Facebook login (denne er ikke essentiel for, at prototypen fungerer, da applikationen stadigvæk skal give mulighed for at oprette en bruger, hvis brugeren ikke har en Facebook konto.)

I denne iteration vil der blive fokuseret på at lave skitse til layoutet samt implementeringen af dette i applikationen ud fra ovenstående krav.

6.6 Design

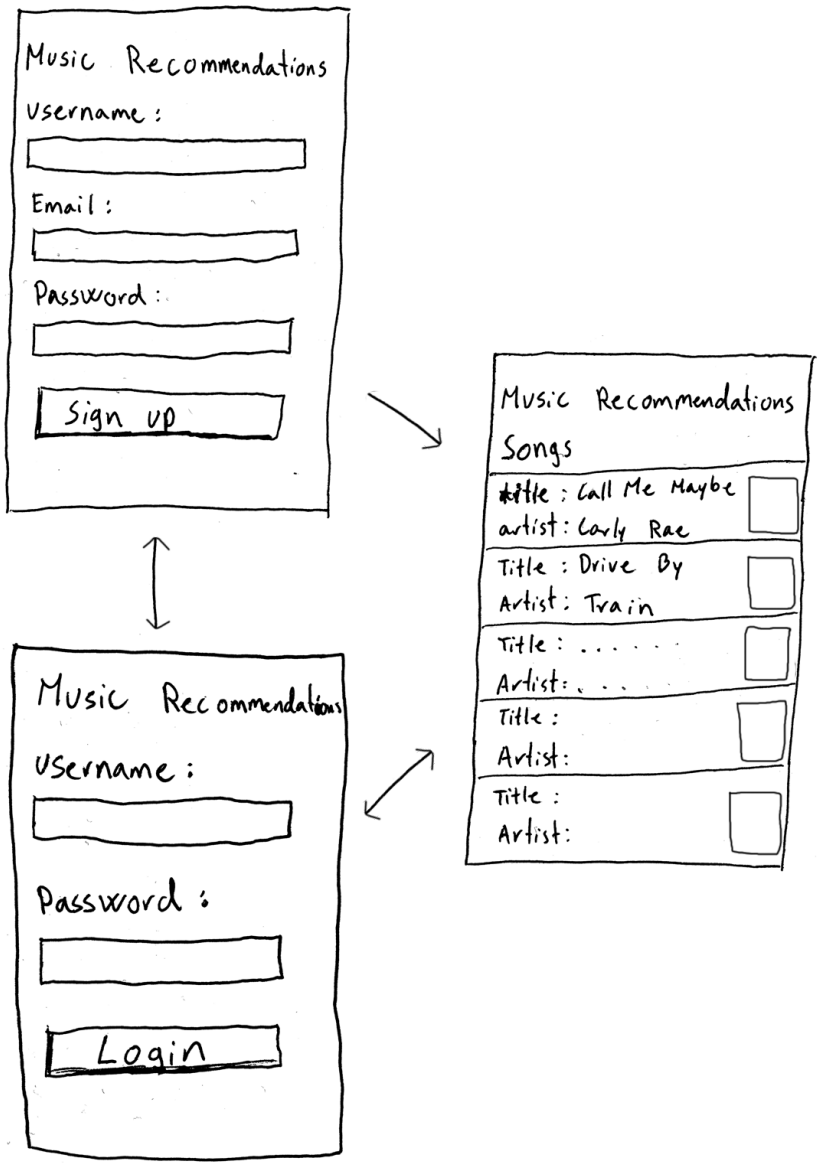
Som tidligere nævnt, i afsnit 3.2.2.3, er det utrolig vigtigt at præsentere anbefalingerne på en ordentlig måde til brugerne. Derudover ønskede de adspurgte testbrugere et design, hvor oprettelsen er simpel og hurtig, samt at anbefalingerne er let tilgængelige.

Med det in mente er layoutet til applikationen blevet skitseret, som vist på figur 6.1, hvor anbefalingerne desuden vil være det første brugerne bliver præsenteret for, såfremt de er logget ind.

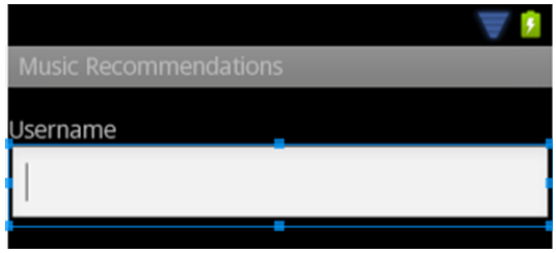
Anbefalingerne vil blive præsenteret med titel, artist samt et coverbilledet af sangen. Dette skulle gerne gøre anbefalingerne lidt mere spændende og derved bevare brugerens interesse.

6.7 Implementering

Da applikationen kun indeholder de tre skærmbilleder; Opret bruger, Login skærm samt præsentation af anbefalinger, er det blevet besluttet at springe mockups processen over, da denne næsten tager den samme tid, som implementeringen af brugergrænsefladen i Android.



Figur 6.1: Skitserne til layoutet samt forbindelserne mellem dem.



Figur 6.2: Figuren viser layout for feltet username.

Applikationens GUI[1] består af tre activities, som repræsenterer de tre før nævnte skærmbilleder. Hver activity's layout styres via xml. Detaljeret beskrivelse af Android komponenter samt activities livscyklus kan læses i afsnit 7.1.

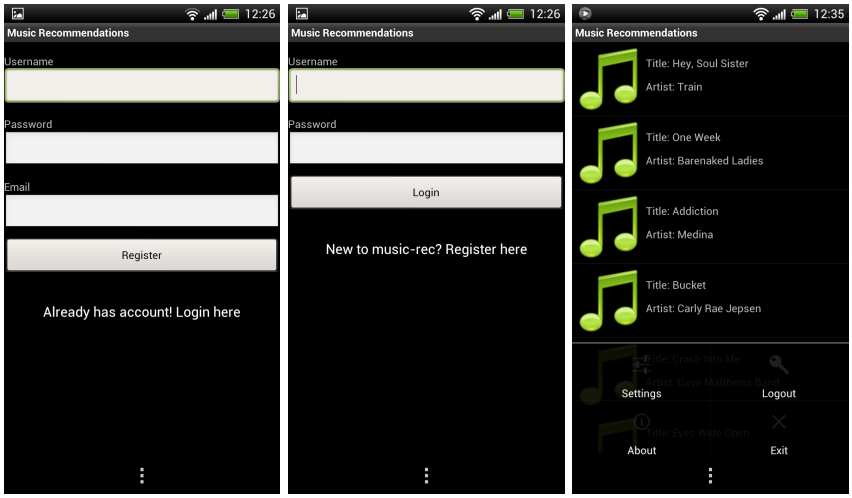
Koden til implementeringen af tekstfeltet 'Username' i loginskærmens layout, ses i bilag A.1, hvor TextView præsenterer teksten 'Username'. Teksten bliver kaldt fra en ekstern placering (ressource filen String.xml). Dette er blevet besluttet, da det gør det utrolig nemt at udvikle en applikation i flere sprog, da alle tekster kan rettes ét sted. Vælger brugeren derfor et anden sprog kaldes blot en anden String ressource i stedet.

EditText er feltet brugeren kan skrive i og typen textVisiblePassword er valgt, da dette viser et keyboard uden specielle tegn for brugeren ved markering. Både EditText og TextView er indkapslet af LinearLayout, som er selve containeren for designet. For at loginskærmen (LoginActivity) benytter layoutet login.xml sættes dette i LoginActivity's onCreate metode, som kode eksemplet i bilag A.2 viser. Resultatet af ovenstående er vist på figur 6.2.

Applikationens layout er lavet i den ganske glimrende editor der følger med Android SDK'et til Eclipse og er senere tilrettet i xml'en, da dette er tidsbesparende. De tre activities der udgør brugergrænsefladen ses på figur 6.3. Menuens layout, der også ses på figur 6.3, er dog lidt anderledes bygget op, da xml layoutet her udelukkende består af en række items i en menu container, der blot henviser til en tekst samt et billedet til visningen i menuen. Koden findes i bilag A.3.

Denne menu bliver indlæst i onCreateOptionsMenu() metoden, som er metoden der bliver kaldt, hver gang en bruger trykker på smartphonens menu knap. Herudover er der implementeret OnClickListeners på alle knapper, der gør det muligt at navigere frem og tilbage mellem de fire activities. Koden for onCreateOptionsMenu() er vedlagt i bilag A.4

[1] Graphic user interface



Figur 6.3: Screenshots af applikationens layout på en HTC OneX.

6.8 Test & feedback

Denne test er afholdt i en fokusgruppe af samme testbrugere og havde til formål at præsentere applikationen visuelt for de fem testbrugere samt lade dem interagere med applikation. Hver bruger har derfor fået den første prototype installeret på sin egen smartphone, som var det en applikation hentet fra markedet (Google Play). På denne måde fås ikke blot brugernes feedback, men applikationen testes samtidigt på en bred skare af smartphones. De testede smartphones er:

- HTC Wildfire S – 2.3.3 Gingerbread
- HTC Desire – 2.3.3 Gingerbread
- HTC One^X – 4.0 Ice Cream Sandwich
- Samsung Galaxy S II – 4.0 Ice Cream Sandwich
- Samsung Galaxy Gio s5660 – 2.3.3 Gingerbread

Resultatet var at applikationen kørte fint på alle de testede smartphones uden force closes[2] eller andre fejl. Desuden var den visuelle prototype grobund for endnu flere idéer til applikationen. Brugerne havde følgende idéer og feedback:

[2]Force close er, når programmet crasher eller tvinges til at standse ved en uventet fejl.

- Applikationen kunne tilgås fra notifikationsbaren, som fx med Spotify under afspilning, når man nemt ville fra Spotify til anbefalingerne.
- Ryste mobilen for at få nye anbefalinger.
- Mulighed for at applikationen selv skifter numre ud fra hvad brugeren laver, er sammen med samt hvor brugeren befinder sig.
- Mulighed for at kunne bedømme anbefalingerne.

Udover idéerne kom brugerne med positiv feedback til det simple design, som de dog synes var lidt kedeligt. Derudover mente de to musik entusiaster, at der nok ville være for få muligheder for at få specikke anbefalinger. De ønskede derfor at kunne skrue på parameterne, der danner grundlag for anbefalingerne.

6.9 Evaluering af 1. iteration

Det at præsentere testbrugerne for en elektronisk prototype i en fokusgruppe var en succes. Ikke mindst fordi applikationen virkede på mange forskellige smartphones, men fordi der blev genereret mange nye idéer.

Muligheden for at kunne tilgå applikationen fra notifikationsbaren er en rigtig god feature, som vil blive prioriteret højt, idet brugeren hurtigt kan tilgå anbefalingerne. Rystefunktionen og automatisk afspilning af sange er to spændende features, men pga. manglende tid, er de dog ikke så højt prioriteret.

Derudover vil der blive ventet med at give brugerne mulighed for at påvirke anbefalingerne i form af at skrue på parametrene samt eksplicit bedømme en sang, da dette oftest ikke er løsningen. Pointen er, at brugerne ikke behøver tænke over, hvilken type sange de hører. Systemet skulle gerne kunne finde frem til dette ud fra den implicite bruger feedback, som er valg af sange i den direkte afspilning.

Designet/layoutet som sådden, udover brugevenligheden, er ikke i fokus under prototype udviklingen og vil derfor ikke blive ændret yderligt.

Kapitel 7

2. Iteration

Anden iteration bliver brugt til få kendskab til Android SDK'et, herunder applikationers opbygning mm. Herudover vil de væsentlige funktioner, der danner grundlag for anbefalinger såsom oprettelse af brugeren samt de valgte kontekstuelle parametre blive designet, implementeret og testet.

7.1 Teori

En del af processen i løbet af projektet har været at sætte sig ind i Androids opbygning samt applikationers livscyklus, da disse er en del anderledes end i Java. Nedenstående beskriver kort, hvordan Android er bygget op mht. applikationer. [3, 2]

7.1.1 Activity livscyklus

Det er vigtigt, at have en grundlæggende forståelse for en activitys livscyklus i Android for at kunne forvalte ressourcerne på enheder med begrænsede ressourcer ordentligt samt sikre en problemfri interaktion fra brugeren.

Alle installerede applikationer kører i deres egen proces, som kan indeholde flere activities og andre komponenter. Hver activity følger livscyklusen, der er illustreret på figur 7.1. Her ses de fire overordnede stadier en activity kan befinde sig i – en applikation kan lige være startet (launched), kørende og synlig for brugeren (running), være afsluttet af brugeren ved metoden finish() (shutdown) eller være nedlagt af systemet pga. manglende ressourcer til højere prioriterede processor/applikationer i stacken (app process killed). Metoderne i de firkantede

bokse kan udføre handlinger, når en activity bevæger sig mellem de forskellige stadier. Fx tjekkes der i den udviklede applikation, at øget tilgængelighed er slået til, hver gang brugeren åbner applikationen, da sangene ellers ikke kan blive registreret. Dette kan der læses mere om i afsnit 7.4.4.

De tre væsentlige levetider/loops i en activity er:

Hele levetiden for en activity, hvilket er fra onCreate(), der kaldes første gang programmet startes – til metoden onDestroy(), der bliver kaldt, når applikationen afsluttes helt.

Den **synlige levetid**, der er mellem onStart() til onStop(). Her vil den være aktiv på skærmen, selvom den dog kan være transperant eller en anden activity kan være i forgrunden.

Forgrundslevetiden er når brugeren kan se og interagere med den givne activity. Denne levetid er mellem metodekaldene onResume() til onPause().

7.1.2 Android komponenter

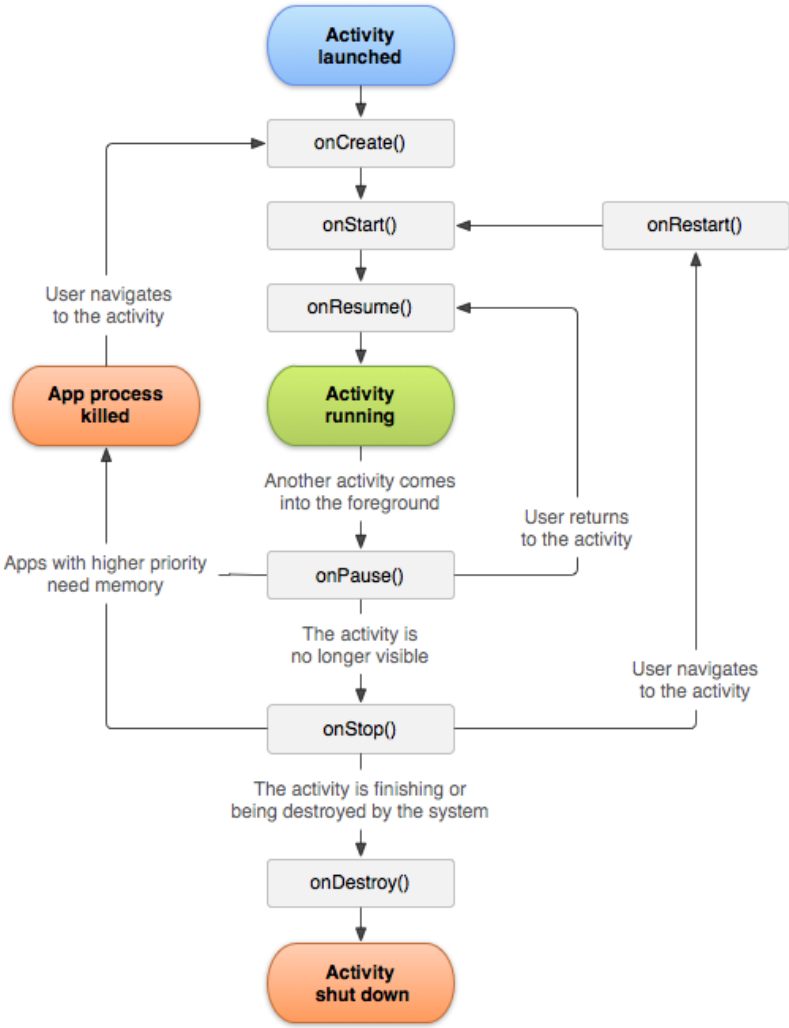
I Android eksisterer der fire essentielle komponenter/byggesten til at udvikle applikationerne med. Hver af dem har forskellige formål samt en klar definition på om komponenten skabes eller ødelægges.

De fire komponenter en applikation kan udvikles med er:

7.1.2.1 Activities

I Android består præsentationslaget (brugergrænsefladen) af activities. En simplificeret beskrivelse er, at hver activity repræsenterer et skærmbillede i en applikation. Det er dog ikke helt så simpelt, da activities også kan blive vist som dialoger eller være transparente. Derudover er alle activities uafhængige af hinanden, selvom de sammen danner en sammenhængende brugeroplevelse. Dvs. en activity kan i princippet startes fra en hvilken som helst anden applikation, hvis dets egen applikation tillader det. En activity bliver lukket ned, så snart den ikke længere er synlig.

Et eksempel på en activity er den visning/skærmbillede der præsenterer anbefalingerne for brugeren.



Figur 7.1: Illustration af en activitys livscyklus, heri vigtige stadier og metoder.
Kilde: http://developer.android.com/reference/android/app/Activity.html

7.1.2.2 Services

En service er en komponent, der udfører længere varende operationer i baggrunden eller udfører arbejde for andre processer/applikationer. Denne komponent har ingen brugergrænseflade, som brugeren kan interagere med. En komponent som fx activity kan starte en service og lade den køre i baggrunden eller binde/unbinde til den, hvis der skal interageres med den. En service kan derfor modsat en activity køre uhindret i baggrunden, indtil Android lukker denne ned, hvis andre vigtigere processer kræver flere ressourcer.

Et eksempel på en service er applikationens registrering af afspillede sange fra Spotify, der konstant kører i baggrunden. Denne kan der læses mere om i afsnit 7.4.4.

7.1.2.3 Content providers

En content provider bruges til at styre/dele data applikationerne imellem. Disse data kan være gemt i det lokale fil system, i Androids SQLite database, på internettet eller andre steder applikationen kan tilgå. Via provideren kan andre applikationer tilgå/ændre data, hvis dette tillades af provideren. Content providere kan dog også bruges til at gemme data internt i applikationen.

Et eksempel på en content provider er Androids kontaktpersoner, som alle applikationer kan tilgå og ændre.

7.1.2.4 Broadcast recievers

En broadcast reciever bruges til at opfange og reagere på de meddelelser, der bliver sendt til/af systemet. En meddelelse fra systemet kan fx være at skærmen er slukket eller, at der netop er blevet taget et billede med kameraet. En broadcast reciever kan interagere med brugeren gennem notifikationer, dog bruges recieveren oftest kun til at starte en activity eller en service.

7.1.3 Aktivering af komponenter

For at opstarte komponenterne activity, service eller broadcast receiver benyttets en intent. En intent definerer den handling, der skal ske. Fx ved tryk på en knap, så affyres en intent, der starter den nye activity eller service.

Et eksempel på en række intents er, når brugeren trykker på en anbefaling og disse starter Spotify op, søger efter den valgte sang samt starter afspilningen af den valgte sang.

7.1.4 Android manifest

Før Android kan starte en komponent skal denne være specificeret i manifest filen (AndroidManifest.xml). Dvs. alle applikationens activities, services, broadcast recievere samt content providere skal være defineret med deres placering i applikationsstrukturen samt diverse andre attributter efter behov. Det er også her applikationens minimums API niveau, samt hvilke features applikationen kræver adgang til (user permissions/rettigheder) bliver sat. Fx minimum API version 2.3.3 Gingerbread og fuld internetadgang, som i denne applikation.

7.2 Kravspecifikation

Da implementeringen udelukkende bestod af GUI i 1. iteration er ingen af de funktionelle krav opnået endnu. Fremvisningen af første prototype gav dog anledning til flere krav og idéer, som er blevet tilføjet kravspecifikationen.

1. Oprette en bruger.
2. Logge ind/ud
3. Applikationen skal automatisk kunne registrere afspillede sange fra Spotify.
4. Genkende menneskelig aktivitet såsom stilstand, gå, løb eller transport via accelerometer.
5. Bestemme lokation for en afspillet sang.
6. Definere omgivelser ud fra lyd i form af støjniveau lav eller høj.
7. Kunne generere anbefalinger til brugeren ud fra en given kontekst.
8. Præsentere anbefalede sange til brugeren.
9. Åbne applikationen fra notifikationsbaren, når applikationen kører.
10. Starte anbefalede sange i Spotify.

11. Automatisk afspilning af sange ud fra en given kontekst.
12. Rystefunktion til at opdatere listen med anbefalede sange.
13. Facebook login

Hvoraf implementering samt test af de første 6 punkter vil være i fokus i denne iteration. Fremvisningen gav desuden ingen ændringer til de ikke funktionelle krav fra 1. iteration i afsnit 6.4.

7.3 Design

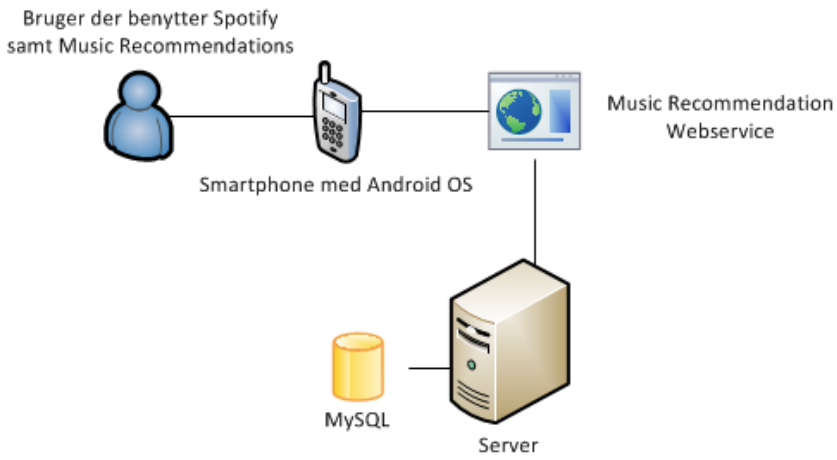
Grundet markedsanalysen i afsnit 2.5, valg af anbefalingsmetode i afsnit 3.4 samt brugernes krav om at kunne skifte mobil kræver applikationen adgang til en ekstern database, hvori all bruger information bliver gemt, i stedet for i mobilens lokale SQLite database.

Da applikationen endvidere ønskes udbredt til flere platforme på sigt, er den klassiske client-server model derfor benyttet, og en webservice vil håndtere al kommunikation mellem applikationen (client) og serveren i form af HTTP requests. Serveren vil herefter tage sig af logikken bagved enhver anbefalingsmetode applikationen måtte benytte. Dette har den fordel, at når backend fungerer, kan applikationen forholdsvis hurtigt distribueres til et hav af andre platforme, da den native[1] applikation derved kun består af GUI'et til at præsentere de anbefalede sange til brugeren samt metoder til at registrere afspillede sange og brugerens kontekst. Figur 7.2 illustrerer det foreløbige system.

7.3.1 Android applikation

Den native applikation har til formål at præsentere anbefalingerne til brugeren. Derudover er en af applikationens væsentligste opgaver at indsamle information, der kan danne grundlag for anbefalingerne. I denne iteration vil informationen bestå af accelerometer og mikrofon data, wifi placering samt afspillede sange. Nedenfor er overvejelserne omkring hvert element beskrevet yderligere.

[1] En applikationen der er udviklet til at kører i et bestemt miljø fx Android eller iOS



Figur 7.2: Illustration af det foreløbige System

7.3.1.1 Mikrofon data

Lydniveauet kan sige meget omkring brugerens omgivelser. Er der stille, kan brugeren eksempelvis være hjemme eller på kontoret i stille omgivelser, hvorimod meget støj kan betyde offentlig transport såsom bus eller tog. Mobilens mikrofon data kan benyttes til at bestemme støjniveauet, som er yderligere beskrevet nedenfor.

Data fra mobilens mikrofon fås som et short[] array indeholdende værdier, der betegner signalet digitalt. Disse værdier er et udtryk for den øjeblikkelige elektriske spænding fra mikrofonen, der svinger fra +32767 til - 32768 og modtages hvert sekund. I denne rapport vil der ud fra disse data, blive forsøgt at bestemme støjniveauet udfra en simpel gennemsnitsberegning af den absolutte værdi over tid. Data vil derfor blive indsamlet i ovenstående scenarier og analyseret for dermed at finde det absolutte gennemsnit der angiver støjniveauet i de givne scenarier, som enten højt eller lavt. Dette er beskrevet i implementeringen i afsnit 7.4.2.

7.3.1.2 Accelerometer data

Selv den mindste bevægelse brugeren foretager sig vil give udslag på smartphonens accelerometer. Disse informationer kan benyttes til at beregne/gætte på en brugers udførte handling i form af fx stilstand, gå, løb, transport, som denne applikation vil fokusere på.

$$energy_t = |x_t - x_{t-1}| + |y_t - y_{t-1}| + |z_t - z_{t-1}|$$

hvor x, y og z værdierne er sensor værdier til et givent tidspunkt t.

Figur 7.3: Summering af accelerometer værdier.

Accelerometerets data giver en x,y og z værdi, som summeret giver en samlet G påvirkning vha. af formlen vist på figur 7.3, hvor der ligeledes korrigeres for jordens tyngdeacceleration på ca. 1G.

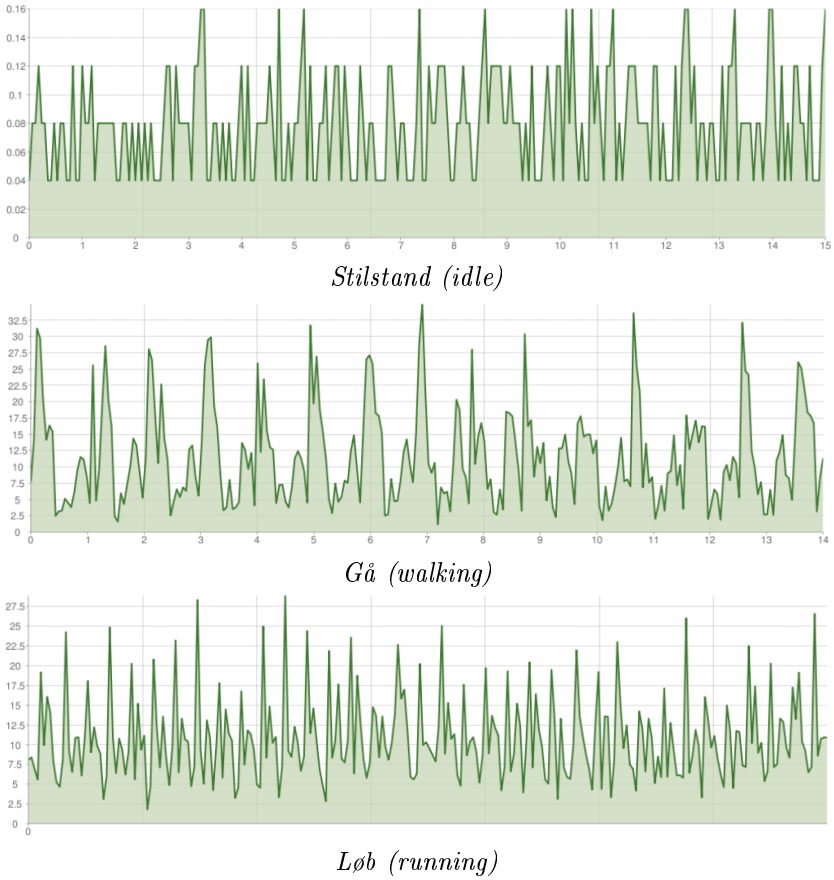
Som Peter Frøkjær Strand beskriver det i sin rapport[22] har hver gangart, som fx gå og løbs summerede påvirkning et unikt mønster/udslag, som vist på figur 7.4. Ved fx at benytte algoritmen Fourier Transform[2] til at analysere disse udslag kan det derved fastlægges om brugeren, der bærer smartphonen, er i færd med at gå eller løbe. Ligeledes kan det selvfølgelig fastlægges om brugeren er i stilstand, da der ingen eller meget små udslag vil være. Er brugeren derimod i stilstand i længere tid, men samtidigt skifter lokation, kan brugeren dermed antages at være under transport i form af bil, bus eller tog.

Da anbefalingerne er kontekstbaserede, vil det beregnede gæt derfor være en vigtig parameter til grundlag for anbefalingerne.

7.3.1.3 Wi-Fi

Wi-Fi netværk er omkring os overalt i dag i form af "signal bølger". Disse bølger bliver konstant udsendt fra diverse accesspoints og er til stede derhjemme, på arbejdet, i skolen, i toget samt bussen osv. Dette kan derfor sige noget om brugerens sociale kontekst mht. hvem brugeren er sammen, hvilket er en betydelig parameter i brugerens valg af sange. I applikationen vil den indbyggede Wi-Fi scanner dermed blive brugt til at identificere, hvilke netværk brugeren er i nærheden af under de afspillede sange. En sådan scanning vil oftest give utrolig mange Wi-Fi netværk og der vil derfor være behov for at filtrere dette output for at finde frem til, hvem brugeren egentlig er sammen med. I denne applikation antages det, at det bedste netværk er det tætteste eller ihvertfald det mest sigende om brugerens sociale kontekst.

[2] Fourier Transform



Figur 7.4: Graf der viser mønstret for den summerede G påvirkning under stilstand, gå samt løb.
Kilde: Peter Frøkjær Strands rapport [22]

7.3.1.4 Afspillede sange

For at kunne anbefale nye sange til brugeren kræves noget input i form af historik over tidligere afspillede sange, som nævnt i afsnit 3.2. Endvidere ønskes kun de sange brugeren rent faktisk har lyttet til medregnet i anbefalingsalgoritmen, således at en sang ikke bliver medregnet ved afspilning i fx to sekunder. Last.fm benytter sig eksempelvis af noget lignende, hvor en sang kun bliver scrobblet[3], hvis 50% eller mere af sangen er afspillet. Dette skulle gerne resultere i mere præcise anbefalinger pga. den mindre støj i form af fx sange der hurtigt bliver startet for igen at blive skippet.

Spotify tilbyder i skrivende stund tre API'er, der alle har været i overjelserne til at få infomation om den igangværende sang i Spotify[20]

Metadata API er et web API, som kan bruges til at slå op i Spotify's musik katelog. Desværre kan hverken igangværende sang eller tidligere afspillede sange tilgås via dette API.

Spotify Apps API er et API, hvor man ved brug af HTML5, CSS og JavaScript kan udvikle tredjeparts applikationer til Spotify. Dette API giver adgang til stort set alle features i Spotify, såsom at se den igangværende sang, tidligere afspillede, oprette playlister samt tilføje sange til afspilningskøen. Desværre giver Spotify kun mulighed for, at brugerenkan installere de udviklede applikationer på enten Mac OSX eller Windows. Vi kan derfor ikke bruge dette API, da vi er interesserede i afspillede sange fra brugerens smartphone.

Libspotify er et C API, hvori de fleste Spotify features kan benyttes. Dette kan vi dog ikke benytte til Android platformen. Derudover kræver det, at man opretter sin egen medieafspiller, hvori man tilgår Spotify's streaming services.

Derudover giver Spotify brugeren mulighed for at dele sine afspillede sange med andre via Facebook og Last.fm. Følgende API'er har dermed også været med i overvejelserne:

Last.fm API: Da Spotify giver brugeren mulighed for at scrobble til last.fm dvs. tilføje afspillede sange til brugerens Last.fm konto har dette API også været med i overvejelserne til en måde, hvorpå vi kan få afspillede sange fra. Den igangværende sang kan dog ikke ses direkte, men kun tidligere

[3] At scrobble er når afspillede sange automatisk bliver sendt til Last.fm's database.

afspillede sange med det eksakte tidspunkt for afspilningen. Det ville derfor være muligt at lave en løsning med dette API, da det eksakte tidspunkt for hver afspillet sang er tilgængeligt. Dog ville det kræve, at mobilen hele tiden havde accelerometer samt wifi scanneren tændt for at vide, hvilken kontekst brugeren befandt sig i under de afspillede sange. Derudover ville der være risiko for, at afspillede sange på brugerens computer også ville blive scrobblet til Last.fm, da brugeren ellers aktivt skal gå ind og ændre indstillinger i Spotify, hver gang der skiftes enhed.

Facebook Open Graph API : Spotify giver ligeledes brugeren mulighed for at dele afspillede sange med sine venner på Facebook. Dog gælder samme problemstilling mht. konstekstuelle parametre, som ved brug af Lasf.fm.

Målet var derfor at finde en måde, hvorpå den igangværende sang fra Spotify kunne registreres direkte fra mobilen. Dette viste sig, at være et problem, som der kan læses mere om under implementeringen 7.4.4.

7.3.2 Webapplikation

Som tidligere nævnt er det webapplikationen/serverens opgave at tage sig af logikken mht. anbefalinger, oprettelse af brugere og login. Herunder vil webapplikationen indeholde en webservice, der har til formål at håndtere kommunikation mellem webapplikationen og mobilapplikationen.

Designmønstret Model-View-Controller vil derfor blive benyttet til webapplikationen. Dvs. når mobilapplikationen sender et request, vil dette request blive uddelegeret til den korrekte controller, som herefter vil udføre de krævede handlinger for derefter at returnere et respons til mobilapplikationen. Requests fra mobilapplikationen sker i form af HTTP request.

Webservicen vil blive implementeret som en RESTful webservice, der benytter JSON[4], således at alt fungerer som ressourcer samt respons direkte returneres i form af JSON. Et request til serveren kunne eksempelvis se således ud http://domæne.dk/songs, som ville returnere en liste sange i JSON format.

[4] JavaScript Object Notation

7.4 Implementering

I denne iteration har der, som tidligere nævnt, været fokus på implementeringen. Herunder er implementeringen for de væsentligste elementer i protoypen indtil videre beskrevet.

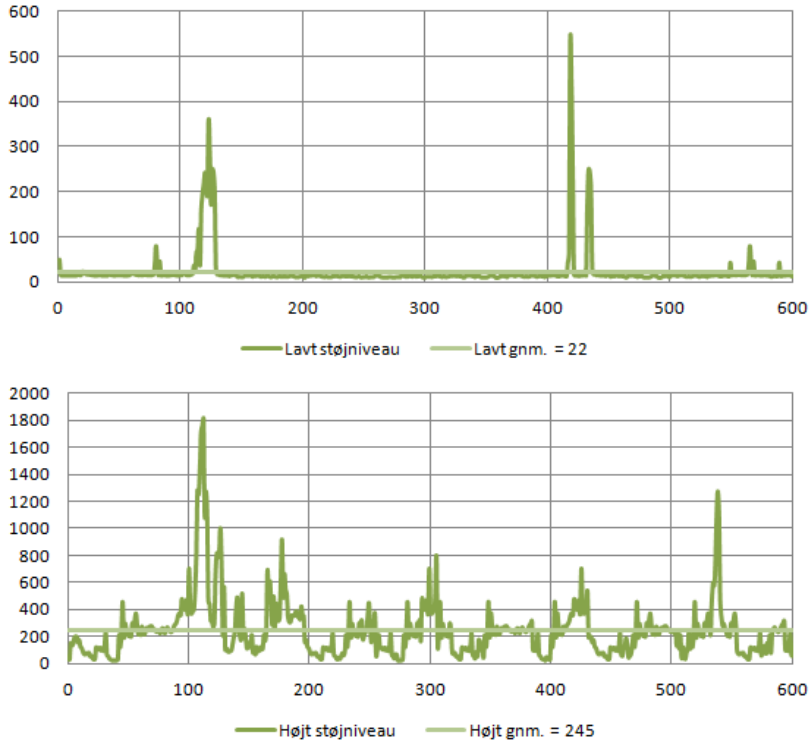
7.4.1 Wi-Fi scanner

Til at scanne efter bedste WiFi benyttes Android SDK'ets WifiManager. Denne klasse indeholder API'et til styring af alle aspekter mht. WiFi forbindelser, der kan anvendes til at returnere en liste af alle tilgængelige WiFi netværk fra systemets sidste scanning. Denne liste løbes blot igennem og netværket med bedste signal findes og returneres. Når bedste WiFi signal bliver tilknyttet en afspillet sang, vil det være i form af Access Pointets MAC adresse (BSSID). Koden er vedlagt i bilag A.5.

7.4.2 Støjniveau detektor

Til at bestemme støjniveauet er oprettet en service SoundService, der kører i baggrunden. Ved opstart af denne service bliver AudioRecorder initialiseret, hvorefter en tråd oprettes til at indhente data fra mikrofonen. Denne tråd kalder blot metoden en metode (*getSamples()*), hvori selve optagelsen startes. I denne metode sørger en while løkke for at indlæse data direkte i en buffer, hvilket har den fordel, at optagelsen ikke behøver at blive gemt før analyse af data. Disse data bliver indlæst hvert sekund og den absolutte total værdi bliver udregnet og lagt i et array *largeSection*, der skal udregne gennemsnittet for en periode på 30 sekunder for dermed at sikre, at pludselige lyde ikke slører resultatet for meget. Er gennemsnittet for perioden på 30 sekunder over 200 vil støjniveauet bliver betegnet som højt, hvorimod alt under vil blive betegnet som lavt.

Grænsen på netop 200 er sat på baggrund af de indsamlede data, som ses på figur 7.5, hvor der er optaget lyd i 10 minutter på et stille kontor samt under en bustur. Det ses at gennemsnittet ligger lige over 200 ved højt støjniveau, hvorfor grænsen er sat til dette. Højt støjniveau betegnes i denne implementering, som vedvarende baggrundsstøj såsom personer der snakker normalt eller højere. Det lave støjniveau har et gennemsnit på 22, da stilheden er blevet afbrudt hurtigt et par gange. Der er dog langt op til de 200, som betegner højt støjniveau. Koden for while løkken ses i bilag A.6.



Figur 7.5: Figuren illustrerer støjnivauet på et stille kontor (øverst) samt under en bustur (nederst).

7.4.3 Bevægelsesdetektor (Motion detection)

Bevægelsesdetektoreren er implementeret med udgangspunkt i Peter Frøkjær Strands ContextResolver[22]. Denne var udviklet som en applikation, der broadcastede det summerede gæt til alle applikationer. Al funktionalitet mht. algoritmen er dermed blevet implementeret direkte i prototypen i stedet, for at undgå to applikationer, hvoraf den ene blot broadcaster til den anden, som derefter modtager det summerede gæt via en broadcastreceiver.

I prototypen er der oprettet en service SensorService, der kører i baggrunden og modtager events fra sensoren vha. metoden onSensorChanged(), hvori accelerometer værdierne behandles. Først bestemmes den øjeblikkelige bevægelse ud fra en mindre indsamlet session, i metoden handleSmallSection(), i form af stilstand (idle), minimal (moving_sm), mellem (moving_me), og høj bevægelse (moving_hi), gang (walking) eller løb (running). Herefter beregnes gennemsnitshastigheden i forhold til sidste event, og MovementStateManageren der indeholder alle de små sessioner, beregner et summeret gæt på brugerens bevægelse. Har brugeren været i stilstand i mere end et minut, men lokationen er blevet ændret (bedømt ud fra GPS) vil det summerede gæt være vehicle. Således må det antages at brugeren er under transport. Seneste gæt samt det summerede gæt bliver til sidst gemt i den lokale SQlite database, således at det senere kan tilknyttes en afspillet sang, som en kontekstuel parameter. Koden for metoden handleSmallSection() er vist i bilag A.7.

7.4.4 Registrering af sange fra Spotify

Som nævnt i afsnit 7.3.1.4, tilbyder Spotify ikke et API til Android, hvor de nødvendige features kan tilgås[5]. Sangene afspilles endvidere i Spotify's egen medieafspiller. Disse kunne derfor ikke tilgås, som var det afspillet fra mobilens indbyggede medieafspiller. Desuden har Android heller ikke en fast indbygget medieafspiller i kernen og producenterne laver derfor deres egen, hvilket ville betyde, at der højst sandsynligt skulle udvikles forskellige løsninger til alle smartphone modeller.

Alle Android applikationer kan dog nemt decompiles, så kildekoden kan aflæses. Dette blev gjort for at undersøge om Spotify broadcastede information om valgte sange, som kunne modtages i applikationen vha. en broadcastreciever7.1. Desværre bliver intet broadcastet fra Spotify applikationen, men heldigvis sendes

[5] API'et er dog udkommet under projektets færdiggørelse, hvilket nævnes under fremtidsplaner 10

en notifikation til statusbaren med sangtitel samt kunstner, hver gang brugeren starter afspilningen af en ny sang eller næste sang i køen begynder.

Denne notifications tekst kan tilgås ved, at brugeren giver applikationen rettigheden til øget tilgængelighed på smartphonen, hvilket kun kan gøres manuelt af brugeren pga. sikkerhed. Denne indstilling er essentiel for at applikationen virker korrekt, da applikationen ellers vil være meningsløs, hvis ikke afspillede sange kan registreres. Der tjekkes derfor om denne rettighed er givet hver gang brugeren starter applikationen.

Næste udfordring er, at kun sange der bliver afspillet i en sektion af minimum 30 sekunder ønskes registreret for at sikre sig, at brugeren rent faktisk har lyttet til sangen.

Ved at opsnappe og analysere de notifikationer Spotify udsendte ved brug, blev skemaet i tabel 7.1 udarbejdet.

Udfra skemaet i tabel 7.1 ses det, at kun de events der starter en afspilning såsom Play, Previous og Next song indeholder "-", som adskiller sangtitlen og kunstnernavnet. Derfor er der tale om en sang, hver gang der udsendes en notifikation, der indeholder en bindestreg fra programmet Spotify, som notifikationerne filtreres på. Med en timer kan det derfor fastslås hvor længe en sang er blevet afspillet. Timeren bliver nulstillet ved hver event, der ikke indeholder en bindestreg. Dvs. at afspilles der en sang fx "Criminal Mind – Lukas Graham" i minimum 30 sekunder, hvorefter Spotify automatisk går videre til næste sang eller brugeren aktivt trykker videre, så bliver sangen gemt som afspillet, da næste event "Drive By - Train" indeholder en bindestreg og timeren er over 30 sekunder. Havde brugeren nulstillet timeren ved fx at trykke rundt i programmet ville sangen ikke blive gemt, da timeren herved nulstilles ved events, der ikke indeholder bindestregen. Tomme strenge ignoreres desuden, hvilket betyder at brugeren godt kan starte en afspilning og trykke sig tilbage og ud af Spotify. Når Spotify så går videre til næste sang vil den afspillede sang efterfølgende blive gemt som i eksemplet ovenfor.

Denne SpotifyGrabber[6] er lavet som en service, der nedarver fra klassen AccessibilityService. AccessibilityService er en service der kører i baggrunden og modtager respons fra Android OS hver gang AccessibilityEvents bliver "affyret". Disse events beskriver tilstande og overgange i brugergrænsefladen/userinterface fx. Når fokus skifter eller når der bliver trykket på en knap etc. Servicen nedarver følgende metoder:

- onAccessibilityEvent(AccessibilityEvent event)

[6] Servicen i implementeringen kaldes SpotifyGrabber

Action/event	*Notifikation/besked*
Start af program	"Spotify"
Play	"Titel - Kunstner"fx "Drive By - Train"
Pause	////
Previous song	//// efterfulgt af "Titel - Kunstner"
Next song	//// efterfulgt af "Titel - Kunstner"
Search	"Search"
Tracks	"Tracks"
Albums	"Albums"
Artists	"Artists"
Playlists	"Playlists"
What's New	"What's New"
Top Tracks	"Top Tracks"
Feed	"Feed"
Settings	"Settings"efterfulgt af "Spotify"
More	"More"
Øvrige såsom: About, Inbox, Library etc.	////
Alle events	Efter alle events udsendes ////

Tabel 7.1: Skema over events i Spotify samt den opfangede besked vha. øget tilgængelighed.

- onBind(Intent intent)
- onInterrupt()
- setServiceInfo(AccessibilityServiceInfo info)

Metoden setServiceInfo() benyttes til at sætte AccessibilityServiceInfo, som beskriver den øget tilgængelighed service. Pakkenavnet der ønskes fra AccessibilityEvents er sat til "com.spotify.mobile.android.ui", som er Spotify's pakkenavn. Derfor vides det, at alle de events metoden onAccessibilityEvent() modtager stammer fra Spotify og der skal dermed ikke tages højde for det. Metoden setServiceInfo er vist i bilag A.8.

Er teksten fra den indkomne event er tom ignoreres dette og ingen handling sker, men er teksten ikke tom, bliver der tjekket om denne indeholder en bindestreg og dermed er en sang. Hvis ja, bliver sangen gemt i arrayet playedSongs. Herudover tjekkes der hele tiden for om der er gået minimum 30 sekunder siden sidste event, og er der det, startes en ny tråd, hvori sidste sang fra playedSongs (den netop afsluttede sang) bliver valgt. Desuden søges efter det stærkeste wifi signal i nærheden samt andre parametre såsom motion og sound. Disse bliver knyttet til playedSong og sendt af sted mod webservicen. Koden for onAccessibilityEvent ses i bilag A.9, der viser hvordan sangene fra Spotify bliver "grabbet".

7.4.5 Webapplikation

Webapplikationen er som Android applikationen implementeret i Java og frameworket Spring MVC er benyttet, da dette gør det nemt at implementere ud fra designmønstret Model-View-Controller. Herudover er Maven benyttet til at opdele applikationen i moduler, hvor et multi-modul vil samle de fire moduler til én samlet applikation. Herunder er hvert modul kort beskrevet.

music-rec samler de fire nedenstående moduler til ét.

music-rec-model er modulet, hvor domæne objekterne bliver defineret og fungerer som datagrundlag for alle modulerne.

music-rec-persist er modulet, der indeholder alle DAO'erne[7]. Det er dette modul, der har kontakt med databasen, og alle forespørgsler til databasen vil gå igennem dette modul.

[7] Data Access Objects

music-rec-service er modulet, hvor hele forretnings - og anbefalingslogikken ligger. Det er dermed dette modul, der genererer anbefalingerne, står for oprettelse af brugere samt verificerer ved login.

music-rec-webapp er modulet der konfigurerer selve webservicen, hvor HTTP request bliver modtaget og respons sendt tilbage til mobilapplikationen i form af JSON.

I denne iteration er webservicens primære funktioner at stå for oprettelsen af brugere, verificere login samt gemme de afspillede sange i databasen. Herunder vil processen, fra modtagelsen af request til respons sendes tilbage til mobilapplikationen, blive beskrevet, for registreringen af en afspillet sang.

POST requestet med den givne sang, bliver sendt til http://domæne.dk/song, og modtages af Controlleren, der udløser de nødvendige metoder for at registrere den afspillede sang. Først kaldes servicen InputHandler, som udføre logikken bag registreringen. Herefter returneres sangen, samt en accept ved korrekt oprettelse eller en fejlkode ved fejl i oprettelsen, til applikationen i form af JSON. Til at parse objekterne til JSON benyttes API'et JacksonMapper[10]. Koden for Controlleren er vist i bilag A.10.

Logikken bliver udført i InputHandleren, som er vist i bilag A.11. Eksisterer brugernavnet fra det modtagne request kontrolleres adgangskoden. Er adgangskoden korrekt vil metoden addPlayedSong blive kaldt. I metoden addPlayedSong vil der blive kontrolleret, om sangen tidligere er blevet tilføjet. Eksisterer sangen, vil id'et blive returneret med det samme, hvis ikke vil sangen blive oprettet og id'et returneret. Herefter vil brugerens afspilning blive registreret med de tilknyttede kontekstuelle parametre. Er sangen allerede blevet afspillet i den samme kontekst vil antallet af afspilninger blive talt op i databasen. Kontakten til databasen sker gennem playedSongDao'en, hvis metoder ikke er nødvendige at vise. Koden for metoden addPlayedSong er vist i bilag A.12.

Ovenstående princip gælder for alle request, herunder oprettelse af brugere samt verificering af login. Disse kan findes i den vedlagte kode.

7.5 Test & feedback

Denne test havde til formål at teste den implementerede funktionalitet i applikationen på alle testbrugernes smartphones, samt indsamle data omkring brugernes afspilninger.

Den samlede test var derfor delt op i to dele, hvoraf den første del, som i 1. iteration i afsnit 6.8, blev afholdt i en fokusgruppe bestående af de fem testbrugere. Her fik alle installeret den nye prototype, hvorefter de fik en gennemgang for brug af applikationen og oprettede deres brugere.

For at teste funktionaliteten, der indsamler og tilknytter de kontekstuelle parameter *bevægelse* samt *lokation* til afspillede sange, blev brugerne sat til at afspille en oprettet playlist under handlingerne stilstand, gå, løb og kørsel i bil. Hver handling blev udført i 15 minutter, hvilket svarer til ca. fire sange. Test af funktionen, der indsamler den konstekstuelle parameter *støjniveau*, blev herefter udført således at brugerne først sad stille og arbejdede foran en computer i samme rum, for derefter at holde fokusgruppe, hvor der blev snakket normalt eller højere. Disse test blev gennemført med succes og de korrekte parametre blev tilknyttet sangene under de givne aktiviteter.

Herefter fulgte anden del, hvor brugerne i løbet af en uge skulle bruge applikationen i praksis. Brugerne skulle dermed blot høre musik, som de plejede via Spotify med applikationen kørende i baggrunden, således at vi kunne teste applikationen under diverse forhold. Samlet set havde brugerne følgende feedback under testperioden:

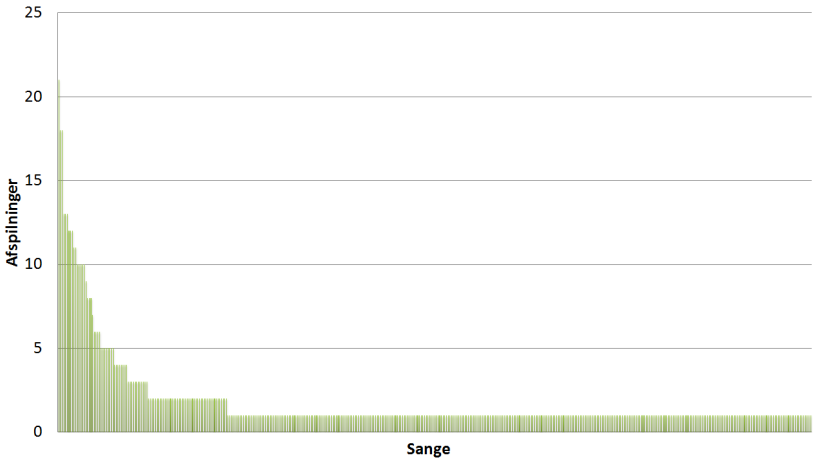
- Force close ved login, når der er dårlig forbindelse.
- Ligner at applikationen fryser under login/oprettelse.

Derudover havde brugerne ikke oplevet nogle problemer med applikationen og de var spændte på at få genereret anbefalinger.

7.6 Analyse af indhentet data

Prototypen her i 2. iteration har indhentet data fra testbrugerne og de sange de har afspillet. Nogle sange er blevet afspillet flere gange end andre og samtlige sange er blevet rangeret efter antal afspilninger og herefter illustreret i figur 7.6. De indhentede data ligner tilnærmelsesvis den generelle form for The Long Tail, der er illustreret på figur 1.1.

Det ses tydeligt, at nogle sange er blevet afspillet væsentlig flere gange end andre sange, som måske kun er blevet afspillet en enkelt gang. Der lader til, at være et mønster i de afspillede, idet mest afspillede sange går igen på top-listerne i



Figur 7.6: Figuren illustrerer longtail blandt testbrugerne

Spotify og at de mindst afspillede sange ligger nederst eller slet ikke findes på top-listerne i Spotify.

De tre faktorer af Chris Anderson [1]; *forskellighed og bredt udvalg*, *uligheder/popularitet*, *netværksindflydelse* viser sig at kunne påvises i de indhentede data. Den første faktor er, at der er forskellighed blandt de afspillede sange, og det ses tydeligt, da typen af afspillet musik blandt brugerne rangerer sig fra heavy-metal til elektronisk musik. Den anden faktor er, at der blandt de afspillede sange findes uligheder og at populariteten blandt sangene er forskellige. Dette går igen i vores data, da de fleste af de mest afspillede sange i vores data også er at finde øverst på Spotify's toplister i den givne periode. Den tredje og sidste faktor er, at netværket og ens sociale venner har en indflydelse på de afspillede sange. Tre af testbrugerne er venner og de sange de har afspillet stemmer overens med, at de nogenlunde har afspillet forskellige sange lige mange gange, der må derfor alt andet lige være tale om, at brugerne imellem hinanden, har haft en indflydelse på de sange, som de hver især har lyttet til.

7.7 Evaluering af 2. iteration

I denne iteration er funktionerne, der skal skabe datagrundlaget for anbefalingerne blevet implementeret, herunder oprettelsen af brugere, samt de tre kontekstuelle parametrer. Disse funktioner er alle blevet testet med succes og de mindre fejl vil blive rettet i næstkommende iteration.

De indhentede data blev analyseret og kvaliteten af de data var gode nok til, at der kunne foretages en analyse op mod den generelle form for The Long Tail. Det viste sig således, at de indhentede data havde mønster af The Long Tail.

I næste iteration vil der derfor være fokus på at danne anbefalinger til brugerne, der muligvis vil kunne udligne The Long Tail, og dermed kunne anbefale brugere nye sange, som man ikke tidligere har lyttet til.

Kapitel 8

3. Iteration

De væsentligste elementer, der skal danne grundlag for anbefalinger blev implementeret og testet i 2. iteration. Fokus i 3. iteration er derfor på design og implementering af anbefalinger samt præsentation af disse til brugeren, således at en funktionel prototype vil blive udviklet.

8.1 Kravspecifikation

Test og feedback i forrige iteration gav flere idéer til applikationen, som er blevet prioriteret forskelligt. Idéen med at kunne åbne applikationen fra notifikationsbaren er blevet prioriteret højt og er kommet med i denne iteration, hvorimod rystefunktionen og Facebook-integration er blevet nedprioriteret, grundet fokus i iterationen. Disse vil dog komme med i en fremtidig iteration.

Kravspeficikation for 3. iteration ser dermed således ud:

1. Kunne generere anbefalinger til brugeren ud fra en given kontekst.
2. Præsentere anbefalede sange til brugeren.
3. Åbne applikationen fra notifikationsbaren, når applikationen kører.
4. Starte anbefalede sange i Spotify.

Derudover vil de mindre fejl fra testen i 2. iteration i afsnit 7.5 blive rettet. Efter denne iteration skulle applikationen dermed være en funktionel prototype, der illustrerer konceptet bag applikationen med kontekstbaserede anbefalinger.

8.2 Design

I denne iteration, er der som nævnt i afsnit 8.1, fokus på anbefalingerne samt præsentationen til brugeren. Nedenstående vil derfor være en gennemgang af, hvorledes anbefalingerne vil blive genereret samt prototypens samlede systemoversigt.

8.2.1 Anbefalinger

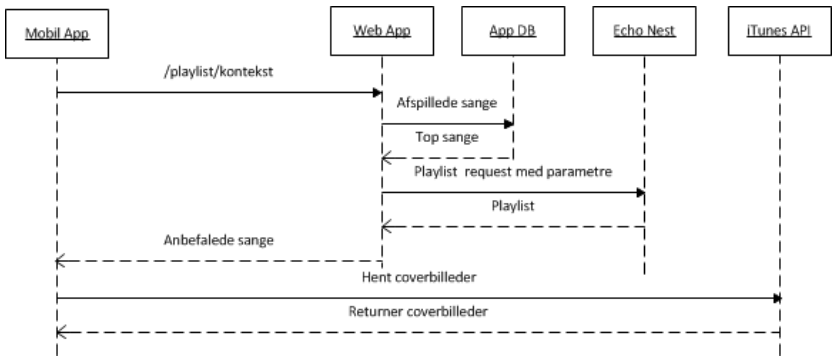
Til at danne anbefalinger, der ligner de afspillede sange ud fra nogle givne parameter, er der brug for et opslagsværk/database, hvori sangene og deres parametre kan sammenlignes direkte. Til dette findes en række API'er, heriblandt Last.fm og Echo Nest.

Last.fm's API giver adgang til en række features såsom top album og sange for en given bruger, ud fra lokation i form af land eller via et tag fx "Disco". Derudover kan der, hvis brugeren har en profil hos Last.fm, søges efter top-N naboer og deres mest afspillede sange, men kan ikke bruges i denne sammenhæng, da brugernes kontekst ikke er knyttet til disse afspilninger. Endvidere er der kun mulighed for at finde lignende artister samt sange ud fra henholdsvis artistnavn og sangtitel. Echo Nest derimod, giver mulighed for at skrue på diverse parametre, der er knyttet til sangene, hvilket giver bedre mulighed for at filtrere anbefalingerne.

Parametrene der kan skrues på er fx tempo i sangen, om sangen kan betegnes som stille, om den er god at danse til, samt hvor meget energi der er i sangen etc. Samlet set giver disse parametre rig mulighed for at generere anbefalinger til brugeren i form af en playlist, der passer til brugerens kontekst. Eksempelvis kan det tænkes, at brugeren lytter til sange med et højt tempo og en masse energi i under løb.

For at kunne danne anbefalingerne ud fra ovenstående parametre er det vigtigt at kende de samme parametre for de afspillede sange. Da vores applikation kun får information i form af artist og sangtitel kan metoden Search i Echo Nest's Track API benyttes til at få disse oplysninger, som efterfølgende kan danne grundlag for anbefalingerne. Herunder vises, hvordan et request til disse oplysninger kunne se ud:

http://developer.echonest.com/api/v4/song/search? api_key=KEY &format=json &artist=ArtistNavn &title=SangTitel &bucket=audio_summary &results=1



Figur 8.1: Sekvensdiagram der viser processen fra anbefalingsrequest til præsentation.

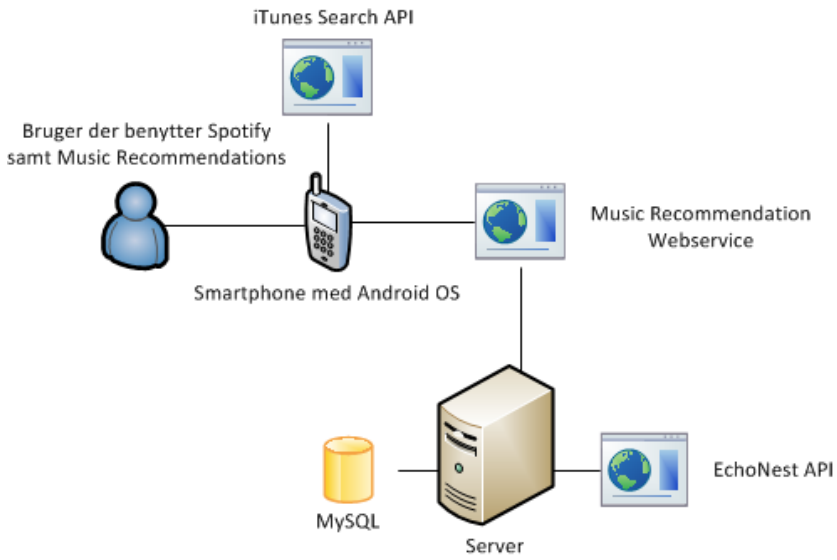
Hvor API nøglen, artistnavn og sangtitel placeres i URL'en. Derudover vælges formatet JSON samt bucket=audio_summary, der giver de ønskede sang parametre. Et response for eksempelvis Carly Rae Jepsen - Call me Maybe kan ses i bilag A.13.

Havde sangen eksempelvis været med i de 10 mest afspillede sange, og samtidig været den sang med det laveste tempo, under brugerens løbeture, kunne sange med samme tempo anbefales under fremtidige løbeture.

Der findes Million Song Dataset[15], som er et samlet datasæt over en million populære sange fra Echo Nest, hvor de fleste parameter fra Echo Nest's analyse er knyttet til sangene. Det er også en "klynge"af komplementære datasæt fra SecondHandSongs, musiXmatch, Last.fm og Taste Profile. Til hver sang tilføjet id'er til andre services fx musicbrainz, playme.com og 7digital, som tilbyder API'er til at søge efter metadata for en given sang. Datasættet kan dermed bruges som en genvej til andre services, hvor eksempelvis sangtekster og information om kunstneren kan findes.

Hele datasættet kan downloades og fylder 280GB, men pga. manglende ressourcer under udviklingen af prototypen, har det ikke været muligt at gøre tilgængeligt på webserveren. Et udsnit af datasættet kan hentes med 10.000 sange. Det har det vist sig at test-brugernes afspillede sange ikke eksisterer i dette udsnit, hvorfor det er valgt at benytte Echo Nest's API direkte.

Som ovenstående beskriver, kan der tilføjes mange parametre i et playlist request til Echo Nest. I prototypen vil der dog kun blive returneret anbefalinger ud fra top 5 artister ud fra en given kontekst, men der vil være fokus på at implementere en generisk metode, der kan benyttes til alle former for playlist requests.



Figur 8.2: Systemoversigt over hele applikationen.

Sekvensdiagrammet på figur 8.1 giver indblik i processen - fra requestet med brugerens givne kontekst bliver sendt, til den visuelle præsentation af sangene.

8.2.2 Systemoversigt

Som beskrevet i afsnit 6.6 er det vigtigt at gøre anbefalingerne spændende. iTunes Search API vil derfor blive benyttet til at indhente coverbilledet på de anbefalede sange, således at brugerens interesse bliver bevaret. Dette gøres ved at sende et request indeholdende; artist, title samt angive at der søges efter en sang. fx `http://itunes.apple.com/search?term=carly+rae+jepsen+call+me+maybe&entity=song&limit=1`, som returnere et JSON respons indeholdende link til coverbilledet for en given sang.

Sammen med Echo Nest API'et, der står for anbefalingerne, er der dermed tilføjet to API'er i forhold til sidste iteration, og system oversigten er illustreret på figur 8.2.

8.3 Implementering

Først og fremmest er de mindre fejl fra 2. iteration blevet rettet. Begge rapporterede fejl skyldtes, at HTTP requestet blev kørt i UI tråden, hvilket gjorde at applikationen kunne force close[1] ved langsomt respons. Denne forbindelse er nu oprettet i sin egen tråd, og brugeren vil blive præsenteret for en dialogboks indtil respons modtages, hvilket også forhindre applikationen i at force close samt User Interfacet i at "fryse".

Herunder er implementering beskrevet fra mobil applikationens request om anbefalinger til den visuelle præsentationen til brugeren.

8.3.1 Anbefalinger via Echo Nest

Webservicen, hvori kommunikationen mellem mobil applikationen og webapplikationen foregår, er som tidligere nævnt implementeret som en RESTful webservice 7.3.2. Mobil applikationen sender derfor et HTTP GET request indeholdende de konstekstuelle parameter, der forklarer brugerens kontekst, til webapplikationen. Requestet kunne eksempelvis se således ud; musicrecappdtu.appspot.com/service/playlists/martintest/walking/low/20:4e:7f:22:f0:52, for testbrugeren Martin. Hvor brugernavn samt de tre kontekstuelle parametre bevægelse, støjniveau samt lokation i form af et WiFi's BSSID sendes med i URL'en. Denne URL udløser metoden searchPlayListUser(), som er vist i bilag A.14.

I searchPlayListUser bliver metoden recommendSongs() kaldt, hvori brugerens top 5 artister i den givne kontekst bliver fundet via getTopArtists(). Herefter bliver de returnerede artister tilføjet til ParamList, der er et HashMap af strenge indeholdende de forskellige parametre, som ønskes sendt afsted i Echo Nest requestet. Koden for recommendSongs() ses i bilag A.15.

Metoden searchPlayList() er desuden en generisk metode, der står for alle playlist request til Echo Nest ved blot at få parameterlisten med som argument. På sigt vil der dermed hurtig kunne sendes en playlist request med parametre, såsom ønsket tempo, energy samt danceability etc. Det endelige request til Echo Nest ser eksempelvis således ud http://developer.echonest.com/api/v4/playlist/static?api_key=N6E4NIOVYMTHNDM8J&artist=Medina&format=json&results=20&type=artist-radio, hvor der hentes 20 anbefalede sange ud fra artisten Medina.

[1] Uventet lukning af applikation

Responset fra Echo Nest bliver dernæst, som i mobil applikationen, mapped ved hjælp af API'et Jackson JSON Mapper. Fra det mappede response kan en sangliste til slut returneres til mobil applikationen i form af JSON, som vist nedenfor.

```
"songs": [
    {
        "artist_id": "ARVOHFA1187B99EA07",
        "id": "SOZYVPR1374288C8B7",
        "artist_name": "Joey Moe",
        "title": "Jorden Er Giftig"
    }
]
```

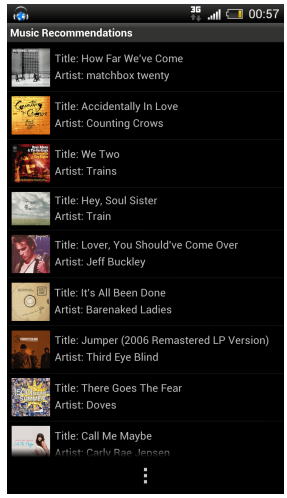
8.3.2 Præsentation af anbefalingerne

Præsentationen af anbefalingerne er implementeret, som en ListActivity. I denne ListActivity kaldes updateRecommendations() i onCreate() metoden, således at anbefalingerne bliver opdateret ud fra den givne kontekst, når brugeren starter applikationen.

I metoden updateRecommendations() oprettes først en instans af klassen SongAdapter *m_adapter*, der indeholder metoden getView, som står for opdateringen af rækkerne i listen. Herefter oprettes en tråd, hvori metoden getSongs() kaldes. Denne afsender requestet til webservicen med de givne kontekstuelle parametre og modtager respons i form af et JSON array, der bliver mapped til en liste med anbefalede sange *m_songs*. Udover de mappede informationer fra webservicen title og artist, bliver der sendt et request afsted til iTunes Search API, hvor URL til coverbilledet for hver sang bliver hentet.

Når getSongs er færdig sættes en Runnable[2] til at køre i UI tråden, hvori *m_adapter* bliver gjort opmærksom på ændringerne i datasættet af de anbefalede sange *m_songs*, således at listen bliver opdateret. Her benyttes den indhentede URL for hvert coverbilledet til at downloade dette og lægge det i cache. Eksisterer billedet dermed i forvejen, vil der ikke blive brugt ressourcer på at downloadet billedet. Koden kan ses i ListActivity'en RecommendationsActivity samt klassen ImageDownloader og figur 8.3 viser et eksempel på præsentationen af anbefalingerne. Koden for getSongs(), SongAdapter, og Runnable findes i bilag A.16, A.17 og A.18.

[2]Runnable repræsenterer typisk en kommando/noget kode der kan udføres, og bliver ofte brugt i en anden tråd.



Figur 8.3: Præsentation af anbefalinger til brugeren.

8.3.3 Afspilning af sange i Spotify

Da selve anbefalingslisten er oprettet som en ListActivity benyttes metoden onListItemClick() til at behandle klik på en given række/sang i visningen. I denne metode bliver den anbefalede sang oprettet ud fra den valgte sang i listen, som er et RecommendedSong objekt. Herefter oprettes en Intent, hvortil nogle handlinger knyttes, således at Spotify starter op, søger efter sangen og starter afspilningen. Koden er vist i bilag A.19.

8.4 Test & feedback

Efter implementeringen af anbefalingsdelen, som foretager anbefalingerne til sangene, blev næste version af prototypen sendt til testbrugerne. Brugerne skulle her teste om de fik vist og modtaget nye anbefalede sange i applikationen. Herunder er indrapporteret et udsnit af det feedback, der blev modtaget fra testbrugerne.

Testbruger 1 forklarer efter opdatering af applikationen. *"Nu har jeg lyttet til mange forskellige sange, og selvom jeg hele tiden lytter til forskellig musik, så bliver jeg altid anbefalet de samme sange igen og igen. Der er altså gengangere og ikke særligt mange nye sange."*

Testbruger 2 forklarer. *"Jeg modtager hver gang en liste med 10-15 forskellige sange, og nogle af de sange jeg får i listen er nogle jeg bestemt ikke finder særligt gode, men de går ofte igen på listen over anbefalingerne."*

Derudover var der ingen tekniske problemer og applikationen virkede dermed uden problemer på alle testede enheder.

8.5 Lancering på Google Play

Applikationen er efter 3. iterations test og feedback blevet signeret[3] og publiceret på Google Play, som en gratis beta applikation. Dette har vist sig at være en succes, hvor applikationen har haft over 200 downloads på kort tid. Hovedsageligt har det været folk fra USA og Spanien, der har downloadet applikationen med henholdsvis 33,5% og 28,5% af de samlede downloads. Holder disse tal stik fremover, kunne applikationen på sigt laves i en spansk version.

Derudover har lanceringen givet mulighed for at teste applikationen på endnu flere enheder, som vist på figur 8.4. Statistikken fra Google taler desuden for, at applikationen er relativ stabil, da der efterfølgende kun har været indsendt 3 fejlrapporter.

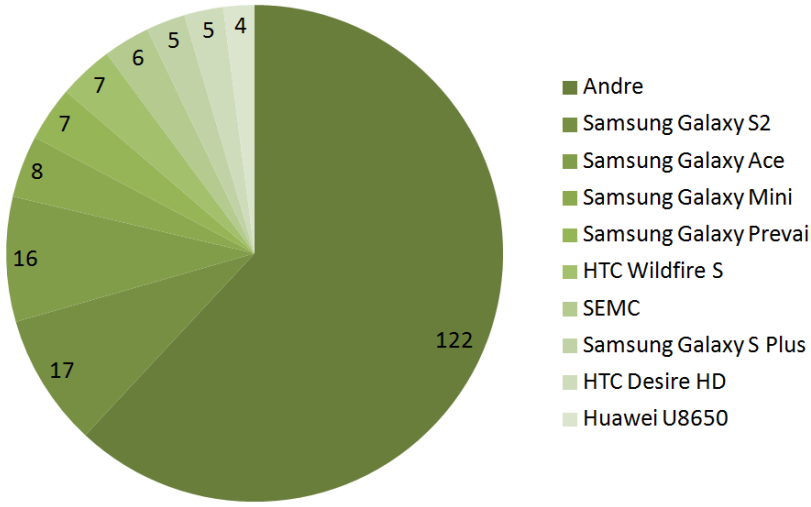
Applikationen kan ses på Google Play via dette link `http://bit.ly/MusicRec` eller ved at søge på 'Music Recommendations'. En guide til applikationen findes i afsnit 9.

8.6 Evaluering af 3. iteration

Prototypen er på nuværende tidspunkt fuldt funktionel, hvor det er muligt for brugeren at oprette sig og modtage anbefalede sange ud fra bestemmelsen af brugerens kontekst. Der er dog stadig nogle løse ender jf. afsnit 8.4, hvor der blandt andet skal findes en løsning på gengangere af sange, som brugeren får vist. Der skal yderligere findes en løsning til filtrering af de anbefalede sange fra Echo Nest, således brugeren ikke modtager anbefalinger på sange, som aldrig bliver afspillet.

Fremtidsplanerne for den videre udvikling af applikationen kan ses i afsnit 10.

[3] Enhver applikation publiceret på Google Play skal signeres med en privat nøgle af udvikleren



Figur 8.4: Antal applikation downloads pr. enhed.

Kapitel 9

Produktguide

Produktet er en prototype, der illustrerer, hvordan det er muligt at benytte brugerens kontekst til at danne bedre anbefalinger til musik.

Prototypen indhenter implicit feedback omkring brugerens kontekst i form af de konstekstuelle parametre; *bevægelse*, *støjniveau* samt *lokation*. Disse bliver tilknyttet de afspillede sange og danner grundlag for anbefalingerne brugeren modtager.

I afsnit 9.1 bliver installation af mobilapplikationen gennemgået, hvor der i afsnit 9.2 bliver gennemgået brug af applikationen.

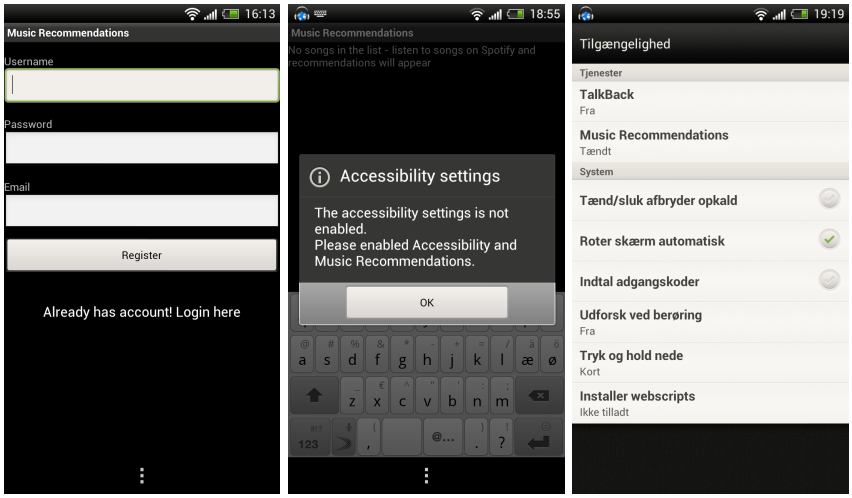
9.1 Installation

Produktet er en mobilapplikation og kan installeres på enhver smartphone med Android 2.3.3. Applikationen kan hentes og installeres i Google Play via dette link: *http://bit.ly/MusicRec*. Efter applikationen er blevet installeret, vil det være muligt at oprette en bruger, som illustreret på figur 9.1, hvor øget tilgængelig skal godkendes efter oprettelse. Efter godkendelse trykkes der blot tilbage til applikationen.

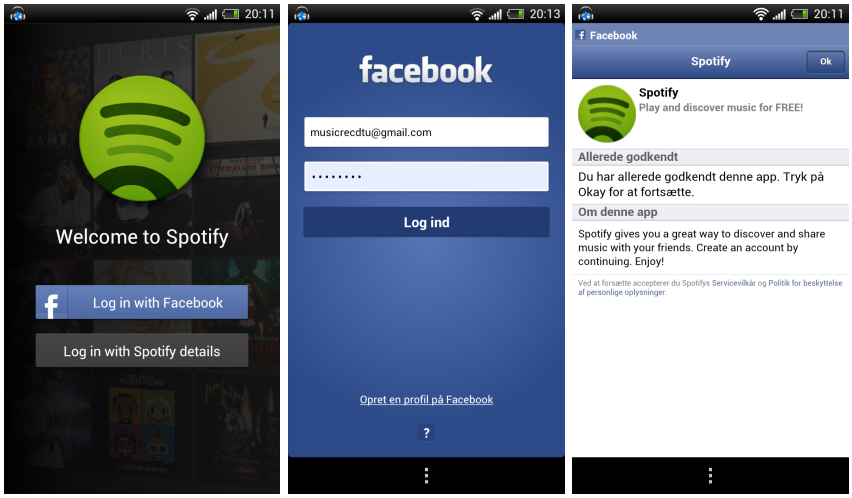
Derudover kræves en Spotify konto til afspilning af musik, hvor følgende konto er stillet til rådighed.

Brugernavn: MusicRecDTU@gmail.com

Adgangskode: nr123456



Figur 9.1: Figuren viser oprettelsen af en bruger (tv.), advarsel om at øget tilgængelighed er slået fra (midt for), samt at øget tilgængelighed er slået til (th.)



Figur 9.2: Figuren viser Spotify login (tv.), Facebook login (midt for), samt godkendelse af Facebooks anmodninger (th.)

Følg nedenstående trin for at benytte ovenstående konto. Trinene er illustreret på figur 9.2.

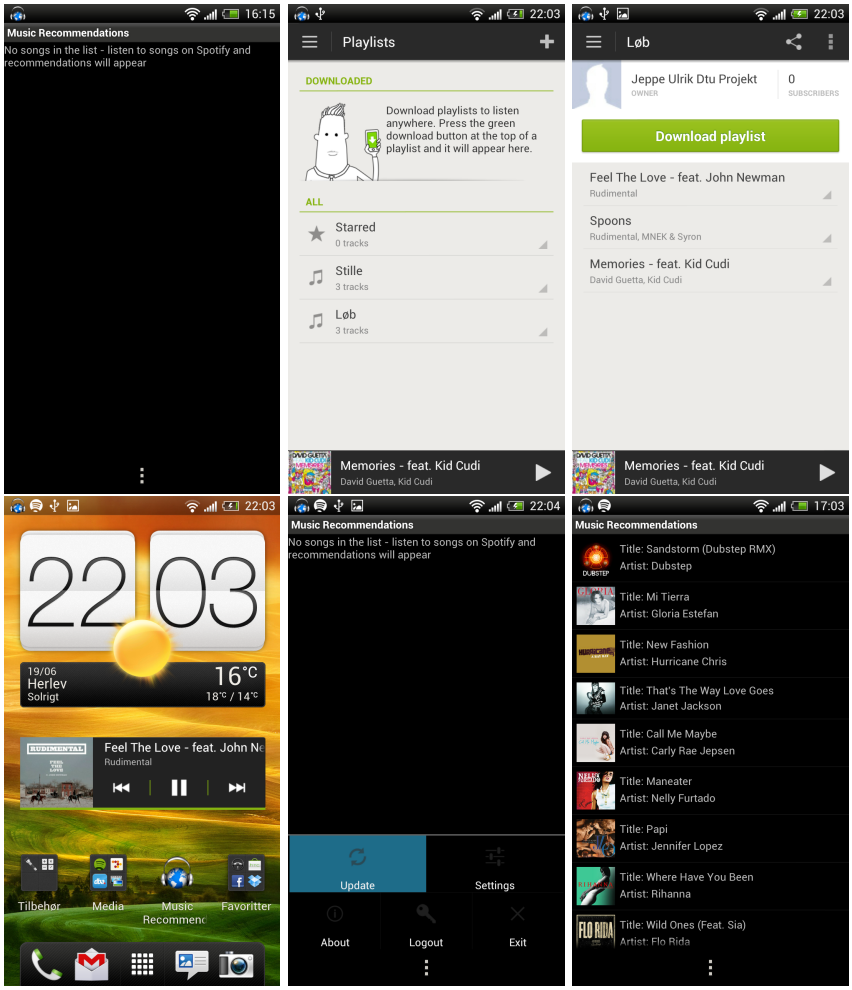
1. Installer Spotify via Google Play - *http://bit.ly/SpotifyMusic*.
2. Vælg *Log in with Facebook*
3. Indtast brugernavn (MusicRecDTU@gmail.com) og adgangskode (nr123456).
4. Accepter samtlige af Facebooks anmodninger.

Applikation er nu klar til brug og guiden i afsnit 9.2 kan benyttes, for afprøve applikationen.

9.2 Brugerguide

Følgende er en gennemgang af applikationen og beskriver, hvordan man får dannet anbefalinger via prototypen Music Recommendations. Dette er ligeledes illustreret på figur 9.3.

1. Lokaliser Music Recommendations applikationen og åbn den.
2. Log ind med brugernavn (**musicrecdtu**) og adgangskoden (**nr123456**).
3. Første gang vil listen være tom.
4. Åben Spotify og start playlisten ”Stille” eller “Løb” (alt efter behov).
5. Gå til mobilens forside og skift sange via Spotifys widget.
6. Sange skal afspilles i minimum 30 sekunder for at blive registreret.
7. Gå til Music Recommendations applikationen igen.
8. Tryk opdater i menuen og anbefalinger vil blive dannet.
9. Gentag step 4, og se hvordan de anbefalede sange i applikationen ændrer sig.



Figur 9.3: Figuren illustrerer brug af applikationen - fra applikation åbnes til anbefalinger bliver dannet

Kapitel 10

Evaluering & fremtidsplaner

Under projektet er en funktionel prototype, der udleder kontekst implicit vha. af sensorer fra brugerens smartphone, blevet udviklet. Prototypen er løbende blevet testet og evalueret af de fem deltagende testbrugere, hvilket har resulteret i en prototype, af en sådan kvalitet, at den er testet og herefter publiceret på Google Play.

Prototypen kan med stor succes knytte kontekstuelle parametre til afspillede sange og præsentere nye anbefalede sange, som er relevante for brugeren. Filtreringen af de anbefalede sange virker endnu ikke optimalt, som nævnt i evalueringen fra tredje iteration. Dette vil naturligvis være et fokusområde i de næstkommende iterationer, hvor brugerens implicitte feedback skal være med til at filtrere de anbefalede sange. Eksempelvis at databasen skal kunne gemme oplysninger om alle sange, en bruger har fået anbefalet. Vælger brugeren aldrig en given anbefalet sang over en vis periode, vil denne blive filtreret fra, således at brugeren altid bliver præsenteret for helt nye og relevante sange, samt sange brugeren tidligere har været glad for.

Næste fase i applikationsudviklingen kan være at trække data fra de sociale medier såsom Twitter og Facebook, således at brugerens kontekst, herunder social sammenhæng, kan blive endnu mere detaljeret. Derudover kan der som nævnt i afsnit 4.1 implementeres metoder til at udnytte flere kontekstuelle parametre såsom vejrforhold samt tilstedeværende ressourcer. Eksempelvis kan en beregning på synlige netværk i form af bluetooth, WiFi etc. give en indikation på, om brugeren er alene eller sammen med andre. Ved at kombinere samtlige kontekstuelle parametre, vil der eksempelvis ikke blot kunne gættes på hvad brugeren laver, men også hvem brugeren er sammen med og i hvilken sammenhæng.

Det tidligere omtalte Spotify API libsspotify i afsnit 7.3.1.4 er endvidere ved projektets afslutning blevet lanceret, hvilket giver mulighed for automatisk at

oprette playlister til afspilning samt automatisk skift af sange ud fra brugerens skiftende kontekst.

Fremtiden for applikationen er baseret på indtjening ved at sælge data og under dette projekt er vejen til datagrundlaget blevet skabt. På længere sigt vil data dermed kunne analyseres for at finde mulige mønstre og tendenser ud fra en given kontekst. Eksempelvis kan der være værdi i at vide, hvilken type musik brugeren ønsker at høre under en løbetur, således at musikken kan markedsføres på den helt rigtige måde.

Kapitel 11

Konklusion

En prototype, hvis formål er at foretage mere præcise anbefalinger på relevant musik til brugeren, er blevet udarbejdet. Prototypen er blevet publiceret på Google Play og potentielt set landet hos flere millioner af brugere verden over.

Inden udvikling af applikationen er der foretaget en markedsanalyse af eksisterende applikationer på det mobile marked. Analysen viser, at der findes andre applikationer, som kan give brugeren anbefalinger på ny musik. Applikationen er derfor blevet udviklet med udgangspunkt i markedsanalysens resultater og differentierer sig fra de eksisterende applikationer ved, at inddrage en brugers kontekst til at danne anbefalinger på relevant musik til brugeren. Der er under udvikling af prototypen arbejdet efter feature-driven developments modellerings principper, hvor én feature udvikles ad gangen og brugeren hele tiden er i fokus.

Musikanbefalinger kan foretages på baggrund af flere forskellige metoder, hvor nogle er bedre end andre i forskellige situationer. Der er i projektet udarbejdet en gennemgang af de forskellige anbefalingssystemer samt anbefalingsmetoder, og det er fundet, at den mest optimale løsning til applikationen er en kombination mellem indholdsbaseret og kontekst-baseret filtrering. Ved at kombinere disse to er det gjort muligt, at kunne bestemme hvilke sange en bruger har lyttet til i forskellige sammenhænge. Anbefalinger vil da blive udarbejdet ud fra brugerens tidligere afspillet musik i selvsamme sammenhæng, og på den måde vil anbefalingerne være mere dynamiske i forhold til anbefalingsmetoder, der ikke tager højde for en brugers kontekst.

I projektet er der arbejdet med fire kontekstuelle parametre: bevægelse, støjniveau, lokation og tid, som alle har enten direkte eller indirekte indflydelse på en brugers valg af musik og det er derfor vigtigt at tage højde for alle fire, når der skal foretages anbefalinger. Den fysisk konstekstuelle parameter bevægelse, kan med succes udledes via en smartphones accelerometer og kan sige utrolig meget om brugerens kontekstuelle sammenhæng. Fra mikrofonen kan støjniveauet som

parameter bestemmes som værende højt eller lavt og på den måde beskrive de omkringværende omgivelser. Lokation kan udledes i form af WiFi Access Points og kan både beskrive brugerens placering samt ved adgang til en brugers sociale netværk kan det bestemmes, hvem brugeren er sammen med i den givne sammenhæng.

Det digitale marked for musik tilbyder i dag utallige sange og bla. hos Spotify over 15 millioner forskellige sange. Det har aldrig før været så svært for en bruger at kunne søge og finde relevant musik i mængden. Betegnelsen The Long Tail stiller derfor nogle teknologiske udfordringer, hvor der helt konkret er behov for mekanismer, således en bruger nemmere har mulighed for, at finde relevant musik. Applikationen søger derfor at afhjælpe eller løse disse udfordringer.

Litteratur

[1] C. Anderson. The Long Tail: Why the Future of Business Is Selling Less of More, 2008. [Online; sidst set d. 13-06-2012].

[2] Android-Documentation. Android Activity Lifecycle. http://developer.android.com/reference/android/app/Activity.html, 2012. [Online; sidst set d. 11-05-2012].

[3] Android-Documentation. Application Fundamentals. http://developer.android.com/guide/topics/fundamentals.html, 2012. [Online; sidst set d. 11-05-2012].

[4] Apple. Rock og løb. http://www.apple.com/dk/ipod/nike/run.html, 2011. [Online; sidst set d. 13-05-2012].

[5] A. Butkus. Enhancing Media Personalization by Extracting Similarity Knowledge from Metadata. http://orbit.dtu.dk/services/downloadRegister/5007971/phd198_ab-final.pdf, 2010. [Online; sidst set d. 13-06-2012].

[6] Carleton-College. Memory-based algorithms. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/memorybased.html, 2012. [Online; sidst set d. 13-06-2012].

[7] Carleton-College. Model-based algorithms. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html, 2012. [Online; sidst set d. 13-06-2012].

[8] CNN. Piper Jaffray: Android app revenue is 7% of iPhone's. http://tech.fortune.cnn.com/2011/11/21/piper-jaffray-android-app-revenue-is-7-of-iphones/, 2012. [Online; sidst set d. 17-06-2012].

[9] K. G. Dailymail.co.uk. Twitter secrets for sale: Privacy row as every tweet for last two years is bought up by data firm. http://www.dailymail.co.uk/sciencetech/article-2107693/Twitter-sells-years-everyones-old-vanished-Tweets-online-marketing-companies.html, 2012. [Online; sidst set d. 11-05-2012].

[10] JacksonMapper. Documentation. http://jackson.codehaus.org/, 2012. [Online; sidst set d. 19-06-2012].

[11] Jogfm. About. http://jog.fm, 2012. [Online; sidst set d. 13-05-2012].

[12] D. A. o. S. K. Dey. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. http://www.cc.gatech.edu/fce/ctk/pubs/HCIJ16.pdfl, 2000. [Online; sidst set d. 13-06-2012].

[13] S. Kramtchenko. Feature-driven development. http://www.featuredrivendevelopment.com/files/FDD_vs_XP.pdf, 2004. [Online; sidst set d. 04-06-2012].

[14] Last.fm. Frequently Asked Questions. http://www.last.fm/help/faq, 2012. [Online; sidst set d. 13-05-2012].

[15] Million-Song-Dataset. Documentation. http://labrosa.ee.columbia.edu/millionsong/, 2011. [Online; sidst set d. 13-06-2012].

[16] Moodagent. Moodagent Backgrounder. http://www.moodagent.com/about/backgrounderl, 2012. [Online; sidst set d. 13-05-2012].

[17] H. Siles. Hybrid Content-Based Collaborative-Filtering Music Recommendations. http://essay.utwente.nl/743/1/Thesis_Document_by_Hugo_Siles.pdf, 2007. [Online; sidst set d. 11-05-2012].

[18] Slacker. About Slacker Radio. http://www.slacker.com/company/about/, 2012. [Online; sidst set d. 17-06-2012].

[19] Sounddrop. Press Center. http://soundrop.com/press, 2012. [Online; sidst set d. 13-05-2012].

[20] Spotify. Technologies. https://developer.spotify.com/technologies/, 2012. [Online; sidst set d. 17-05-2012].

[21] SpotOn-Radio. Description. http://itunes.apple.com/us/app/spoton-radio/id471762418?mt=8, 2012. [Online; sidst set d. 13-05-2012].

[22] P. F. Strand. Context Resolver. https://github.com/hrstrand/ContextResolver, 2012. [Online; sidst set d. 17-05-2012].

[23] Wikipedia. iTunes Store. http://en.wikipedia.org/wiki/ITunes_Store, 2012. [Online; sidst set d. 17-06-2012].

[24] Wikipedia. Viral Marketing. http://da.wikipedia.org/wiki/Viral_marketing, 2012. [Online; sidst set d. 17-06-2012].

A Kode eksempler

A.1 Username XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

<TextView
        android:id="@+id/text_username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:text="@string/text_username"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <EditText
        android:id="@+id/editUsername"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textVisiblePassword" >
        <requestFocus />
    </EditText>
</LinearLayout>
```

A.2 LoginActivity onCreate

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.login); // login.xml layout
}
```

A.3 Menu XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/menu_settings"
        android:icon="@drawable/icon_settings"
        android:title="@string/menu_settings_title"/>

    <item
        android:id="@+id/menu_logout"
        android:icon="@drawable/icon_logout"
        android:title="@string/menu_logout_title"/>
</menu>
```

A.4 LoginActivity onCreateOptionsMenu

```
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

A.5 WiFi scanner getBestSignal

```
private ScanResult getBestSignal(){

    WifiManager wifi;
    List<ScanResult> results;

    ScanResult signal = null;

    wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
    results = wifi.getScanResults();
    if (results != null) {
        for (ScanResult result : results){
            if (signal == null || WifiManager.compareSignalLevel(signal.level, resul
                level) < 0) {
                signal = result;
            }
        }
    }
    return signal;
}
```

A.6 SoundService inRecordMode

```
while(inRecordMode) {
    mAudioRecord.read(audioBuffer, 0, mAudioBufferSampleSize);
    totalSoundSmallSection = 0;
    for (int i=0; i < audioBuffer.length;i++){
        totalSoundSmallSection = totalSoundSmallSection + Math.abs(audioBuffer[i]);
    }

    avgSoundSmallSection = totalSoundSmallSection/audioBuffer.length;
    largeSection.add(avgSoundSmallSection);

    if (System.currentTimeMillis() - sampleStart > 30*1000) {
        totalSoundLargeSection = 0;
        for (int i=0; i<largeSection.size()-1; i++){
            totalSoundLargeSection = totalSoundLargeSection + largeSection.get(i);
        }

        String soundGuess;
        long avgSoundLargeSection = totalSoundLargeSection / largeSection.size();

```

```
20          if (avgSoundLargeSection > 200){
21              soundGuess = "HIGH";
22          }
23          else {
24              soundGuess = "LOW";
25          }
26
27          storeGuess(soundGuess);
28          sampleStart = System.currentTimeMillis();
29          largeSection.clear();
30      }
31  }
```

A.7 Bevægelsesdetektor handleSmallSection

```
1  private void handleNewSmallSection() throws IOException {
2      SmallSectionResolver ssr = SmallSectionResolver.createSectionResolver(rawValues)
3      MovementGuess latestGuess = ssr.calculateMovement();
4      Log.i(LOGID, "movementguess is " + latestGuess);
5      rawIndex = 0;
6
7      long now = System.currentTimeMillis();
8      SmallSection smallSection = new SmallSection(lastSmallSectionStartTime, now,
           latestGuess, new LocationWrapper(locationManager.getLastKnownLocation(
           LocationManager.NETWORK_PROVIDER)));
9      lastSmallSectionStartTime = now + 1;
10
11     float smallSpeed = currentSession==null?0.0f:currentSession.calcAvgSpeed(30, 60
           System.currentTimeMillis());
12     MovementStateManager.get().handleData(smallSection.getGuess(), smallSpeed);
13
14     storeGuess(latestGuess, MovementStateManager.get().getSummarizedGuess());
15 }
```

A.8 SpotifyGrabber - setServiceInfo

```
1  @Override
2  private void setServiceInfo() {
3      AccessibilityServiceInfo info = new AccessibilityServiceInfo();
4
5      // We are interested in all types of accessibility events.
6      info.eventTypes = AccessibilityEvent.TYPES_ALL_MASK;
7
8      // We want to provide specific type of feedback.
9      info.feedbackType = AccessibilityServiceInfo.FEEDBACK_AUDIBLE |
           AccessibilityServiceInfo.FEEDBACK_GENERIC |
10             AccessibilityServiceInfo.FEEDBACK_HAPTIC |
11             AccessibilityServiceInfo.FEEDBACK_SPOKEN |
12             AccessibilityServiceInfo.FEEDBACK_VISUAL;
13
14     // We want to receive events in a certain interval.
15     info.notificationTimeout = EVENT_NOTIFICATION_TIMEOUT_MILLIS;
```

```

    // We want to receive accessibility events only from certain packages (Spotify)
    info.packageNames = PACKAGE_NAMES;

    setServiceInfo(info);
}
```

A.9 SpotifyGrabber - onAccessibilityEvent

```
@Override
public void onAccessibilityEvent(AccessibilityEvent event) {
    if (!event.getText().isEmpty()){
        if (event.getText().toString().contains("-")){
            playedSongs.add(event.getText().toString());
        }

        currentTimeDiff = System.currentTimeMillis() - currentTime;
        if (currentTimeDiff > 30000){
            if(event.getText().toString().contains("-")){
                new Thread(new Runnable() {
                    public void run() {
                        String songTitle = "";
                        String songArtist = "";
                        if (playedSongs.size()>1) {
                            songTitle = getSongName(playedSongs.get(playedSongs.size
                                ));
                            songArtist = getArtistName(playedSongs.get(playedSongs.s
                                -2));
                            }

                        String wifi1 = "00:00:00:00:00:00";
                        List<ScanResult> bestWifis = getBestSignal();
                        ScanResult signal1 = bestWifis.get(0);
                        if (bestWifis.get(0) != null){
                            wifi1 = signal1.BSSID;
                        }

                        dbsource.open();
                        ContentValues values = new ContentValues();
                        values.put(DatabaseSQLiteHelper.COLUMN_WIFI_1, wifi1);
                        dbsource.updateInfoRow(1, values);
                        Cursor c = dbsource.getRow(1);
                        dbsource.close();

                        String motion = c.getString(c.getColumnIndex(DatabaseSQLiteH
                            .COLUMN_MOTION));
                        String sound = c.getString(c.getColumnIndex(DatabaseSQLiteHe
                            COLUMN_SOUND));
                        String username = c.getString(c.getColumnIndex(
                            DatabaseSQLiteHelper.COLUMN_USERNAME));
                        String password = c.getString(c.getColumnIndex(
                            DatabaseSQLiteHelper.COLUMN_PASSWORD));

                        // Send song to webservice
```

```
                        PlayedSong playedSong = new PlayedSong(username, password,
                            songTitle, songArtist, motion, sound, wifi1, "");
                        playedSong = WebserviceManager.sendJacksonPostRequest(Playe
                            class, "/song", playedSong);
                        }
                }).start();
            }
        }
        currentTime = System.currentTimeMillis();
    }
}
```

A.10 Webapplikation - songController

```
@RequestMapping(method=RequestMethod.POST, value="/song")
public @ResponseBody PlayedSongForm addSong(@RequestBody PlayedSongForm sf) {
    boolean succes;
    succes = inputHandler.SongInput(sf.getUsername(), sf.getPassword(), sf.getTitle
        .getArtist(), sf.getMotion(), sf.getSound(), sf.getWifi());

    PlayedSongForm playedsongform = null;
    if (succes) {
        playedsongform = new PlayedSongForm(sf.getUsername(), "", sf.getTitle(), sf
            getArtist(), sf.getMotion(), sf.getSound(), sf.getWifi(), "0006"); // pl
            song has been added
    }
    else {
        playedsongform = new PlayedSongForm(sf.getUsername(), "", sf.getTitle(), sf
            getArtist(), sf.getMotion(), sf.getSound(), sf.getWifi(), "0002"); // wr
            password
    }
    return playedsongform;
}
```

A.11 Webapplikation - songInputHandler

```
@Override
public boolean SongInput(String username, String password, String title, String arti
    String motion, String sound, String wifi) {

    if (userService.userExist(username)){
        if (userService.checkPassword(username, password)) {
            playedSongService.addPlayedSong(username, title, artist, motion, sound,
                ;
            return true;
        }
        return false;
    }
    return false;
}
```

A.12 Webapplikation - addPlayedSong

```
@Override
@Transactional(propagation = Propagation.REQUIRED)
public void addPlayedSong(String username, String title, String artist, String motio
    String sound, String wifi) {

    String songId = songDao.existSong(title, artist);
    String userId = userDao.findIdByUsername(username);
    String playedsongId = playedSongDao.existPlayedSong(userId, songId, motion, sour
        wifi);
    if (songId == null) {
        Song song = new Song(title, artist);
        songDao.insert(song);
        songId = song.getId();
    }

    if (playedsongId == null){
        PlayedSong playedsong = new PlayedSong(songId, userId, motion, sound, wifi)
        playedSongDao.insert(playedsong);
    }
    else {
        playedSongDao.increasePlayedByOne(playedsongId);
    }
}
```

A.13 JSON Echo Nest respons

```
"songs": [
    {
        "audio_summary": {
        "key": 7,
        "mode": 1,
        "time_signature": 4,
        "duration": 193.50313,
        "loudness": -7.018,
        "energy": 0.37346185644448354,
        "tempo": 119.934,
        "audio_md5": "84fd52175a4999149263e4f50cefc79e",
        "analysis_url": "https://echonest-analysis.s3.amazonaws.com/TR/
            upl9Jgy8KzOKcYRKJ6wQwy4vbEwbCGwk_X4Eg7/3/full.json?Signature=4
            gDYeKoYVkFQV8BpKmG7VMn370Q%3D&Expires=1338717813&AWSAccessKeyId=
            AKIAJRDFEY23UEVW42BQ",
        "danceability": 0.7811083177610919
    },
        "artist_id": "ARRWGYU12086C17800",
        "id": "SOXLHDK1372FC963BC",
        "artist_name": "Carly Rae Jepsen",
        "title": "Call Me Maybe"
    }
]
```

A.14 Webapplikation - searchPlayListUser

```
@RequestMapping(method=RequestMethod.GET, value="/playlists/{username}/{motion}/{sou
    wifi}", headers="Accept=application/json, application/xml")
public @ResponseBody SongList searchPlayListUser(@PathVariable String username,
    @PathVariable String motion, @PathVariable String sound, @PathVariable String wi

    ParamList paramList = new ParamList();
    paramList = recommendService.recommendSongs(username, motion, sound, wifi);

    SongList songList = new SongList();
    if (echoNestService.searchPlayList(paramList) != null) {
        songList = new SongList(echoNestService.searchPlayList(paramList).getSongs()
    }
    return songList;
}
```

A.15 Webapplikation - recommendedSongs

```
@Override
@Transactional(propagation = Propagation.REQUIRED)
public ParamList recommendSongs(String username, String motion, String sound, String
    ) {
    User user = userService.getUser(username);
    List<Artist> artists = new ArrayList<Artist>();
    ParamList paramList = new ParamList();

    // Get the 5 most played artist for user
    artists = playedSongService.getTopArtists(user.getId(), motion, sound, wifi, 5)
    if (artists != null) {
        for (Artist a : artists) {
            paramList.addParam("artist", a.getName().trim());
        }
    }
    paramList.addParam("results", "20");
    paramList.addParam("type", "artist-radio");
    return paramList;
}
```

A.16 Præsentation af anbefalinger: getSongs

```
private void getSongs(){
    try{
        m_songs = new ArrayList<RecommendedSong>();

        dbsource.open();
        Cursor c = dbsource.getRow(1);
        dbsource.close();

        String username = c.getString(c.getColumnIndex(DatabaseSQLiteHelper.
            COLUMN_USERNAME));
```

```
10          String motion = c.getString(c.getColumnIndex(DatabaseSQLiteHelper.COLUMN_MO
                ;
11          String sound = c.getString(c.getColumnIndex(DatabaseSQLiteHelper.COLUMN_SOU
12          String wifi1 = c.getString(c.getColumnIndex(DatabaseSQLiteHelper.COLUMN_WIF
13
14          RecommendedSongList recResponse = WebserviceManager.sendJacksonGetRequest(
                RecommendedSongList.class, "http://musicrecappdtu.appspot.com/service/
                playlists/"+username+"/"+motion+"/"+sound+"/"+wifi1);
15
16          if (recResponse.getSongs() != null) {
17              for (RecommendedSong rs : recResponse.getSongs()) {
18                  RecommendedSong recSong = new RecommendedSong();
19                  recSong.setTitle(rs.getTitle());
20                  recSong.setArtist(rs.getArtist());
21                  recSong.setCoverArtURL(getCoverUrl(rs.getArtist(), rs.getTitle()));
22                  m_songs.add(recSong);
23              }
24          }
25
26      } catch (Exception e) {
27          Log.e("Error in getSongs", e.getMessage());
28      }
29      runOnUiThread(returnRes);
30  }
```

A.17 Præsentation af anbefalinger: SongAdapter

```
1  private class SongAdapter extends ArrayAdapter<RecommendedSong> {
2
3      private ArrayList<RecommendedSong> items;
4
5      public SongAdapter(Context context, int textViewResourceId, ArrayList<Recommende
           > items) {
6          super(context, textViewResourceId, items);
7          this.items = items;
8      }
9      public View getView(int position, View convertView, ViewGroup parent) {
10         View v = convertView;
11         if (v == null) {
12             LayoutInflater vi = (LayoutInflater)getSystemService(Context.
                   LAYOUT_INFLATER_SERVICE);
13             v = vi.inflate(R.layout.row, null);
14         }
15
16         RecommendedSong recSong = items.get(position);
17         if (recSong != null) {
18
19             TextView tt = (TextView) v.findViewById(R.id.toptext);
20             TextView bt = (TextView) v.findViewById(R.id.bottomtext);
21             ImageView img = (ImageView) v.findViewById(R.id.icon);
22             img.setImageResource(R.drawable.music_icon);
23
24             if (tt != null) {
25                 tt.setText("Title: "+recSong.getTitle());
```

```
            }
            if(bt != null){
                bt.setText("Artist: "+ recSong.getArtist());
            }

            if (recSong.getCoverArtURL() != null){
                ImageDownloader imageDownloader = new ImageDownloader();
                imageDownloader.download(recSong.getCoverArtURL(), img);
            }
        }
        return v;
    }
}
```

A.18 Præsentation af anbefalinger: Runnable

```
private Runnable returnRes = new Runnable() {

    public void run() {
        if(m_songs != null && m_songs.size() > 0){
            m_adapter.notifyDataSetChanged();
            for(int i=0;i<m_songs.size();i++)
                m_adapter.add(m_songs.get(i));
            }
        m_ProgressDialog.dismiss();
        m_adapter.notifyDataSetChanged();
    }
};
```

A.19 Afspilning i Spotify

```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    try {
        super.onListItemClick(l, v, position, id);
        RecommendedSong recSong = (RecommendedSong)l.getItemAtPosition(position);

        // Shows text for selected song
        String strTextToDisplay = "Selected song is : " + recSong.getTitle() + " - "
            recSong.getArtist();
        Toast.makeText(this, strTextToDisplay, Toast.LENGTH_LONG).show();

        // Starts the song in Spotify
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.setAction(MediaStore.INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH);
        intent.setComponent(new ComponentName("com.spotify.mobile.android.ui", "com
            spotify.mobile.android.ui.Launcher"));
        intent.putExtra(SearchManager.QUERY, recSong.getArtist() + " " + recSong.ge
            ());
        startActivity(intent);
    }
    catch(Exception e) {
```

```
19            Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
20        }
21    }
```

B Indhentet data

B.1 Afspillede sange under testperioden

Artist	Titel	Afspilninger
Train	Drive By	21
L.O.C.	Noget Dumt	18
Carly Rae Jepsen	Call Me Maybe	18
Ida	I Can Be	13
Clemens	Ingen Kender Dagen	13
Michel TelÃ³	Ai Se Eu Te Pego	13
Original - Svenstrup & Vendelboe	Glemmer Dig Aldrig (feat. Nadia Malm)	12
Gotye	Somebody That I Used To Know	12
Lukas Graham	Ordinary Things	12
Lil Wayne	Mirror	11
feat. Sia - Flo Rida	Wild Ones	11
David Guetta	Titanium (feat. Sia)	10
Lukas Graham	Drunk In The Morning	10
Lukas Graham	Criminal Mind	10
Niklas	Veninder	10
The Animals	The House of the Rising Sun	10
Niklas	Top Swag	9
Radio Edit - Kato	Never Let U Go (feat. Snoop Dogg & Brandon Beal)	8
KESI	Ku Godt	8
Pitbull	International Love Featuring Chris Brown	8
Rasmus Hedegaard Remix - Lukas Graham	Ordinary Things	7
feat. Bruno Mars - Snoop Dogg & Wiz Khalifa	Young, Wild & Free	6
Usher	Dirty Dancer featuring Enrique Iglesias	6
Kelly Clarkson	Stronger (What Doesn't Kill You)	6
Lukas Graham	Don't Hurt Me This Way	6
Malk De Koijn	Nalk	5
Joan Baez	House of the Rising Sun	5
Ã˜stkyst Hustlers	Jazzys Gambit	5
Equinox - Skrillex	First Of The Year	5
Kato	Never Let U Go (feat. Snoop Dogg & Brandon Beal)	5
Line	Efter Dig	5
Malk De Koijn	Pige, Girl	5
Donkeyboy	City Boy	5
Puls	DOPE ft. Ole Henriksen	4
Chris Brown	Turn Up The Music	4
Malk De Koijn	Klap Din Hoddok (Hund Bider Mand)	4
Live in Copenhagen - Agnes Obel	Close Watch	4
The Animals	House of the Rising Sun	4
Stanley Most	Kom kom	4
A-Live (The Bongo Song) - Safri Duo	Played	4
feat. Emerge [Original Version] - Kongsted	Mardi Gras	4
The Tremeloes	Silence Is Golden	3
feat. Nicki Minaj - David Guetta	Turn Me On	3
Iron Maiden	Fear Of The Dark	3
LMFAO	Sexy And I Know It	3
Malk De Koijn	Vi Tager Fuglen PÃ¥ Dig	3
The Everly Brothers	Cathy's Clown	3
David Guetta	Crank It Up (feat. Akon)	3
Jessie J	Domino	3
Yepha	Det GÃ¥r Ned (feat. Niklas)	3
Original - Kongsted & Emerge	You Got That (Right Energy) (feat. S.O.A.P.)	3
Mix) (feat. A'typisk, Gilli, Ataf, Mass Ebdrup, Kesi, Sivas & Clemens) - Kato	Fuck Hvor Er Det Fedt (At VÃ¦re Hip Hop'er) (Part 2 G	3
Club Edit - Kongsted	The Spirit	3
Lukas Graham	Before The Morning Sun	2
Gene Chandler	Duke of Earl	2
Radio Edit - Avicii	Levels	2
The Foundations	Build Me Up Buttercup	2
Sebastian	Fuld Af Nattens Stjerner	2
Paul Revere & The Raiders	Kicks	2
Original Instrumental Mix - Gouryella	Ligaya [Mix Cut]	2
Sebastian	Blod & guld	2
Taio Cruz	Hangover	2
Rihanna	We Found Love	2
Iron Maiden	Run To The Hills	2
The Searchers	Love Potion Number 9	2
Skrillex	First Of The Year (Equinox)	2
'60s Rock Heroes	Surfinâ€™ USA (as made famous by The Beach Boys)	2
Joey Moe	Banger Til Min Banger	2
Elevator Music	Wonderful Tonight	2
Iron Maiden	The Wicker Man	2
Malk De Koijn	5 Ã˜res Ting	2
David Guetta	Turn Me On (feat. Nicki Minaj)	2
Medina	Synd For Dig	2
Rasmus Seebach	MillionÃ¦r (feat. Ankerstjerne)	2
Ben E. King	Stand By Me	2
Bonus Track - Oh Land	Speak Out Now	2
Paul Oakenfold Full On Fluoro Remix - Planet Perfecto Knights	ResuRection [Mix Cut]	2
The McCaffrey Folk Singers	Black Velvet Band	2
feat. Niklas - Yepha	Det GÃ¥r Ned	2
Agnes Obel	Philharmonics	2
Original Mix - Abstract Vision	Kinetic [Mix Cut]	2
ASOT 550 Anthem [Mix Cut] - Original Mix - W&W	Invasion	2
Piano Sessions - Agnes Obel	Avenue	2
Sebastian	Fuld af nattens stjerner (Piratpladen)	2
Bill Haley	Rock Around the Clock	2
Elevator Music	Forever And Ever	2

Christina Milian	Dip It Low	2
Explicit Album Version - Gucci Mane	What It's Gonna Be	2
Live in Copenhagen - Agnes Obel	Smoke & Mirrors	2
Playa Circle	Duffle Bag Boy	2
Aura Dione	Friends	2
Medina	12 Dage	2
Malk De Koijn	BAB Melkon	2
Sharon Shannon	blackbird	2
LEN "CHIP" HAWKES (BASS GUITAR)	Even The Bad Times Are Good	2
David Guetta	I Just Wanna F. (feat. Timbaland & Dev)	2
Kenny Rogers	Just Dropped In (To See What Condition)	2
Dash Berlin's Sense Of Touch Remix - Lange	Touched [Mix Cut]	2
Sebastian	Admiral Benbow (velkomst)	2
Justin Bieber	Boyfriend	2
Safri Duo	Athena	1
Elevator Music Radio	Bach Air on the G String	1
Elevator Music	Love Story Theme	1
Sebastian	Hispaniola	1
Ashanti	Foolish	1
Elevator Music Radio	God Rest You Merry, Gentleman Classical Light Music	1
Original Mix - Orjan Nilsen	Amsterdam [Mix Cut]	1
Tony Burrows	United We Stand	1
Elevator Music	Just The Way You Are	1
Original Mix - A-Lusion	Call & Answer	1
Korn	Wicked	1
Radio Edit - Safri Duo	Rise (Leave me alone)	1
Setrise & Johann Stone	Icesave	1
Original Mix - Luke Bond	Amaze [Mix Cut]	1
Maurice Williams	Stay	1
Sebastian	DrÃ¸mmen	1
Original Mix - Gaia	4 Elements [Mix Cut]	1
Elevator Music	I'll Follow You There	1
Sebastian	PapegÃ¸jesangen (Mastodonterne)	1
Woe, Is Me	Hot 'N Cold	1
Korn	Freak On A Leash	1
Far East Movement	Live My Life	1
Korn	Lost	1
Mitch Ryder	Devil With A Blue Dress On	1
Korn	Mr. Rogers	1
Elevator Music Radio	Jesu, Joy of Man's Desiring	1
Eric Burdon (The Animals)	House Of The Rising Sun	1
Radio Edit - Tacabro	TacatÃ	1
Sebastian	Hyldest til England	1
Sebastian	Hispaniola (Piratpladen)	1
Radio Edit - Michael Rune	Min Indre Stemme (feat. Nadia Gattas)	1
Agnes Obel	Louretta	1
Piano Sessions - Agnes Obel	On powdererd Ground	1
Agnes Obel	Brother Sparrow	1
Sebastian	Naboerne (Piratpladen)	1
DJ Party	Run It	1
Original Mix - John O'Callaghan	Broken [Mix Cut]	1
Deep Blue Something	Breakfast At Tiffany's	1
The Be Good Tanyas	House Of The Rising Sun	1
Agnes Obel	On Powdered Ground	1
The McCaffrey Folk Singers	The Hills Of Connnemara	1
Suspekt	Klaus Pagh	1
Lukas Graham	Red Wine	1
Aqua	Good Morning Sunshine	1
Lukas Graham	Moving Alone	1
The Tokens	The Lion Sleeps Tonight	1
The Keep	House of the Rising Sun (originally by The Animals)	1
Iron Maiden	2 Minutes To Midnight	1
Coldplay	Paradise	1
Elevator Music	We've Only Just Begun	1
Hungarian Dances - Elevator Music Radio	Brahms	1
Iron Maiden	Aces High	1
Bob Docker And His Orchestra	A Summer Place (Theme)	1
Dance Party DJ	Jailhouse Rock (DJ Remix)	1
Sebastian	Naboerne	1
The Miracles	Mickey's Monkey	1
Jay Z	Ni**as In Paris	1
Lukas Graham	Never Let Me Down	1
Elevator Music	How Can I Tell You	1
Korn	K@#*%!	1
Elevator Music Radio	Grieg Solveig's Song from Peer Gynt Suite n2	1
Elevator Music Radio	Radio Enigma	1
Rasmus Thude;Young	Til MÃ¥nen & Tilbage Part 2 (Rasmus Thude & Young)	1
The Grass Roots	Let's Live For Today	1
Nocturne n.3 - Elevator Music Radio	Liszt	1
Elevator Music Radio	New Day	1
Agnes Obel	Just So	1
Yo Gotti	Blow Your Ass Off	1
Romantic Strings & Orchestra	Nadia's Theme (Theme to "The Young And The Restless")	1
Iron Maiden	Hallowed Be Thy Name	1
Armin van Buuren's Intro Edit - Omnia	The Fusion [Mix Cut]	1
1st part - Elevator Music Radio	Albinoni Adagio	1
No Doubt	Don't Speak	1

Eine kleine Nachtmusik (A Little Light Music) - Elevator Music Radio	Mozart	1
Elevator Music Radio	Cafe Lounge	1
Original Mix - Audien	Keep This Memory [Mix Cut]	1
Tim Hardin	House Of The Rising Sun	1
Original Mix - VillaNaranjos	Granadella [Mix Cut]	1
Suspekt	Weekend Kriger	1
Love Dreams n.3 - Elevator Music Radio	Liszt	1
Elevator Music Radio	Peace Within	1
Agnes Obel	Falling, Catching	1
Original Mix - Alphaverb	Realization Of A Dream	1
Elevator Music Radio	Cala Jondal	1
Sonata No. 16 C major (Sonata facile) , KV 545 (1788) 1 allegro - Elevator I	Mozart	1
Iron Maiden	Satellite 15	1
OMC	How Bizarre	1
Klauss Goulart & Mark Sixma's Deep Universe Remix - Kid Alien	The Atmosphere	1
Roger Shah Naughty Love Mix - Sunlounger	Try To Be Love [Mix Cut]	1
Elevator Music Radio	Blue Marlin	1
David Guetta	Little Bad Girl (feat. Taio Cruz & Ludacris)	1
Nocturnes op 27 2 lights music - Elevator Music Radio	Chopin	1
Animals	House of the Rising Sun	1
Agnes Obel	Avenue	1
Rihanna	Skin	1
Sean Tyas Remix - Dash Berlin	Waiting [Mix Cut]	1
Eximinds Remix - Alexander Popov	When The Sun [Mix Cut]	1
Iron Maiden	The Trooper	1
Headhunterz & Noisecontrollers Radio Edit - Brooklyn Bounce	Club Bizarre	1
feat. Sia - David Guetta	Titanium	1
The Drifters	Under the Boardwalk	1
Original Mix - Active Sight	Out Of Our Lives [Mix Cut]	1
1910 Fruitgum Company	Simon Says	1
Iron Maiden	The Number Of The Beast	1
Scope DJ Remix - Dr. Rude	Midnight	1
The Beatles	Press Conference In America, August 1965 (Part 1)	1
Iron Maiden	Transylvania	1
BeatbangerZ	Close To You (Cc.K Remix Edit)	1
Piano Sessions - Agnes Obel	Just So	1
Iron Maiden	Blood Brothers	1
Shaka Loveless	Tomgang	1
Sebastian	Find ham, find ham	1
Joey Dee & The Starliters	Peppermint Twist	1
YOMC Club Mix - Dj Eremit	Tanz der Seele [Mix Cut]	1
Abstract Vision & Elite Electronic	Kinetic	1
Medina	Lyser I MÃ¸rke	1
Aqua	Freaky Friday	1
Piano Sessions - Agnes Obel	Riverside	1
The Miracles	Tears Of A Clown	1
Orjan Nilsen Midsummernite Remix - Tiddey	Keep Waiting	1
Shogun Remix - Dash Berlin	Better Half Of Me [Mix Cut]	1
The McCaffrey Folk Singers	Irish Rover	1
Live - Nina Simone	House of the Rising Sun	1
Elevator Music Radio	Open Minded	1
Original Mix - Alexander Popov	Attractive Force [Mix Cut]	1
Romantic Strings & Orchestra	What The World Needs Now Is Love	1
Sebastian	BrÃ¸dre skal vi dele	1
Sandi Thom	The House Of The Rising Sun	1
The Shangri-Las	Leader Of The Pack	1
Walls of Jericho	House Of The Rising Sun	1
Original Radio Mix - Brisby	The House Of The Rising Sun	1
Explicit Album Version - Gucci Mane	Trap Talk	1
Elevator Music Radio	Balearic Peace and Serenity	1
Santo & Johnny	Sleep Walk	1
Sebastian	Tak fordi I kom	1
Evanescence	My Heart Is Broken	1
Nocturne 54 op 4 - Elevator Music Radio	Grieg	1
Sebastian	SÃ¸rÃ¸versangen	1
Malk De Koijn	Weekend Kriger	1
Juicy J, Nicki, Gucci Mane	Girl After Girl feat. Nicki, Gucci Mane	1
Elevator Music	If	1
Elevator Music Radio	Beethoven Ode to Joy	1
Remaster - The Doors	L.A. Woman	1
Elevator Music	I Found Your Diary Underneath A Tree	1
Joey Moe	Dobbeltslag	1
Original Mix - Nash & Pepper	Ushuaia Memories [Mix Cut]	1
Gymnope?die No.2 - Elevator Music Radio	Erik Satie	1
Kaare Norge	House Of The Rising Sun	1
Safri Duo	Apollo	1
Gymnopedie n.1 - Elevator Music Radio	Erik Satie	1
Frijid Pink	House Of The Rising Sun	1
Avicii	Sunshine	1
Sam & Dave	Soul Man	1
Original Mix - The Blizzard	Piercing The Fog [Mix Cut]	1
Sebastian	Ostesangen	1
Elevator Music	My Love	1
Live in Copenhagen - Agnes Obel	Philharmonics	1
Original 1968 Recording - The Tremeloes	I Shall Be Released	1
Ave Verum - Elevator Music Radio	Mozart	1
feat. Skrillex - Korn	Get Up!	1

Elevator Music Radio	Amazing Grace	1
Dash Berlin Remix - Filo & Peri	This Night [Mix Cut]	1
The Third Bardo	Five Years Ahead Of My Time	1
7" Original Radio - Robin Cook	I Won't Let The Sun Go Down On Me	1
Nocturne - Elevator Music Radio	Chopin	1
The Mersey Beats	Wishin' And Hopin'	1
Explicit Album Version - Gucci Mane	Gucci Speaks	1
Elevator Music Radio	Ibiza Sun	1
Jorn van Deynhoven Remix - Ram	RAMsterdam [Mix Cut]	1
Original Mix - Andy Ling	Fixation [Mix Cut]	1
Mat Zo	Superman	1
Dolly Parton	The House Of The Rising Sun	1
Agnes Obel	Wallflower	1
Elevator Music	Aubrey Was Her Name	1
Requiem - Suspekt	Elektra	1
Canon in D Classical Background Music - Elevator Music Radio	Pachelbel	1
Radio Edit - Peter Andre	Mysterious Girl	1
Original Mix - Armin Van Buuren	Stellar [Armin van Buuren presents Gaia Mix Cut]	1
The Animals	Dont Let Me Be Misunderstood	1
Romantic Strings & Orchestra	Music Box Dancer	1
Aurosonic Progressive Mix - Headstrong	Helpless 2011	1
The Troggs	Wild Thing	1
The Alan Parsons Project	Mammagamma	1
Kyau & Albert Remix - Above & Beyond	You Got To Go	1
Scenes from Childhood Opus 15 - almost too serious - Elevator Music Radi	Schumann	1
Album for the young Opus 68 - The Happy Farmer - Elevator Music Radio	Schumann	1
Kim Larsen & Kjukken	This Is My Life (Live 2006)	1
Tritonal Feat. Fisher	Slave (Tritonal & Ben Gold Club Dub)	1
The Association	Windy	1
Herman's Hermits	I'm Into Something Good	1
Denis Leary	Traditional Irish Folk Song	1
Suspekt	Elektra	1
Explicit Version - Nicki Minaj	Starships	1
The Musicmakers	Silence Is Golden (Originally Released by The Tremeloes)	1
Svenstrup & Vendelboe - "Glemmer dig aldrig" feat. Nadia Malm	Official Teaser :labelmade: 2012	1
Bto	House Of the Rising Sun	1
Live - Nina Simone	House Of the Rising Sun (Live)	1
Original Mix - Nicky Romero	Toulouse	1
Elevator Music Radio	Islands of Chill	1
Solid Stone Remix - X&Y	New Beginnings	1
Gary Barlow	Forever Love	1
Radio Edit - Arty	Kate	1
Original Mix Edit - Orjan Nilsen	Amsterdam	1
Iron Maiden	Afraid To Shoot Strangers	1
State Of Mind	Make You Cry	1
Blake Jarrell Remix - Armin Van Buuren	Youtopia	1
Lukas Graham	Nice Guy	1
Korn	Good God	1
Protoculture Remix - James Dymond	Overthrow [Mix Cut]	1
Sebastian	Det Sorte Tegn	1
Mundy	The Galway Girl	1
Album for the young Opus 68 - knecht reprecht - Elevator Music Radio	Schumann	1
Iron Maiden	Phantom Of The Opera	1
Original Mix - Bioweapon	All About Music	1
Debussy - Elevator Music Radio	Clair de Lune	1
James Last	The House Of The Rising Sun	1
The Searchers	Sweets For My Sweet	1
Korn	No Place To Hide	1
Sean Paul	She Doesn't Mind	1
(Tribute to The Tremeloes) - Studio Allstars	Silence Is Golden	1
Elevator Music Radio	Sunrise	1
Explicit Album Version - Gucci Mane	Weirdo	1
Original Mix - DJ Ezteq	Forget & Smile	1
Sonate 01 Opus 2 with Gentle Bird Sounds for Meditation Relaxation - Ele	Beethoven	1
feat. Rocko & Webbie - Gucci Mane	I Don't Love Her	1
Original Mix - Coast 2 Coast	Home [Mix Cut]	1
Arietta - Elevator Music Radio	Grieg	1
Vanity Fare	Hitchin' A Ride	1
Suspekt	Nyt Pas	1
L.O.C.	Momentet feat. UŞO	1
Eve	U, Me & She	1
Apocalyptica	Worlds Collide	1
Elevator Music Radio	Moonlight Sonata Mvt 1	1
Jennifer Lopez	Dance Again Feat. Pitbull	1
Elevator Music Radio	Auld Lang Syne	1
Joey Moe	Livet Er Et Dansegulv	1
Elevator Music	Unchained Melody	1
Iron Maiden	Running Free	1
Original Mix - Ã˜rjan Nilsen	Between The Rays [Mix Cut]	1
Iron Maiden	Wasted Years	1
The Musicmakers	The House of the Rising Sun (Originally Released by The Animals)	1
Armin van Buuren presents Gaia [Mix Cut] - Original Mix - Armin Van Buu	J'ai Envie De Toi	1
Live in Copenhagen - Agnes Obel	Sons & Daughters	1
Evanescence	The Change	1
Sebastian	Den flyvende hollÃ¦nder	1
Korn	A.D.I.D.A.S.	1
Nina Simone	The House of the Rising Sun	1

featuring Don Omar - Lucenzo	Danza Kuduro feat. Don Omar	1
Live in Copenhagen - Agnes Obel	Katie Cruel	1
Dance DJ & Company	Loca People	1
Original Mix - Abstract Vision	Blossom [Mix Cut]	1
Ã¨stkyst Hustlers	Hustlerstil	1
Medina	Kl. 10	1
Trini Lopez	If I Had A Hammer	1
Michael Jackson	History	1
Puls	Hvis Du GÃ¥r	1
Sharon Shannon	the galway girl	1
Piano Sessions - Agnes Obel	Brother Sparrow	1
Elevator Music	Just You & I	1
Elevator Music Radio	Blue Reflection	1
Elevator Music Radio	Playa D'en Bossa	1
Original Mix - Coone	The Undefeated	1
Suspekt	Vi Ses I helvede	1
Sebastian	Dumt & dyrt at takke nej	1
Raaban Dirty Dutch Edit - Baby Alice	Heaven Is a Dancefloor	1
American Idol Performance - Haley Reinhart	House Of The Rising Sun	1
MC5	Kick Out The Jams (Original Uncensored Version 1968)	1
Korn	Porno Creep	1
Elevator Music Radio	Muzak and Elevator Music	1
In the Hall of the Mountain King - Elevator Music Radio	Grieg	1
En-Motion Remix - The Thrillseekers	Synaesthesia [Mix Cut]	1
Evanescence	The Other Side	1
Extended Club Version - Michael McDonald	Sweet Freedom	1
The Tremeloes	Hello World	1
Ã¨stkyst Hustlers	Han FÃ¥r For Lidt	1
Elevator Music Radio	Dreaming of Pangea	1
Malk De Koijn	En Gang	1
Eric Burdon	House Of The Rising Sun	1
Elevator Music Radio	Turquoize Haize	1
Agnes Obel	Beast	1
Book 1 - Fugue 14 with Relaxing Nature Sounds - Elevator Music Radio	Johann Sebastian Bach	1
Live in Copenhagen - Agnes Obel	Louretta	1
Beethoven Music - Elevator Music Radio	Fur Elise	1
Book 1 - Fugue 02 - Elevator Music Radio	Johann Sebastian Bach	1
Medina	For Altid	1
Korn	Twist	1
One Direction	What Makes You Beautiful	1
Piano Sonata 09 opus 14 - Elevator Music Radio	Beethoven	1
Sanne Salomonsen	Den Lille LÃ¸gn	1
Original Mix - Arty	Around The World	1
Edit - The Pitcher	Sky Rocket	1
Radio Edit - Antique	Opa Opa	1
Ave Maria - Elevator Music Radio	Gounod	1
The McCaffrey Folk Singers	The Pub With No Beer	1
Iron Maiden	Can I Play With Madness	1
Safri Duo	Baya Baya	1
Radio Edit - EaggerStunn	Kugledans	1
Original Mix - Sean Tyas	Lift [Mix Cut]	1
The Drifters	Save The Last Dance for Me	1
Suspekt	Far Gir En Is	1
Elevator Music Radio	Enya Breeze	1
Iron Maiden	Prowler	1
Van Morrison	Brown Eyed Girl	1
Hangover ft. Flo Rida -	Taio Cruz	1
Korn	Word Up!	1
Sebastian	De Engelske Dyder	1
Suspekt	Helt Alene	1
Elevator Music Radio	The Joy of Living	1
David Guetta	Titanium ft. Sia	1
Michael Jackson	Little Susie / Pie Jesu	1
Dash Berlin Remix - First State	Reverie [Mix Cut]	1
Agnes Obel	Close Watch	1
The Dubliners	The Rocky Road To Dublin	1
Agnes Obel	Over the Hill	1
Korn	Swallow	1
Book 1 - Fugue 10 - Elevator Music Radio	Johann Sebastian Bach	1
The Sonics	Cinderella	1
Theodis Ealey	House Of The Rising Sun	1
Original Mix - Airbase	Escape [Mix Cut]	1
Hungarian Rhapsodies 1885 n.12 - Elevator Music Radio	Liszt	1
Agnes Obel	Riverside	1
Prelude n.9 Opus 28 - Elevator Music Radio	Chopin	1
Korn	Lowrider	1
Korn	Chi	1
Elevator Music	My Heart Will Go On (Love Theme From Titanic)	1
Suspekt	The Valley	1
Ciara	Promise	1
Original Mix - Fix To Fax	Meridian [Mix Cut]	1
Elevator Music Radio	Sunset Light Music	1
Elevator Music Radio	Natural Health	1
Murphy & The Mob	Born Loser	1
Sonata Pathetique Classical Music Background - Elevator Music Radio	Beethoven	1
Joey Moe	Du' En Ã‰ner	1
Kreisleriana Opus 16 1838 - Elevator Music Radio	Robert Schumann	1

Sebastian	SÃ¸rÃ¸versangen (Piratpladen)	1
Suspekt	Ruller Tungt	1
Hank Ballard	The Twist	1
Shadows Of The Night	Gloria	1
Elevator Music Radio	Brahms Lullaby	1
Armin van Buuren Remix - Wiegel Meirmans Snitker	Nova Zembla [Mix Cut]	1
Noah	Alt Er Forbi	1
Elevator Music Radio	Carol of the Bells	1
Original Mix - Jaren	Man On The Run [Mix Cut]	1
feat. Skrillex & Kill The Noise - Korn	Narcissistic Cannibal	1
Elevator Music Radio	Intuition	1
I alt		**815**