

Bachelor

Bachelor

*Bachelor in Software Development,
IT-University of Copenhagen*

Jakob Melnyk, jmel@itu.dk
Frederik Lysgaard, frly@itu.dk

May 22st, 2012

Indholdsfortegnelse

1	Baggrund for projektet	2
2	Design af brugergrænsefladen	6
3	Usability testing	12

1 Baggrund for projektet

1.1 Udfordringen ved et bookingsystem

I lang tid har lokale booking på ITU været et uoverskueligt og problemfyldt område, der har forhindret at ITU's lokaler kunne bruges optimalt. Vi vurderer derfor, at det er vigtigt at udvikle et ordentligt lokale bookingsystem således, at ITU kan udnytte deres mange lokaler optimalt. Det udfordrende ved et bookingsystem er, at det er et servicesystem der det både skal understøtte servicer til booking af lokaler samt opdatering og vedligeholdelse af systemet. Der er derfor mange aktører og mange arbejdsmråder der skal tages hensyn til. For at have overblik over, hvad der skulle understøttes og hvad der var vigtigst fulgte vi en kravspecifikation som blev udarbejdet i kurset ”Anskaffelse og kravspecifikation” på ITU i efteråret 2012. Kravspecifikationen er lavet med fokus specifikt på ITU og hvad et bookingsystem til ITU kræver.

1.2 Kravspecifikationen og ændringer

I kravspecifikationen beskrives hvordan den nuværende booking situation er på ITU, i øjeblikket foregår bookingen af lokaler i mange forskellige systemer, hvor der er en aktør som fungerer som bindeledd for administrationen af lokaler. For at forbedre den løsning foreslår kravspecifikationen, at der laves et system som alle aktørerne arbejder op imod. Det er så vidt muligt den løsning vi vil lave en begrænset implementering af. Figur ?? viser den nuværende situation.

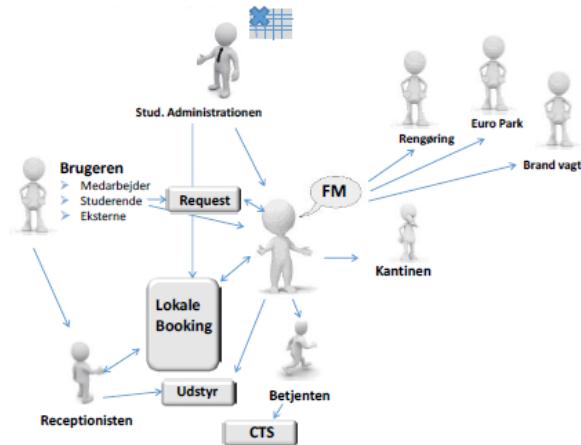


Figure 1.1: Nuværende system flow for booking systemet på ITU

1.2.1 Ændringer til kravspecifikationen

Kravspecifikationen præsenterer en datamodel, der beskriver hvilke informationer systemet skal bruge. Vi har lavet nogle ændringer så den passer til vores implementering af bookingsystemet. Vi har fjernet lokale status og lavet en ”mange til én” forbindelse mellem booking og lokale da vi gerne ville have, at en booking kun kunne have et lokale. Vi fjernede også ”Lokale egenskaber” da vi vurderede, at det gav mest mening hvis ”Inventar” repræsenterede et unikt stykke inventar og ikke havde antal i ”Lokale egenskaber”. Vi slettede også alle ”pris” værdier fra modellen, da vi ikke havde nogen intention om at understøtte fakturering.

1.3. ARBEJDSOPGAVER

1.3 Arbejdsopgaver

Denne sektion beskriver hvilke arbejdsopgaver vi understøtter i første release samt giver en prioritering af, hvilke arbejdsopgaver der bør fokuseres på i følgende releases.

1.3.1 Arbejdsområde 1: Booking

C1: Administrerer booking

Arbejdesopgave	
C1.1	Understøttet
C1.1a	Understøttet
C1.2	Understøttet
C1.2a	Understøttet
C1.3	Understøttet
C1.4	Understøttet
C1.5	Understøttet

C2: Forespørg på booking

Arbejdesopgave	
C2.1	Understøttet
C2.1a	Understøttet
C2.2	Understøttet
C2.3	Understøttet

1.3.2 Arbejdsområde 2: Booking behandling

C3: Modtag forespørgsel fra Eksern

Arbejdesopgave	
C3.1	Understøttet
C3.2	Understøttet
C3.2a	Understøttet
C3.3	Ikke understøttet
C3.4	Understøttet
C3.5	Understøttet
C3.6	Ikke understøttet

C4: Klargør lokaler

Arbejdesopgave	
C4.1	Ikke understøttet
C4.2	Ikke understøttet
C4.3	Ikke understøttet

C5: Udlever nøgle

Arbejdesopgave	
C5.1	Ikke understøttet
C5.2	Ikke understøttet

1.3. ARBEJDSOPGAVER

1.3.3 Arbejdsområde 3: Forplejning

C6: Håndter forplejning

Arbejdesopgave	
C6.1	Ikke understøttet
C6.2	Ikke understøttet
C6.3	Ikke understøttet
C6.4	Ikke understøttet

C7: Fakturer forplejning

Arbejdesopgave	
C7.1	Ikke understøttet
C7.5	Ikke understøttet

C8: Opadter menukort

Arbejdesopgave	
C8.1	Ikke understøttet

1.3.4 Arbejdsområde 3: Systemadministration

C9: Opdater listen for ekstra udstyr

Arbejdesopgave	
C9.1	Understøttet

C10: Opdater lokaler

Arbejdesopgave	
C10.1	Understøttet
C10.2	Understøttet

C11: Administrerer bruger

Arbejdesopgave	
C11.1	Ikke understøttet
C11.1a	Ikke understøttet

C12: Håndter statistik

Arbejdesopgave	
C12.1	Ikke understøttet
C12.2	Ikke understøttet

1.3. ARBEJDSOPGAVER

1.3.5 Arbejdsområde 4: Finans

C13: Behandel faktura

Arbejdesopgave	
C13.1	Ikke understøttet
C13.2	Ikke understøttet

1.3.6 Prioritering af arbejdsopgaver

Til den første release af systemet valgte vi at fokusere på at understøtte systemadministration og booking af lokaler og udstyr, hvilket tydeligt kan ses i afsnit?? da alle arbejdsopgaver i arbejdsområderne C1,C2,C3,C9 og C10 er understøttet. I kravspecifikationen er de arbejdsområder også blandt dem som er vægtet højest og det er samtidig de arbejdsområder som indeholder kernefunktionerne i systemet. Det som ikke blev prioritert højt nok til at komme med i første release var de arbejdsopgaver som fokuserede på integrationen med kantinen, arbejdsopgaverne C6.1- C6.4 er eksempler på sådanne opgaver.

Grunden til at vi ikke tog de arbejdsopgaver med var, at vi vurderede, at de ikke var nødvendige for at have et system der understøttede basis funktioner, derudover lagde arbejdsopgaverne op til, at der skulle implementeres et interface specifikt til kantinen hvilket vi vurderede ville tage lang tid at implementere og vi ville også have tre brugergrupper at tage hensyn til ift. usability og testing. Fokus for næste release vil derfor være at få udarbejdet et interface som kantinen kan bruge til at integrere med resten af systemet og få finpudset de allerede eksisterende funktioner i kravspecifikationen. Behandling af faktura har også en høj vægtning, så det vil der også blive lagt fokus på.

2 Design af brugergrænsefladen

Dette kapitel beskriver det generelle design af brugergrænsefladen samt de beslutninger, som ligger bag.

2.1 Generelle mål

Vi har valgt at designe vores brugergrænseflade ud fra reglerne om design af virtuelle vinduer[SL, s. 169] samt Ease Of Use principperne[SL, s. 9]. I forbindelse med dette valg har vi sat følgende mål for designet:

- Strømliniet brugergrænseflade
- Få forskellige skærmbilleder
- Overblik
- Effektivt

2.1.1 Strømliniet brugergrænseflade

Vi har valgt at designe skærmbillederne med samme grundstruktur. Denne lighed bør gøre det intuitivt at gå fra et skærmbillede til et andet i forbindelse med udførelse af opgaver. Desuden følger det designregel 1¹ om få vindueskabeloner.

2.1.2 Kort vej fra en opgave til en anden

Brugergrænsefladen skal gøre det hurtigt og nemt for brugeren at komme fra en opgave til en anden. Dette skal gøres ved at have få skærmbilleder involvereret i en enkelt task (designregel 2²).

2.1.3 Overblik

Brugeren skal have mulighed for nemt at danne sig overblik over bookinger, udstyr og forplejning (regel 6³). Derfor skal vi have separate skærmbilleder, som giver overblik over hver type.

2.1.4 Effektivt

Det skal være effektivt at udføre opgaver for brugere, som anvender systemet ofte.

2.2 Brugergrænsefladens udvikling og udseende

Vores skærmbilleder er opdelt i tre typer: Gitter, Almindelig og Pop-ups.

Gitterskærmbillederne bruger vi til booking af lokale, forplejning og udstyr samt administration af udstyr og lokaleinventar.

Almindelige vinduer anvender vi, hvis man skal ændre noget på et stykke udstyr/inventar eller et lokale. Pop-up skærmbilleder er generelt advarsler eller fejlbeskeder.

¹Few window templates

²Few window instances per task.

³Necessary overview of data

2.2. BRUGERGRÆNSEFLADENS UDVIKLING OG UDSEENDE

2.2.1 Gitterskærmbilledet

Gitterskærmbillederne er de skærmbilleder som står for at vise, hvad der kan bookes i vores system: lokaler, udstyr og forplejning. Figur 2.1, side 7 og Figur 2.2, side 7 er screenshots af vores første udkaste til gitterbilledet og vores endelige design.

Figure 2.1: Den endelige udgave af gitteret



Figure 2.2: Første udgave af gitter layoutet

Vores første mockup af gitterskærmbilledet havde et gitter, hvor hver række var et lokale og tiderne var kolonner. Brugerne skulle klikke i et felt for at vise, at man ønskede at booke på et bestemt tidspunkt. Når man havde valgt de tider og lokaler, man gerne ville booke, skulle man trykke på en "Book" knap. Som man kan se på Figur 2.1 valgte vi at beholde den struktur, men vi tilføjede checkboxes til gitteret. Dette blev gjort efter første runde af usability tests hvor vi observerede, at brugeren var i tvivl om, hvor man skulle klikke for at vælge tider for en booking, derudover tilføjede vi også, at udstyr m.m., man har booket, ligger

2.2. BRUGERGRÆNSEFLADENS UDVIKLING OG UDSEENDE

øverst i gitteret, når man får oversigten over fx udstyr. Dette gør det nemt at finde det element, man har booket/bestilt.

2.2.2 Almindelige skærbilleder

Disse typer af skærbilleder er primært til administration af udstyr/inventar/lokale og lignende.

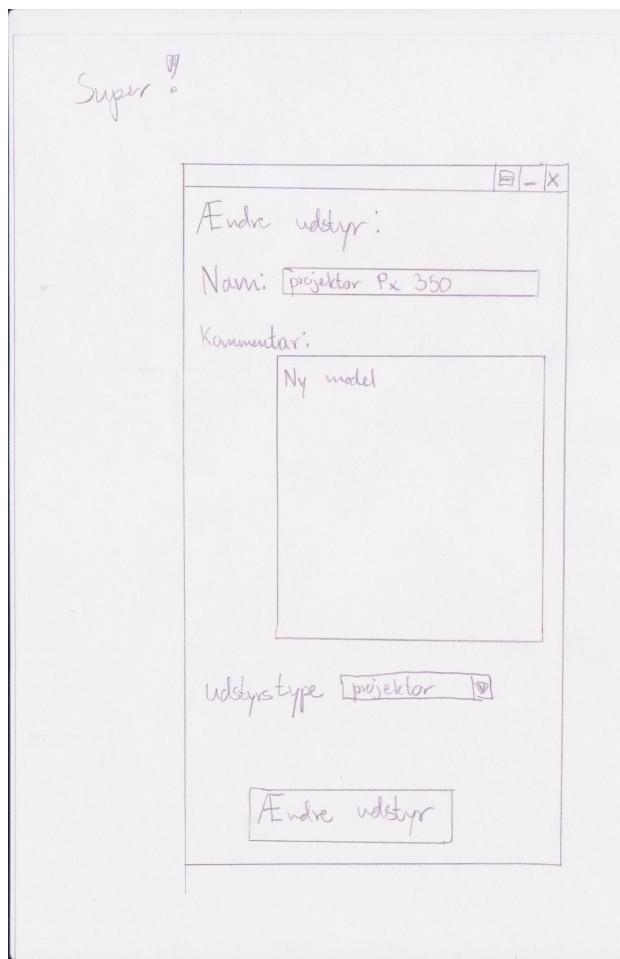


Figure 2.3: Papermockup af skærbilledet til ændring af udstyr

Ændre udstyr

Som man kan se på Figur 2.3 var det første skærbillede til ændring af udstyr smalt i forhold til det endelige, Figur 2.4. Grunden til denne ændring skyldtes, at vi fulgte designregel 1 og derfor gerne ville have, at billederne var visuelt konsistense.

Dine bookinger

Som man kan se på Figurerne 2.5 og 2.6 var den eneste ændring af den endelig version af "Dine bookinger" at vi fjernede forklaringen på, hvad de forskellige statuser repræsenterede. Vi har i designet af skræmbilledet 2.6 fokuset på at overholde Gestalt-lovene [SL, s. 68], specifikt Law of Proximity⁴, så det virkede naturligt at

⁴"Pieces that are close together are perceived as belonging together".

2.2. BRUGERGRÆNSEFLADENS UDVIKLING OG UDSEENDE

Figure 2.4: Skærbilledet til ændring af udstyr

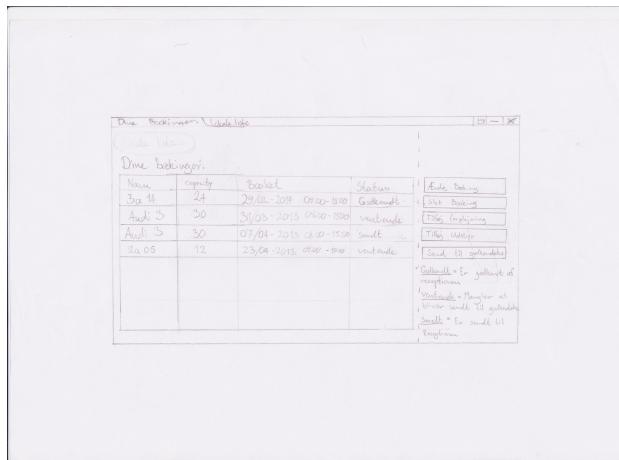


Figure 2.5: Papermockup af skærbilledet til visning af bookinger

Figure 2.6: Skærbilledet til visning af bookinger

knapperne til højre hørte til gitteret.

Godkend bookinger

Figurerne 2.7 og 2.8 viser henholdsvis papermockupen og det færdige skærbilledede af ”Godkend booking” disse viser listen over hvilke bookinger, der mangler at blive godkendt eller afvist af receptionisten. Forskellen fra papermockupen til det færdige skærbilledede er, at vi har valgt at flytte godkend og avis knapperne fra at være tilknyttet til hver enkelt booking til at være uden for gitteret, således at vi ligesom i figur2.6 benytter ”Law of proximity”. Desuden havde ingen af de andre gittre knapper i sig, så vi vurderede, at det burde være ens på alle billeder således at vi overholdt designregel 1.

2.2. BRUGERGRÆNSEFLADENS UDVIKLING OG UDSEENDE

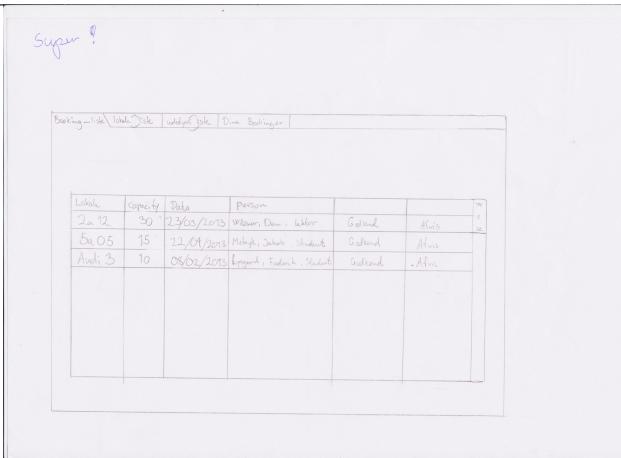


Figure 2.7: Papermockup af skærbilledet til visning af bruger bookinger

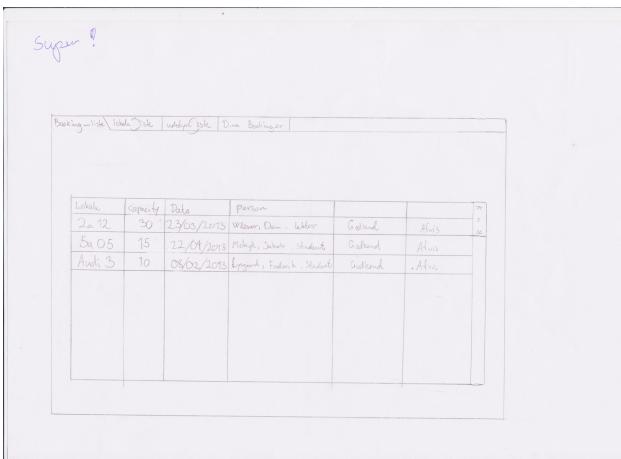


Figure 2.8: Skærbilledet til visning af bruger bookinger

Udstyrsliste

Figurerne 2.9 og 2.10 viser papermockupen og det endelige skærbilledet af listen over udstyr og tilføjelse af nyt udstyr til systemet. Til designet af skærbilledet har vi brugt "Law of proximity" så det er tydeligt, at de to funktioner i skærbilledet ikke hører sammen. Den eneste ændring der er blevet lavet til det sidste skærbilledet er, at der er blevet tilføjet en checkbox, hvor man kan vælge om udstyret skal kunne udlånes. Hvis den ikke udfyldes vil udstyret blive kvalificeret som inventar og det vil derfor ikke blive vist under muligt udstyr, der kan tilføjes en booking.

2.2. BRUGERGRÆNSEFLADENS UDVIKLING OG UDSEENDE

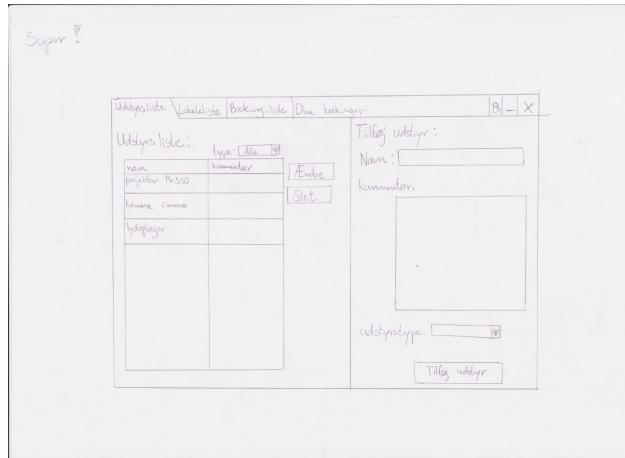


Figure 2.9: Papermockup af skærbilledet til tilføjelse og visning af udstyr

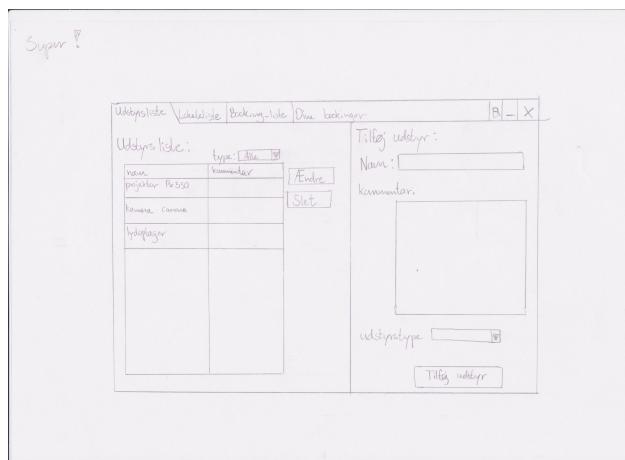


Figure 2.10: Skærbilledet til tilføjelse og visning af udstyr

3 Usability testing

Dette kapitel beskriver, hvad vores strategi for usability testing har været samt resultaterne af vores tests.

3.1 Test strategi

Kravspecifikationen beskriver, at det er nødvendigt at lave usability tests tidligt i processen for at bevise proof of concept. Vi lavede papermockups af vores skærbilleder for at opfylde dette krav.

Vi valgte at lave to runder af usability tests.

Runde 1 bestod af tests med en papermockup på en enkelt bruger.

Runde 2 bestod af tests af det endelige produkt på flere brugere.

Vi valgte kun at teste én bruger i runde 1, da flere testpersoner tidligt i udviklingsfasen kan føre til en overvældende liste af problemer[SL, s. 416]. Derudover er en enkelt testperson ofte nok til at afsløre de mest seriøse problemer i brugergrænsefladen.

Vi mener, at det havde været optimalt, hvis vi kunne have udført en usability test midtvejs i processen (melleml runde 1 og runde 2), men vi vurderede, at vi ikke ville få nok ud af testen, da det ville være svært at finde tid til at teste nok personer samt implementere de mulige ændringsforslag.

3.1.1 Test cases

Vi har to brugergrupper i vores usability tests. Der er almindelige brugere (studerende/undervisere) og administration (Facility Management/Receptionister). Til hver af disse brugergrupper har vi defineret syv test cases.

Test cases til almindelige brugere:

Test Case 1: Du har booket et lokale fra 9-10 til et møde med din vejleder og du vil gerne have en diktafon med, så du kan optage mødet.

Test Case 2: Du skal holde et møde på tre timer om et par dage. Du skal i den sammenhæng booke et lokale til formålet.

Test Case 3: Et af dine gruppemedlemmer har meldt tilbage, at han skal aflevere sin datter i børnehaven. Mødet kan derfor først holdes fra klokken 10.

Test Case 4: Da mødet ligger om morgen, tænker du, at det vil være fornuftigt, hvis der var noget kaffe og morgenbrød klar.

Test Case 5: Du skal til dit projekt optage en kort reklamefilm og har derfor brug for et kamera.

Test Case 6: Et af dine gruppemedlemmer har kamera med, så du behøver ikke længere booke et hos ITU.

Test Case 7: Deltagerne til dit møde i 2a12 har desværre aflyst.

Test cases til administrationsbrugere:

Test Case 1: En elev har kontaktet dig og gjort dig opmærksom på, at der ikke længere er en projektor i 2A12.

3.2. RUNDE 1: RESULTATER OG KONSEKVENSER

Test Case 2: ITU har til sin udstyrssamling erhvervet sig to nye kameraer.

Test Case 3: En student henvender sig til dig ved skranken og spørger om hans booking er blevet godkendt.

Test Case 4: Du har et møde i de kommende dage og vil helst gerne holde det på 4. sal.

Test Case 5: Du skal afholde et fordrag for 30 mennesker den 24, tidspunktet er irrelevant men der skal være plads til min. 30 mennesker.

Test Case 6: Lokalet 5a12 er ikke længere et privat lokale, men skal istedet registreres som et mødelokale.

Test Case 7: Lokalet 2a12 er ikke længere til rådighed i forbindelse med booking.

Vi har designet vores test cases således, at testpersonen ikke får at vide hvor i systemet, de skal udføre deres arbejdsopgave. Dette har vi gjort for at finde ud af, hvor intuitivt vores system er.

De 14 test dækker til sammen störstedelen af systemets workflows. Vi udfører testene som ”think aloud” tests[SL, s. 421]. ”Think-aloud” tests består af at læse test casen op for testpersonen og bede dem tænke højt og forklare, hvad de gør under testen.

Efter testen får brugeren mulighed for at give input til, hvad de synes er godt og hvad der kan forbedres. Dette giver os mulighed for at få et indtryk af, hvad vi har overset i vores design og hvilke elementer, vi bør bruge flere steder i systemet.

I første runde af tests havde vi fokus på de store problemer i systemet, hvor vi i anden runde havde fokus på at følge op på de ændringer, vi havde lavet i forhold til resultaterne fra første runde. I anden runde var vi også interesserede i at finde ud af, om vores system er intuitivt og effektivt nok.

3.2 Runde 1: Resultater og konsekvenser

Efter den første runde af tests var der fem problemer, problemerne er klassificeret i forhold til skalaen fra [SL, s. 439]:

Punkt 1: Minor problem - Usikker på hvordan man vælger booking tidspunkter.

Punkt 2: Cumbersome - Brugeren synes det var besværlig at navigere imellem skærbillederne.

Punkt 3: Medium problem - Svært ved at forstå sletning af forplejning/udstyr og booking.

Punkt 4: Minor problem - Problem med at tilføje udstyr/forplejning.

Punkt 5: Positiv feedback på den generelle struktur af systemet.

Punkt 1 Test personen havde problemer med at finde ud af, hvordan vores gitter til booking af lokaler fungerede. Personen troede ikke, man kunne klikke på felterne i gitteret, så der skulle lidt hjælp til, før personen forstod princippet.

Konsekvenser og løsning Vi blev enige om, at dette problem kunne løses ved at indsætte en checkbox i felterne. Det vil gøre det mere intuitivt, når man skal trykke på tiderne.

Punkt 2 Testpersonen følte, at der manglede nавигeringsmuligheder mellem de forskellige skærbilleder. Det var ikke intuitivt, at man skulle bruges tabsne til at skifte skærbilleder.

Konsekvenser og løsning Vi løste problemet ved at lave en menubar, som var placeret nærmere midten af billedet, så den blev mere synlig.

3.3. RESULTATER AF USABILITYTEST RUNDE 2

Punkt 3 Testpersonen fandt det ikke naturligt, at man skulle bruge lokalelisten til at ændre sin booking.

Konsekvenser og løsning Da det var første test af vores system og det kun var testet på en person valgte vi ikke at gøre noget ved problemet da vi mente, at vores grundstruktur i systemet var solid og derfor ikke burde ændres kun på grund af det problem.

Punkt 4 Testpersonen nævnte, at hun ikke mente, det burde være muligt at tilføje udstyr/forplejning til en booking før, den er blevet godkendt.

Konsekvenser og løsning Vi valgte at imødekomme dette ved at implementere logik, så man ikke kan trykke på knapperne til at tilføje forplejning/udstyr, hvis man ikke har valgt en godkendt booking.

Punkt 5 Testpersonen kunne godt lide den overordnede struktur af systemet og var positiv overfor booking funktionaliteten (gitteret). Personen skulle bare lære at bruge booking funktionaliteten.

Konsekvenser og løsning Denne feedback havde stor indflydelse på, at vi valgte at beholde vores grundstruktur.

3.3 Resultater af usabilitytest runde 2

3.4 Mulige konsekvenser af usabilitytest runde 2

3.5 Sammenligning med alternative systemer

Der findes allerede systemer til booking af lokaler. I dette afsnit vil vi kigge kort på tre af disse og sammenligne dem med vores egen løsning.

3.5.1 Room booking system

”Room booking system” er et webbaseret system som er designet til både at kunne opfylde skolers og virksomheders behov i forhold til booking af lokaler. Interfacet er designet rundt om en kalender, der viser de allerede eksisterende bookinger i farvekoder. Systemet understøtter ikke booking af udstyr eller forplejning og det er ikke koblet op til et AD og der vil derfor være brug for yderligere systemer for at kunne opfylde alle ITU’s behov. Et abonnement som understøtter 200 rum og 500 brugere koster 434 kr. om måneden. Ulempen ved at vælge ”Room booking system” i forhold til vores løsning er som nævnt, at det hverken understøtter forplejning eller udstyr booking, der skal derfor ligges en hel del ressourcer i at få inkorporeret det i løsningen, hvis det da overhovedet er muligt. Hjemmeside til Room booking system: <http://www.roombookingsystem.co.uk/>

3.5.2 School booking

”School booking” er et webbaseret system som primært er designet til brug af skoler. Systemet giver mulighed for både at booke lokaler og udstyr, derudover er der mulighed for, at man kan tilføje ressourcer som eksempelvis kan være forplejning. Systemet giver også mulighed for statistisk udtræk over, hvor ofte en ressource bliver booket samt af hvem, men det understøtter ikke adgang til et AD. GUI’en er simpel og fungerer ligesom vores system, hvor man i et gitter vælger den ressource man gerne vil booke. Den årlige pris for at have ”School booking” kørende for en institution på ITU’s størrelse er omkring 11.000 kr. hvilket inkluderer mulighed for opsætning af 1000 ressourcer, 1000 brugere, samt mulighed for statistik og tilladelse til, at eksterne kan bruge systemet. ”School booking” er klart den dyreste løsning af dem vi nævner, men den er samtidig den bedste. Med undtagelse af opkoppling til AD og en ordentlig integration til kantinen understøtter det alle de funktioner som ITU kræver af et booking system. Sammenlignet med vores system,

3.5. SAMMENLIGNING MED ALTERNATIVE SYSTEMER

skal der ikke særligt meget arbejde til for, at ”School booking” vil være en ligeså god løsning, det kan dog blive problematisk at koble AD op til det, da det skal være hostet på ITU’s server for at fungere. Hjemmeside til School Booking: <http://www.schoolbooking.com/>

3.5.3 Outlook

Outlook giver mulighed for at resevere lokaler, når man inviterer folk til møder, denne løsning giver dog kun mulighed for at booke lokaler og giver ikke et særligt godt overblik, da de enkelte lokaler skal tilvælges og man derfor ikke får et fuldt overblik. Løsningen ville dog være nemt at implementere da ITU er gået over til at bruge microsoft 365, som understøtter outlook som mail klient. Den store ulempe ved at vælge Outlook i forhold til vores system er, at det ikke nemt og automatisk giver brugeren et fuldt overblik over, hvilke lokaler der er ledige og hvilke der er booket. Derudover understøtter Outlook hverken forplejning eller udstyrbooking og der skal derfor ligges meget arbejde i at udvikle systemer ved siden af til at understøtte de funktioner.

3.5.4 Cost benefit analyse

Da vi skulle implementere vores system var det vigtigt for os, at vi valgte den korrekte implementationssstrategi i forhold til, hvor meget gavn brugeren ville få ud af de forskellige løsninger. Vi overvejede derfor hvilken nytte brugerne ville få hvis vi brugte en implementationsmulighed frem for en anden og hvor meget det ville koste os at implementere det. For at få et overblik over, hvordan disse ting forholdt sig til hinanden valgte vi at lave et skema der viste nytten af en implementation kontra omkostninger.

	Java	.NET	WCF (.NET)	Webservices (Java)	Restful (.NET)	Restful (Java)
Thick browser client	XX	XX				
	O	OO				
X	X					
Thick application client	O%	O				
			XXX	XXXX	XXX%	XXXX
			OOOO	OOO	OOOO%	OOO%
			XX	XXX	XX%	XXX
Webservice (browser client)						
Webservice (app. client)			OO%	OO%	OO	OOO

Figure 3.1: Tabel over cost/benefit i forhold til de forskellige implementations muligheder

Figur 3.1 viser vores cost benefit skema over de forskellige implementationsmuligheder. De røde X'er repræsenterer omkostningerne for at implementere den tilsvarende kombination. For at komme frem til omkostningerne af de forskellige implementationsmuligheder tog vi udgangspunkt i hvorvidt vi først skulle tilegne os viden eller om vi allerede havde den fornødende forståelse og således kunne gå i gang med det samme. eksempelvis ville en ”Java, Thick browser client” løsning være dyrere end en ”Java, Thick application client” løsning da vi ikke ved, hvordan man laver en browser klient i java. Derefter kiggede vi på, hvor meget kode der skulle skrives til de forskellige implementationsmodeller for at kunne få et fungerende system. Eksempelvis er alle Webservice løsninger dyrere end Thick client løsninger, da der skal kodes mere, hvis der skal laves en webservice.

De grønne O'er repræsenterer den gavn som brugeren får af den givne implementationsmodel. En af måderne, hvor vi fandt ud af hvor meget gavn de forskellige implementationsmuligheder gav brugeren var at se på, hvor mange platforme der var understøttet.

Eksmepelvis synes vi, at det gav brugeren mere nytte, hvis deres system kunne køre på alle computerere i en browser end hvis man skulle downloade en klient og tjekke om den kunne køre på computeren. Derudover kiggede vi også på, hvor nemt det ville være for brugeren at udvikle videre på systemet, eksempelvis giver implementationsmuligheden ”WCF(.NET), Webservice(browser client)” brugeren mulighed for at udvikle en applikation til windows phone da der er en service at kode op imod.

Udfra de måder at vurderer cost benefit på vurderede vi, at den løsning hvor kunden ville få mest gavn var ”WCF(.NET), Webservice (browser client)” da det ville give brugeren mulighed for at kunne køre systemet på

3.6. LESSONS LEARNED

stort set alle computerere med internet connection og, at de ville være i stand til at lave en mobilapplikation op imod webservicen.

3.6 Lessons learned

I løbet af projektforløbet var der ting som vi gerne ville have gjort anderledes og ting vi burde have overvejet bedre før vi gav os i kast med dem. Vi vil i dette afsnit gennemgå de vigtigste ting vi har lært og hvordan det har påvirket projektet.

3.6.1 Kravspecifikationen er ikke fejfri

Da vi startede projektet var vi alt for sikre på, at kravspecifikationen var en fejfri vejledning som bare skulle følges og så ville vi stå med et godt og færdigt projekt vi fandt dog i løbet af projektet små fejl, hvilket gjorde at vi blev nød til at lave nogle ændringer som vi ikke havde regnet med skulle laves.

3.6.2 Kravspecifikationen er en guide til et færdigt produkt

Da vi satte os for at udvikle første release af systemet gik vi efter at opfylde alle krav fra kravspecifikationen der vi vurderede, at de alle skulle være med i første release. Det som vi ikke overvejede var at kravspecifikationen var en guide til hvordan man skulle udvikle et fuldt fungerende booking system for ITU. Dette var problematisk da vi ikke udviklede hver funktionalitet for sig selv men prøvede at udvikle det hele samtidig, vi endte derfor med at prøve at inkludere for meget funktionalitet på kort tid.

Det vi lærte af den fejl er, at man skal bruge tid på at overveje hvor meget tid man har til rådighed og hvor meget funktionalitet der skal implementeres, og prioritere hvad der er vigtigst at få implementeret først, således at man ikke prøver at lave et fuldt produkt på den tid det ville tage at lave en first release.

3.6.3 Projekter af samme natur er ikke nødvendigvis ens

Et stort problem som vi lærte meget af var forventningen til hvad det krævede at lave booking system. Vi havde tidligere i vores studie lavet et booking system til film som også var skrevet i c# som havde en bagvedliggende service, vi følte derfor vi havde en god basic for hvad sådan et projekt krævede og hvor meget tid der skulle bruges til at implementere det. Det viste sig dog at selvom grundstruktur var ens for de to projekter, at det ikke var ensbetydende med, at de var identiske, derudover undervuderede vi også hvor meget flere gruppemedlemmer gjorde ved arbejdsmængden, i film booking systemet var vi fem gruppemedlemmer, hvilket vi mente vi kunne gøre op for to mennesker hvis vi var mere dedikeret. Vi undervuderede også hvor meget funktionalitet der egentlig

3.6.4 En service er ikke altid bedst

3.6.5 Gitter løsning, godkend men kompliceret

Bibliography

[SL] Lauesen, Søren. User Interface Design, A Software Engineering Perspective. Great Britain: Pearson Educated Limited, 2005. Print.