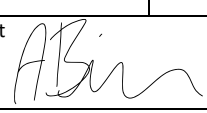



Forår 2017

Forprojekt til Bachelorprojekt

Projekttitel:	CrowdPlay		
Projektnr.:	<i>Udbyder</i>	<i>Vejleder</i>	<i>Bivejleder</i>
-	Anders W. Birkelund Rune D. Rask Jonas R. Hartogsohn	Thomas Nielsen	Kasper L. Jensen
Deltager 1	Studienr. 201404118	Navn Anders W. Birkelund	Underskrift 
Deltager 2	Studienr. 201271001	Navn Rune D. Rask	Underskrift
Deltager 3	Studienr. 201405166	Navn Jonas R. Hartogsohn	Underskrift 

Evaluering af forprojekt Vejleder afleverer kopi af denne side SENEST 26. juni 2017 til studiekontoret!		
Evalueringsdato	Sæt kryds	Sæt kryds
Vejleders underskrift	Godkendt	Ikke godkendt

Indhold

Opgavebeskrivelse.....	3
Kravspecifikation (udkast)	4
MoSCoW	4
Must Have	4
Should Have.....	4
Could Have	4
Won't Have.....	4
Tilsvarende projekter	6
Projektforløb	6
Scrum.....	6
Arbejdstimer.....	6
Arbejdsplads	6
Projektplan (udkast)	7

Opgavebeskrivelse

Forestil dig, at du holder en fest for alle dine venner. Alle hygger sig, stemningen er fantastisk og søde toner fylder rummet fra dit lækre B&O-anlæg. Du kan dog ikke slappe helt af! Hvorfor?

Fordi hver eneste gang en sang er forbi, vil en eller anden vælge den næste og sætter sin telefon til dit anlæg. Dine ører bløder idet det forgyldte mini-jack-stik sættes i den ene smartphone efter den anden – uden at volumen bliver skruet bare en smule ned. Ikke alene er støjen forfærdelig, men at der bliver skiftet sang konstant er temmelig frustrerende for alle gæsterne.

- Det er her CrowdPlay kommer ind i billedet!

CrowdPlay er en løsning, der lader alle deltagende gæster ved en fest være med til at bestemme den musik, der bliver spillet. En smartphone, tablet eller computer tilkobles et musikanlæg, og herfra vil der blive spillet den højst ønskede sang blandt gæsterne. I løbet af festen kan gæsterne up-vote sange, de synes godt om. Jo flere up-votes en sang får, desto højere kommer den op ad play-listen og vil derved blive spillet tidligere end de øvrige sange.

For hvert up-vote en sang får, modtager den gæst, der tilføjede sangen ét point. Dette skaber et konkurrenceelement der giver brugerne et ekstra incitament til at bruge applikationen. Samtidig vil det gennem festens levetid være muligt at følge med i, hvilken gæst der er bedst til at tilføje sange, der rammer bredest hos publikum.

Det ønskes i dette bachelorprojekt at udvikle en system der vha. en teknologisk løsning kan afhjælpe dette problem.

Det ønskes derfor at udvikle:

- App der kan afhjælpe ovenstående problemstilling fra brugerens egen smartphone (Android og iOS).
- Systemet udvikles så skalérbart som muligt, så der fra start tages højde for, at flere tusinde brugere skal kunne anvende systemet samtidigt.
- Mulighed for persistering af bruger-data og øjeblikkelig synkronisering mellem brugere på tværs af platforme.

Udover udviklingen af selve produktet, forventes det, at en større del af projektet vil være dedikeret til kommunikation med brugere af applikationen. Funktionalitet og brugervenlighed ønskes derved testet i løbet af udviklingsprocessen for at opnå den bedst mulige løsning til slutproduktet.

Kravspecifikation (udkast)

Der er opstillet følgende krav i form af en MoSCoW-analyse. Heri defineres de forventninger og krav, som projekt-udbyder har stillet – rangeret fra hvad projektet *skal* indeholde som minimum til, hvad der i første omgang udelades fra projektet, men kunne tilføjes i en videre udvikling.

MoSCoW

MoSCoW-metoden benyttes til prioritering af features inden for et projekt. Herved bliver der enighed omkring vigtigheden af hver feature for systemet.

Must Have

- DJ skal kunne oprette en ny fest
- DJ skal kunne genoprette en gammel fest
- DJ skal kunne tilføje sange til playlisten
- Gæst kan logge ind på applikationen via Facebook
- Gæst skal kunne tilmelde sig festen
- Gæst skal kunne tilføje sange til festens playliste
- Gæst skal kunne stemme på tilføjede sange
- Gæst får "point" når andre gæster stemmer på ens sange
- Gæst skal kunne se en rangliste over gæster
- Gæst skal have mulighed for at se alle omkringliggende fester i google maps
- Gamle fester skal gemmes i database
- Applikation til Android smartphones
- Applikation til iOS(iPhone)

Should Have

- DJ skal kunne tilføje en playliste fra streamingtjeneste
- Gæst skal kunne se detaljeret oversigt over hvilke sange de andre gæster har tilføjet
- Gæst skal kunne starte en afstemning om at fjerne nuværende sang fra playliste
- Benyttelse af udviklingsværktøj til indsamling af brugeraktivitet.
- Performance/load tests

Could Have

- Gæst skal kunne gemme festens playliste på sin egen streamingtjeneste
- Gæst skal kunne tilføje sange fra egne playlister fra streamingtjeneste
- Gæst får en notifikation når førstepladsen på ranglisten overtages.
- Applikation til Chromecast
- Applikation til Apple-TV
- Applikation til Web (Browser)
- Integration med og anvendelse af mere end én streamingtjeneste(Spotify, Tidal, Google Music mv.)
- DJ skal kunne invitere sine Facebook venner

Won't Have

- Micropayments i applikationen

Note om realisering og arkitektur

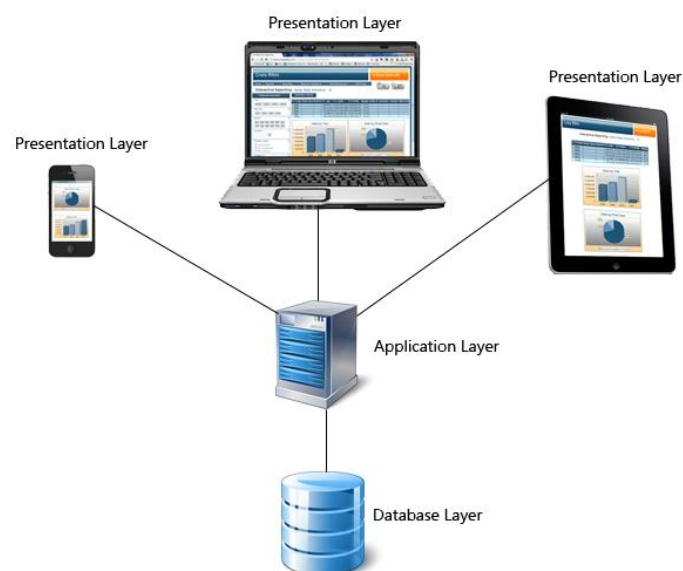
I MoSCoW analysen er nævnt en række features som ønskes opfyldt i den færdige applikation.

Applikationen ønskes som minimum udviklet til Android og iOS, med størst mulig grundlag for udvidelser til flere platforme. Derfor er det blevet diskuteret hvorvidt systemarkitekturen skal være af typen 3-Tier, med Client-, Server- og Databaselag.

I en sådan realisering vil meget af business logic kunne "genbruges" fordi den ligger på samme fysiske tier, og ikke (nødvendigvis) ændres i tilfælde af udvidelse til flere platforme. Samtidig er det en fordel at business logic kan ændres uden en decideret opdatering af client side software.

En anden fordel ved denne arkitektur er en meget agil skalerbarhed i forhold til belastning af systemet. Servere og databaser kan tilføjes og fjernes efter behovet fra clienterne, og servere / databaser kan placeres på geografisk taktiske områder for bedst mulig brugeroplevelse.

Dette er til forskel fra den nuværende realisering af appen der anvender en 2-Tier arkitektur; Database- og presentation/business logic-Tier.



Figur 1 3-Tier arkitektur (eksempel) vist skematisk. 3-Tier arkitektur lagdeler ansvarsområder på 3 fysisk adskilte lag og sikrer dermed høj skalerbarhed i forhold til forskellige platforme og håndtering af belastning.

Tilsvarende projekter

- Jook
 - App til iOS, der giver gæster mulighed for at stemme på musik der bliver tilføjet en spilleliste. Benytter Spotify API og SDK.¹
 -
- Jukelt
 - Dansk StartUp, der har udviklet en app til spillesteder, barer mm., hvor brugere kan betale for at få spillet deres sangønske.²
- JukeStar
 - App til Android, iOS og Web, der giver gæster mulighed for at stemme på musik der bliver tilføjet en spilleliste. Benytter Spotify API og SDK.³

Projektforløb

Det er aftalt i projektgruppen at arbejde ud fra den agile udviklingsmetode Scrum. Dette understøtter ønsket om et featuredriven arbejdsforløb, hvor der fokuseres på én feature pr. Scrum Sprint. Årsagen til valget af disse to metoder er at det giver en god fornemmelse for arbejdsprocessen, da hele projektet skæres i mindre bider, og kan opstilles i en tidsplan.

Scrum

Hver morgen mødes gruppen til et kort Scrum-møde på 10-15 minutter. På Scrum-mødet vil gårsdagens arbejde blive præsenteret, samt hvilke opgaver på Scrum-boardet hver gruppemedlem har planlagt at arbejde på den pågældende dag.

Et Scrum-sprint har en varighed på 5 dage, hvor hvert sprint indeholder udvalgte dele af projektet, såsom features, litteratursøgning og grundlæggende softwareudvikling.

Inden hvert sprint vil der blive holdt et sprint-møde, hvor opgaver fra backloggen vil blive estimeret og fyldt i det kommende sprint. På den måde vil der blive lagt et overordnet mål for den kommende arbejdsuge, men stadig bibeholde den agile arbejdstilgang til projektet.

Arbejdstimer

Mandag-torsdag fra 9:00-16:00

Arbejdsplads

Der er ansøgt om et bachelor-kontor hos Ingeniør Højskolen Aarhus.

¹ <https://itunes.apple.com/th/app/joox-music-live-now/id914406371?mt=8>

² <http://www.dr.dk/nyheder/penge/jukeit-drengene-fra-loevens-hule-sigter-efter-last-christmas-og-spotify>

³ <https://jukestar.mobi/>

Projektplan (udkast)

Gruppens arbejde vil som ovennævnt planlægges og gennemføres ved brug af Scrum. Der vil tidligt i processen opstilles en overordnet tidsplan, som vil tage udgangspunkt i kravene fra MoSCoW-analysen. Her vil en estimering af tiden for *Must*-kravene bliver opstillet for at tage afsæt i disse. Når det konkrete arbejdsforløb går i gang vil det derved være muligt at estimere et nærmere tidsforbrug ved hvert enkelt sprint samt planlægge, hvorvidt *Should*- og *Could*-kravene vil blive implementeret.

Juli-August	September				Oktober				November				December
	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	Sprint	
Kravspecifikation Arkitektur Generel struktur	M1	M2	M3	M4	M5	M6							Test Rapport

Figur 2: SCRUM sprint skabelon. Sprints dedikeret til planlægning lægges i Juli-August. Softwareudvikling efter MoSCoW ligger fra September-November. Systemtest og Rapportskrivning ligger i december.

Under hele udviklingsforløbet, vil der blive holdt pilottests/workshops med mulige brugere, for på den måde at få løbende feedback. Dette giver mulighed for at tilrette design, funktionalitet og brugervenlighed inden der fortsættes til næste iteration. Nærmere planlægning med deadlines vil blive fastlagt i første del af projektforsløbet.

Inden udviklingsforløbet er det planlagt at tale med etablerede software virksomheder, som beskæftiger sig med udvikling af smartphone applikationer. På den måde kan der samles inspiration fra deres erfaringer om arbejdsprocesser og produktudvikling.

Vejledermøder

Det forventes som udgangspunkt at have et ugentligt møde med vejleder. Ved hvert møde tidssættes næste møde. Gruppemedlemmerne skiftes til at tage referater på møderne, og noter gemmes på Git.

Deling af filer og dokumenter

Gruppen vil fra start oprette et Git repository med al gruppens samlede arbejder. Her vil gruppens softwareprojekter, noter, dokumenter mm. Gemmes.

Til dokumentation, noter og rapport vil Latex anvendes. Dette integreres med Git.

Deadlines

1. 1/9 - Kravspecifikation og software arkitektur
2. 1/12 - Applikation til alle platforme
3. 31/12 - Rapport

Projektstyring

Projektet er bygget op omkring en flad ledelsesstruktur, med en demokratisk tilgang når vigtige beslutninger skal træffes. Dette giver grobund for konstruktive diskussioner, hvilket fremmer læring og forståelse for emnerne.

Ansvarsområder vil eventuelt blive uddelegeret, men først når software arkitekturen er på plads.

Prototype

En tidlig prototype er allerede blevet udviklet som eksamensprojekt i kurset SMAP. Denne prototype er lavet til Android OS, og har de fleste af vores nedskrevne *Must*-krav. Prototypen har fungeret som 'proof of concept', hvor det har givet mulighed for at danne et indtryk af potentielle brugers feedback. På trods af at prototypen fungerer efter hensigt, med mange af de ønskede features, vil der blive startet forfra på Android versionen. Årsagen til dette er, at der vil blive gået mere i dybden med software-arkitektur og -design, for at sikre bedre mulighed for vedligeholdelse og videreudvikling. Andre ting, som skalerbarhed, blev heller ikke vurderet særlig grundigt, inden prototypen blev udviklet. Dette kan betyde at opbygningen af applikationen kommer til at se helt anderledes ud i forhold til backend og datamodeller. Dermed vil der næsten ikke være noget af den samme kode at finde, fra prototypen, i den nye version der skal udvikles.