Mittuniversitetet
MID SWEDEN UNIVERSITY

Rumsreservationssystem
- En komplett designprocess

Julia Andersson
Jonathan Ek

MITTUNIVERSITETET
Avdelningen för informationssystem och -teknologi (IST)
Huvudområde: Datateknik
Högskolepoäng: 15 hp
Termin/år: VT 2020
Handledare: Rahim Rahmani
Examinator: Ulf Jennehag
Utbildningsprogram: Datateknik, 180hp

Sammanfattning

Projektets mål har varit att undersöka hur rumsreservationssystem ska utformas så att bokade rum nyttjas och så missförstånd om vilka rum som är bokade ej uppstår. För ändamålet har användargränssnitt skapats genom en prototyp och algoritmer har framställts och implementerats. Framställningen av prototyper samt konstruktion och implementering av algoritmer har skett enligt Design Science. Utvärdering av prototypen har inneburit genomförande av användbarhetstester som har visat att det föreslagna användargränssnittet är lättanvänt. Enhetstester och integrationstester har tillämpats upprepade gånger för att säkerställa att de implementerade algoritmerna fungerar korrekt. En enkätundersökning har genomförts för att ta reda på hur ett befintligt rumsreservationssystem och de föreslagna funktionerna uppfattas. Resultatet av enkätundersökningen tyder på att problemen som identifierats upplevs bland enkätdeltagarna och att de föreslagna funktionerna är eftertraktade. Projektets resultat tyder på att fler enkätdeltagare hade varit önskvärt och att användbarhetstestens resultat kan bli annorlunda om de genomförs på ett riktigt system samt att de bör innefatta personer av olika teknisk kompetens. Analys av algoritmerna nämner antaganden som gjorts för deras framställning, vad som bör tas i åtanke för deras implementation och korrigeringar som kan genomföras. Analys av enhetstesten och integrationstesten nämner att de är begränsade i vissa avseenden. Design Science har varit användbart för projektets syfte men har inte alltid varit perfekt. Framtida arbete innebär att ta reda på hur autentisering för systemen bör utformas, utveckla ett komplett system som implementerar lösningsförslagen, korrigera algoritmerna som framställts samt samla in mer data om hur lösningsförslagen och befintliga system uppfattas.

Nyckelord: Människa-dator-interaktion, Användargränssnitt, XML, HTTP, Java, JSON, Prototyp, Design Science.

Abstract

The goal of the project has been to investigate how room reservation systems should be designed so that booked rooms are utilized and so that ambiguities about booked rooms do not arise. User interfaces have been created through a prototype and algorithms have been designed and implemented. The Design Science method was utilized for activities related to prototypes and algorithms. Evaluation of the prototype involved conducting usability tests which have shown that the user interface is easy to use. Unit tests and integration tests have been applied to ensure that the implemented algorithms work properly. A survey was conducted to examine how an existing system and the proposed features are perceived. The results of the survey indicate that the identified problems are experienced by the respondents and that the proposed features are appreciated. More respondents would however have been desirable. Moreover, the results of the usability tests may differ if they are performed on a complete system. These tests should include people of different technical competence. Analysis of the algorithms mentions assumptions made during design, considerations for implementation and possible corrections. Analysis of the unit tests and integration tests mentions that they are limited in some aspects. Design Science has been useful for the project but has not always been perfect. Future work involves examining how authentication for the systems should be designed, developing a complete system that implements the proposed solutions, correcting the algorithms and gathering more data on how the proposed solutions and existing systems are perceived.

Keywords: Human-computer-interaction, User interface, XML, HTTP, Java, JSON, Prototype, Design Science.

Förord

Vi vill rikta ett tack till vår handledare Rahim. Han har varit en stor hjälp under hela projektets gång med kontinuerliga möten och handledning.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Introduktion

1.1 Bakgrund och problemmotivation

Vi är två datateknikstudenter som skriver vårt kandidatexamensarbete på institutionen för informationssystem och -teknologi. Vårt arbete syftar på att undersöka ett befintligt rumsreservationssystem tillhörande Mittuniversitetet som används inom Sundsvall och Östersunds campus. Genom undersökning hoppas vi kunna föreslå funktioner för en bättre upplevelse och ökad användning av tjänsten.

I nuläget finns det ett system för rumsreservation på Mittuniversitetet i Sundsvall och Östersund. Bokningar kan skapas inom två veckor och ett obegränsat antal bokningar kan göras. Reservationssystemet fungerar både i webbläsaren på dator och i mobila enheter men anpassningen är inte optimal för mobila enheter. Bokningar som görs ligger därefter i schemat hos personen som utfört bokningen. Systemet brister i att säkerställa att studenter och medarbetare nyttjar rum som de bokat då bokningar inte tas bort om rummen inte nyttjas. I stället tillämpas det som kallas 30-minutersregeln vilket innebär att en bokning blir ogiltig om bokaren inte börjat använda rummet 30 minuter efter bokningens början. Ett annat problem med systemet är att studenter och medarbetare kan ändra sin bokning när som helst, även när bokningen har startat, vilket kan leda till missförstånd.

1.2 Frågeställning

Här presenteras frågeställningen som formulerades i början av projektet och som ska besvaras vid projektets slut.

Hur konstrueras rumsreservationssystem som säkerställer att bokade rum nyttjas samt att det inte uppstår några missförstånd om vilka rum som är bokade?

1.3 Övergripande syfte

Det övergripande syftet avser att påvisa vad som bör genomföras för att besvara frågeställningen.

Projektets övergripande syfte är att undersöka tekniska såväl som designmässiga lösningsförslag för utformning av rumsreservationssystem. På den tekniska sidan ska utformning och implementering av systemets serverdel undersökas. På den designmässiga sidan ska det tas reda på hur

användargränssnittet som hör till systemet bör se ut och hur det ska utformas för att göra systemet lättanvänt genom mobila enheter.

1.4 Avgränsningar

I detta projekt valdes avgränsningarna för studien av Mittuniversitetets rumsreservationssystem. Projektet avgränsas till att enbart undersöka behov och utformning av ett användargränssnitt som är lätt att använda på mobila enheter samt att bidra med förslag på utformning av systemets serverdel.

1.5 Konkreta och verifierbara mål

- Framställa och implementera en algoritm för att begränsa antalet bokningar en användare kan göra
- Ta reda på hur funktionalitet för gruppbokning bör utformas
- Påvisa hur testmetodiken som specificeras i kapitel 3 kan användas för att säkerställa systemets korrekthet
- Framställa prototyp av användargränssnittet som tillhör systemet
- Genomföra användbarhetstest på framställda prototyp för att utvärdera användargränssnittets utseende och utformning. Målet är att ta reda på hur ett lättanvänt användargränssnitt framställs.
- Genomföra en enkätundersökning för att ta reda på hur studenter och medarbetare vid Mittuniversitetet uppfattar Mittuniversitetets befintliga rumsreservationssystem samt ny funktionalitet som kan ingå

1.6 Översikt

Rapporten inleds med en sammanfattning som ger en överblick över hela projektets innehåll. Kapitel 1 är en introduktion som bland annat innefattar bakgrund, problem, vilka avgränsningar som tagits samt de mål som ska undersökas och uppfyllas under projektets gång. Kapitel 2 beskriver verktyg som har använts för att genomföra projektet. I kapitel 3 beskrivs metoder som har varit till hjälp för att uppfylla de mål som varit satta och i kapitel 4 presenteras tillämpningen av dessa. Kapitel 5 presenterar konstruktion och implementering. I kapitel 6 presenteras resultat och i följande kapitel 7 analyseras dessa resultat. Slutligen i kapitel 8 diskuteras projektet i sin helhet när det kommer till framtida arbete och återkoppling till frågeställningen. Här diskuteras även om de konkreta målen uppnåtts.

1.7 Författarens bidrag

Jonathan Ek har i projektet ansvarat för undersökningen om serverdelen, vilket har inneburit datamodellering och implementering av databaser samt framställning och implementation av algoritmer. Jonathan har även ansvarat för genomförandet av enhetstester och integrationstester. Han har även sett över nödvändiga korrigeringar av programkoden, databaserna och de algoritmiska lösningsförslagen. Julia Andersson har i projektet ansvarat för datainsamling genom enkätundersökning och användbarhetstest. Utifrån enkätundersökningen sammanställdes en kravfångst och analys av all insamlat resultat genomfördes. Julia har även ansvarat för skapandet av den interaktiva prototypen och användbarhetstester på den. Efter genomförandet av testerna modifierades prototypen för att illustrera testpersonernas förslag på förbättring.

2 Teori

Här förklaras de tekniker och verktyg som använts under projektet. Här ingår mjukvaror, internettjänster, protokoll, formattekniker och designverktyg.

2.1 NetBeans

Netbeans är en integrerad utvecklingsmiljö för datorprogram. Den är i synnerhet till för programmering i Java men stöd finns även för programmeringsspråken C++, PHP med mera.

2.2 GitHub

GitHub är en internetbaserad versionshanteringstjänst som används i samband med versionshanteringssystemet Git. Git tillåter utvecklare att ha flera så kallade "branches" aktiva, vilka kan ses som självständiga versioner av en och samma källkod. Det finns flera fördelar med upplägget så som att nya "branches" kan skapas för att vidareutveckla mjukvara och experimentell källkod kan författas utan att påverka originalkoden.[1]

2.3 Prototyp

En prototyp är en interaktiv design som består av klickbara ytor för att likna den slutliga produkten i användandet samt design. Genom att låta användare testa interagera med gränssnittet kan utvärdering vidtas och förbättringarna implementeras därefter på den slutliga produkten.[2]
En prototyp kan presenteras på olika sätt och nivåer. Nivåerna är Low-fidelity som innebär ett delvis färdigt system men som endast visas med papper och penna, det ger inte någon vidare bild av slutprodukten eller dess funktionalitet. Medium-fidelity innebär enkla demonstrationer av användargränssnittet där användaren ser hur innehåll presenteras men utan att väga in såsom färg och stil. Slutligen är high-fidelity en demonstration som liknar den slutliga produkten genom att innefatta möjligheten att interagera med systemet på samma sätt som den slutliga produkten är tänkt att göra.[3]

2.4 Extensible Markup Language

Extensible Markup Language (XML) är ett märkspråk för att formatera dokument på ett sätt som är läsbart för människor och datorer. Det är ett format som tillåter en att genom så kallade taggar enkelt ge innebörd till innehållet i ett dokument. På grund av dess goda beskrivningsförmåga är XML lämpligt för informationsutbyte mellan olika system och program[4].

```
<?xml version="1.0" encoding="iso-8859-1"?>
<identitetskort>
  <förnamn>Kent</förnamn>
  <efternamn>Agent</efternamn>
  <titel>Hemlig Agent</titel>
  <kontaktinformation>
    <telefon>070112</telefon>
    <e-post>kenta@hemlig.se</e-post>
  </kontaktinformation>
  <hemsida>www.hemlig.se</hemsida>
</identitetskort>
```

Figur 2.1: Exempel på XML

I Figur 2.1 ovan visas ett exempel på XML. Här representeras ett identitetskort som innehåller förnamn, efternamn, titel, kontaktinformation och hemsida. Lägg märke till att telefon samt e-post utgör innehållet utav kontaktinformation.

2.5 JavaScript Object Notation

JavaScript Object Notation (JSON) är ett textformat för dataöverföring som är enkelt att hantera för människor och datorer. Formatets två beståndsdelar är objekt och arrayer. Ett objekt är en samling namn-värde par som separeras genom komman. Ett objekt börjar med "{" och slutar med "}". Varje namn följs av ett kolon. Exempelvis så är { "A" :1, "B" :2} ett objekt med två beståndsdelar. En array börjar med "[", slutar med "]" och medlemmarna separeras genom komman[5]. Exempelvis så är [1,2,3] en array bestående av tre värden. Det finns flera olika datatyper som klassificeras som värden men några anmärkningsvärda är arrayer och objekt.

2.6 Glassfish

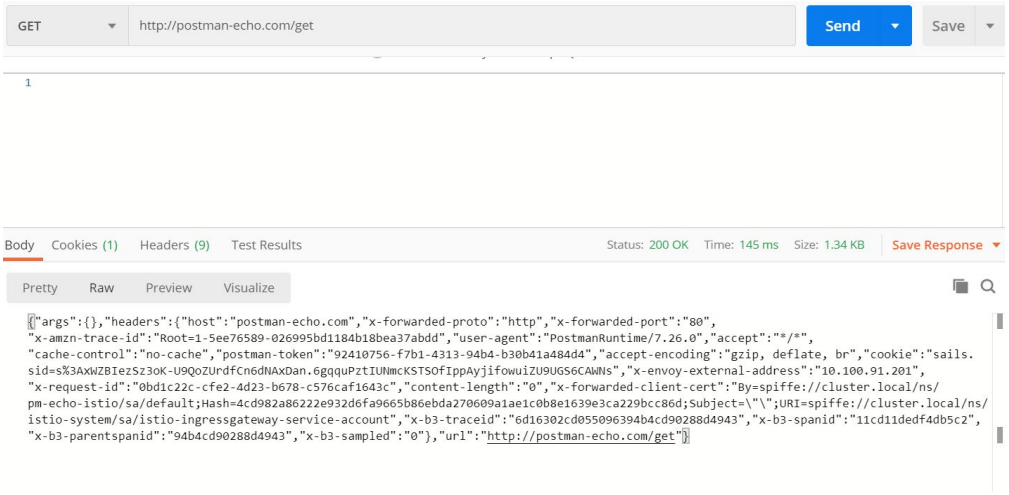
Glassfish är en applikationsserver som är världens första implementation av Java Enterprise Edition (Java EE) 6. Java EE är industristandard för Java-teknologi.[6]

2.7 HyperText Transfer Protocol

HyperText Transfer Protocol (HTTP) är ett applikationsprotokoll enligt TCP/IP-modellen som opererar enligt klient-server modellen och används bland annat för överföring av webbsidor över Internet. Protokollet definierar diverse kommandon som klienten kan skicka till servern. En av dem är *GET*, som bland annat kan be servern att skicka en given fil till klienten.[7]

2.8 Postman

Postman är en programvara som tillhandahåller möjligheten att använda diverse HTTP-metoder för att skicka eller hämta data. De vanligaste metoderna är *GET* för att hämta data, *POST* för att skicka ny data till servern, *PATCH* och *PUT* för att uppdatera befintlig data och till sist *DELETE* för att ta bort befintlig data[8]. I Figur 2.2 nedan påvisas användning av Postman. Här har metoden *GET* använts för att hämta data i JSON-format.



Figur 2.2: Demonstration av Postman

2.9 Doxygen

Doxygen är i praktiken ett standardverktyg för framställning av dokumentation utifrån C++-kod men kan även användas i samband med Java bland annat. Dokumentationen framställs direkt från källkoden och kan bland annat formateras som HTML. Det är även möjligt att hämta ut och visualisera strukturen av odokumenterad källkod.[9]

3 Metod

Detta kapitel kommer förklara de metoder som ligger till grund för studiens utförande. Avsnittet kommer således inte behandla hur dessa metoder kommer att tillämpas för studien utan endast metodernas tillvägagångssätt och syfte. Metodtillämpning kommer beskrivas i nästkommande avsnitt.

3.1 Design Science

Design Science är en vetenskaplig metod vars syfte är att bidra med kunskap om framställningen av objekt som avser att uppnå önskade mål, så kallade artefakter. Det kan även uttryckas som kunskapen om hur artefakter som uppfyller en viss funktion skapas. I anslutning till detta är Design Science Research till för att skapa sådan kunskap. Denna process innefattar stegen Identifiera problem, Föreslå lösning, Implementera lösning, Utvärdering och Slutsatser som presenteras nedan.[10]

- Identifiera problem
  Här identifieras vilka problem det är som ska lösas och hur den slutgiltiga produkten ska utvärderas. Resultatet av denna del är ett förslag på vad som ska undersökas.

- Föreslå lösning
  Detta är en kreativ fas där ny funktionalitet skapas genom existerande eller nya tekniker. Här specificeras hur problemen ska lösas och resultatet utifrån denna fas är en preliminär lösning.

- Implementera lösning
  Den preliminära lösningen vidareutvecklas och implementeras här och resultat av denna fas är en artefakt.

- Utvärdering
  Här ska den skapade artefakten utvärderas enligt de kriterier som specificerats och avvikelser från förväntningar noteras och förklaras. Om avvikelserna är stora kan fas "Föreslå lösning" återbesökas för att behandla dessa.

- Slutsatser
  Den sista delen av metoden nås när resultaten anses vara tillräckliga, även om det fortfarande finns avvikelser från (möjligen reviderade)

förväntningarna. Resultaten presenteras och kunskapen som har tillkommit klassificeras antingen som hård eller mjuk. Hård kunskap är kunskap som tillkommit och som kan återanvändas i andra projekt medan mjuk kunskap är avvikande beteende som saknar förklaring och därav kan utgöra ytterligare undersökning. Här motiveras att den genomförda undersökningen bidrar med kunskap. Det är även möjligt att berätta om vad som fungerade och inte fungerade samt dra slutsatser om resultatens generella applicerbarhet.

3.2 Testmetodik

För att verifiera att den skapade artefakten fungerar som den ska genomförs olika sorters tester. Testerna innefattar små(enhetstest), medelstora(integrationstest) och stora tester(UI-test). Rent allmänt är enhetstesterna enklare att genomföra än integrationstesterna och integrationstesterna är enklare att genomföra än UI-testerna. Därför är enhetstester vanligare än integrationstester och integrationstester är vanligare än UI-tester.[11]

3.2.1 Enhetstest

Enhetstester är relativt små tester som avser att kontrollera att de minsta beståndsdelarna i ett datorprogram fungerar som tänkt. Exempel på sådana beståndsdelar är klasser och i detta fall innebär enhetstest att de funktioner som klasserna tillhandahåller testas. Testerna anses vara lyckade om de enheter som testas visar sig fungera korrekt. En svårighet med enhetstest är att konstruera tester som täcker alla möjliga sätt en enhet kan användas på och därför genomförs ett urval av tester som anses utvärdera enheten tillräckligt.

3.2.2 Integrationstest

Enhetstester tar inte hänsyn till hur olika enheter samspelar och för att testa detta kan integrationstest användas. Integrationstester är tester som validerar samspelet mellan en grupp av enheter. I denna typ av test är det svårt att konstruera tester som täcker alla möjliga sätt en grupp enheter kan samspela och ett urval av tester som anses utvärdera samspelet tillräckligt genomförs.

3.2.3 UI-test

Integrationstest tar inte hänsyn till hur programmet fungerar som helhet. UI-tester avser att validera arbetsflöden som innefattar ett flertal funktioner av datorprogrammet.

3.3 Enkätundersökning

En enkät är en undersökningsteknik som samlar in svar från ett antal respondenter för att sedan sammanställa dess svar. Enkäten består av ett antal frågor och frågorna som ställs kan antingen ha angivna svarsalternativ som visas med kryssrutor, en skallinje eller vara utformade så att användaren själv formulerar sina svar. Formen på enkät kan variera men oftas skickas enkäten enklast ut via ett online-frågeformulär. Ju fler inkomna svar desto bättre och tydligare blir resultatet då ett större antal respondenter och svar ger en ökad tillförlitlighet.[2]

3.4 Användbarhetstest

Syftet med dessa tester är att låta användare interagera med produkten/tjänsten och utvärdera den för att komma fram till möjliga förbättringar. Testet utförs för att se hur respondenten uppfattar användandet och detta dokumenteras för att analyseras och hitta brister i användbarheten. Det är viktigt att välja passande användare för just den tjänst/produkt som utvecklas för att få rättvis data att sedan analysera. En passande användare bedöms att sannolikt använda tjänsten i framtiden. När testet utförs kan användarna uppmanas "tänka högt" samt betygsätta den testade produkten/prototypen med hjälp av en skala i ett formulär för att vara till hjälp vid vidareutveckling[2].

3.5 Kravfångst

För att samla in krav behövs en definition av syfte och mål framställas. Då krav samlas in från intressenter behöver även gruppen av intressenterna samt omfattning och projektets avgränsningar definieras. En av alla tekniker som kan används vid kravfångst är enkäter och resultatet av dessa ligger till grund för kommande utveckling och leverans. Krav kan vidare kategoriseras i funktionella och icke-funktionella. Funktionella krav ställs på ett system, *vad* som ska göras i form av funktioner. De krav som är icke-funktionella innebär *hur* systemet ska fungera. Dessa krav har betydelse för användarens upplevelse över systemet.[12]

4 Metodtillämpning

Här förklaras hur de nämnda metoderna avses tillämpas under projektet. Tillämpningen avser rubrikerna Design Science, Testmetodik, Enkätundersökning, Etiska aspekter, Användbarhetstest, Verktyg och Kravfångst.

4.1 Design Science

Design Science bedömdes vara en lämplig metod för projektet på grund av dess iterativa förhållningssätt till problemlösning. Den var i synnerhet lämplig för uppdelning av problemen i delproblem vars lösningar modelleras och implementeras självständigt.

- Identifiera problem
  I denna fas analyseras frågeställningen för att ta reda på vilka delproblem som bör lösas för att besvara den. Vid eventuell upprepning av denna fas granskas den skapade artefakten för att ta reda vilka krav den ej uppfyller och för de krav som den uppfyller om något bör förändras.

- Föreslå Lösning
  Detta är en relativt kort fas som upprepas flera gånger. Här väljs ett fåtal av de problem som identifierats med avsikten att identifiera möjliga lösningar.

- Implementera Lösning
  Utifrån lösningsförslagen som identifierats skapas modeller som beskriver dem detaljerat. I dessa modeller ingår flödesscheman som beskriver algoritmerna som framställs och ER-diagram som beskriver entiteterna och förhållandena mellan dessa. En prototyp för möjliga användargränssnitt framställs även. För att få en konkret bild om modellernas praktiska tillämpbarhet implementeras de även i programspråket Java med Glassfish som serverprogramvara.

- Utvärdering
  För att utvärdera användargränssnittet kommer användningsscenarier som ska genomföras på en mobil enhet framställas. Det bör dock noteras att konstruera användningsscenarier som täcker alla möjliga sätt produkten kan användas på inte är praktiskt och därför måste ett

urval som är tillräckligt omfattande för att säkerställa att artefakten fungerar bra göras.

Målet med att ogiltiga bokningar tas bort är att andra personer ska kunna boka i praktiken lediga rum och därav är föremålet för utvärdering här hur väl produkten uppfyller detta.

Likaså är målet med att begränsa ombokningar att behandla problemen med 30-minuters regeln och föremålet för utvärdering är hur väl detta uppnås.

Begränsning av antalet bokningar under ett givet tidsintervall har syftet att begränsa antalet rum en person kan boka och utvärdering här innebär hur väl detta uppnås.

Gruppbokningen avser att registrera två eller flera personer på en bokning och utvärdering här innebär hur väl detta uppnås.

- Slutsatser
  Slutsatserna som dras behandlar huruvida lösningsförslagen besvarar frågeställningen, vilken kunskap som tillkommit, vad som fungerade och inte samt resultatens applicerbarhet.

4.2 Testmetodik

Var och en av de tre testvarianterna avses tillämpas till en viss utsträckning. Enhetstesten är de enklaste testerna och tillämpas därför i störst utsträckning. Integrationstesten är något större än enhetstesten och därför tillämpas de i mindre utsträckning än enhetstesten. UI-testen är de mest omfattande testerna och därför tillämpas de i lägst utsträckning.

4.3 Enkätundersökning

För att ta reda på hur de funktioner som föreslås uppfattas av potentiella användare av systemet genomförs en enkätundersökning. Undersökningen är en *mixed method* det vill säga en blandning mellan en kvantitativ och kvalitativ studie [2]. Användaren svarar på frågor med svarsalternativen *Ja* eller *Nej* och på en linjär skala 1-6 där 1 står för *mindre bra* och 6 för *mycket bra* vilket ger kvantitativ data att analysera. Det förekommer även en fråga där användaren i flytande text får uttrycka sitt svar vilket ger en kvalitativ data att analysera. Enkäten tillhandahålls genom en internettjänst och de frågor som ställs behandlar hur systemet används och uppfattas i nuläget samt hur förslag på

ny funktionalitet uppfattas. De som har möjlighet att besvara enkäten är medarbetare eller studenter vid Mittuniversitetet och spridningen sker genom e-post. Samtliga respondenter är anonyma och informeras om enkätens syfte.

Användning av enkät är lämplig för projektets syfte eftersom sådana är enkla att sprida och därav finns det möjlighet att få ett stort antal respondenter. Det är dock viktigt att antalet respondenter är tillräckligt stor för att få en bra bild av verkligheten. För att kunna dra riktiga slutsatser från svaren bör frågorna även struktureras och presenteras på ett lämpligt tillvägagångssätt. Det finns även risk för oärliga svar och sanningshalten i svaren blir då svår att kontrollera. Genom att ha skala 1-6 i enkäten, ett jämnt antal tvingas användaren att ta ställning till om funktionerna går åt *mindre bra* eller *mycket bra*, istället för ett lagomt betyg. Resultatet blir då lättare att tolka.[3] Se bilaga A för fullständig mall.

4.4 Etiska aspekter

För all data som samlas in efter användbarhetstest och enkät presenteras det tydligt för användaren hur datan kommer att användas och att dessa är helt frivilliga att genomföra. Datan som samlas in genom användbarhetstest och enkätundersökning kan inte användas för att identifiera deltagarna och kommer endast återfinnas i projektrapporten efter avslutat projekt.

Framställning av källkod är en etisk fråga i förhållande till vad användarna tillåts göra med datorprogrammen de använder. All källkod som framställs under projektet är licensierad enligt GPLv3. Därav gäller de fyra rättigheter för källkod som definieras av GNU:

1. Programkoden får användas för vilket syfte som helst
2. Programkoden får modifieras för att tillfredsställa användarens behov
3. Användaren får sprida källkoden till vem som helst
4. Alla ändringar av källkoden får spridas fritt

Ett program som tillhandahåller dessa rättigheter kallas fri programvara. När programvara licensieras enligt GPLv3 förblir det alltid fri programvara, oavsett vem som ändrar eller sprider vidare programvaran.

4.5 Användbarhetstest

Användbarhetstester kommer användas som utvärderingsmetod när det gäller användargränssnitt och är en *mixed method*, en blandning av kvantitativ och kvalitativ studie. Testet utförs på fem personer där varje testperson utför testet på en prototyp tillsammans med ett webbformulär med ett antal uppgifter och frågor kopplat till prototypen. Uppgifterna som återges under

testet är kopplade till användargränssnittet och kan betygsättas på en linjär skala 1-6 där 1 står för *svårt* och 6 står för *enkelt*. Skala 1-6 i enkäten tvingar användaren att ta ställning till om funktionerna går åt *svårt* eller *enkelt* i genomförandet, istället för ett lagomt betyg och resultatet blir då lättare att analysera.[3] Resultatet av svaren på uppgifterna ger en kvantitativ data. Dessa är:

1. Skapa en bokning
2. Skapa en bokning med gruppmedlem
3. Visa dina bokningar
4. Ta bort en bokning
5. Redigera en redan befintlig bokning

Frågorna är kopplade till användning och funktionalitet och ger utrymme för användaren att ge egna kommentarer och synpunkter på prototypen vilket resulterar i kvalitativ data. Dessa är:

1. Vilken påverkan tror du användargränssnittet du testat har på din användning av systemet? Motivera
2. Tycker du något saknas i användargränssnittet du testat? Om ja, vad?

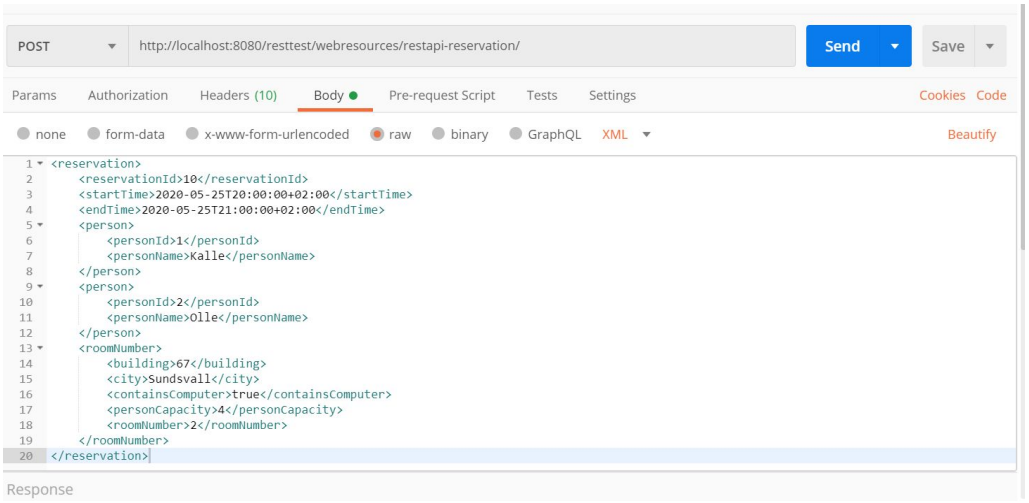
Denna metod ger snabb och konkret återkoppling från potentiella användare som är värdefullt för den fortsatta utvecklingen av produkten. För fullständig mall av användbarhetstestet se bilaga B.

4.6 Verktyg

Samtlig programkod författas i utvecklingsmiljön Netbeans och tillkommer enligt ett arbetsflöde som innebär att utvecklaren identifierar en funktion som ska ingå i datorprogrammet. Tjänsten Github används, en ny “branch” skapas och den identifierade funktionen implementeras. När funktionen är färdig inkluderas den i programkoden som utgör den riktiga versionen av datorprogrammet. Omfattningen på de funktioner som utvecklas enligt arbetsflödet begränsas till att de ej tar längre än en dag att utveckla. Detta medför att funktionerna blir relativt små i omfattning och att källkoden snabbt utökas. Funktioner som bedöms ta längre än en dag att utveckla får delas upp i mindre funktioner. Det utvecklade datorprogrammet exekveras och testas i samband med glassfish som applikationsserver.

Den specificerade testmetodiken tillämpas i samband med programvaran Postman. Den HTTP-metod som används beror på vilken typ av test det är som ska genomföras. Ska nya värden skrivas till databasen används *POST*, ska

en borttagning ske används *DELETE* och ska en redigering genomföras används *PUT*.



Figur 4.1: Demonstration av Postman för att genomföra bokning

I Figur 4.1 ovan demonstreras hur en förfrågan med Postman genomförs. En förfrågan innefattar i synnerhet en länk till den resurs som ska kommuniceras med, HTTP-metoden som används och möjligen en nyttolast som ska överföras till servern. I figuren 4.1 visas det även att det är en *POST* som genomförs och att nyttolasten innehåller XML. Den förfrågan som demonstreras här avser att registrera en ny bokning i Sundsvall med två personer som står för bokningen.

Framställning av dokumentation sker med programvaran doxygen och redovisas i HTML-format. Programkoden dokumenteras genom att beskrivande kommentarer för klasser och metoder skrivs i källkodsfilerna. När källkoden anses vara komplett för projektets syfte används Doxygen för att framställa dokumentationsfilerna.

4.7 Kravfångst

Krav som identifieras och samlas in efter enkätundersökning delas upp i kategorier. Direkt implementerbara krav utgör problem som kan lösas under projektets gång, krav som utgör senare utveckling har bortprioriteras på grund av projektets syfte och framtida krav är krav som saknar kompletta lösningsförslag.

5 Konstruktion

Här diskuteras de föreslagna lösningarnas utformning och framställning. Först presenteras kravfångsten och därefter följer datamodeller, modellering och implementering av algoritmer samt en prototyp av användargränssnittet.

5.1 Kravfångst

Utifrån svaren från den genomförda enkätundersökningen bestäms att ett system som implementerar den föreslagna lösningen med fördel uppfyller dessa krav som visas nedan i tabell 6.1. Undantaget är autentisering som ej baseras på enkätundersökningen utan istället tillkommit från analys av ett befintligt system.

Tabell 6.1: Kravfångst

Direkt implementerbara krav	**Senare utveckling**	**Framtida krav**
Begränsa antal bokningar som sker under ett visst tidsintervall	Autentisering	Ogiltiga bokningar ska behandlas på något sätt som resulterar i att andra kan boka rummen
Begränsning på möjligheten till ombokningar	Använda systemet på mobila enheter	Notis för påminnelse innan bokning
Gruppbokning		

5.2 Datamodellering

För att tydliggöra vilken data som bör finnas inom systemet skapades ett ER-diagram och under förutsättningen att en relationsdatabas används för lagring redovisas även motsvarande basrelationer. Se bilaga C.

5.3 Algoritmer

Av de problem som identifierats är det två stycken som behandlats algoritmiskt. Begränsning av möjligheten till ombokning sker genom att kontrollera att en given bokning som ska redigeras inte är påväg att börja samt att den inte börjat. För att begränsa antalet bokningar under ett givet tidsintervall sker en räkning på hur många bokningar varje person har inom tidsintervallet och om någon person har för många bokningar tillåts inte en ny

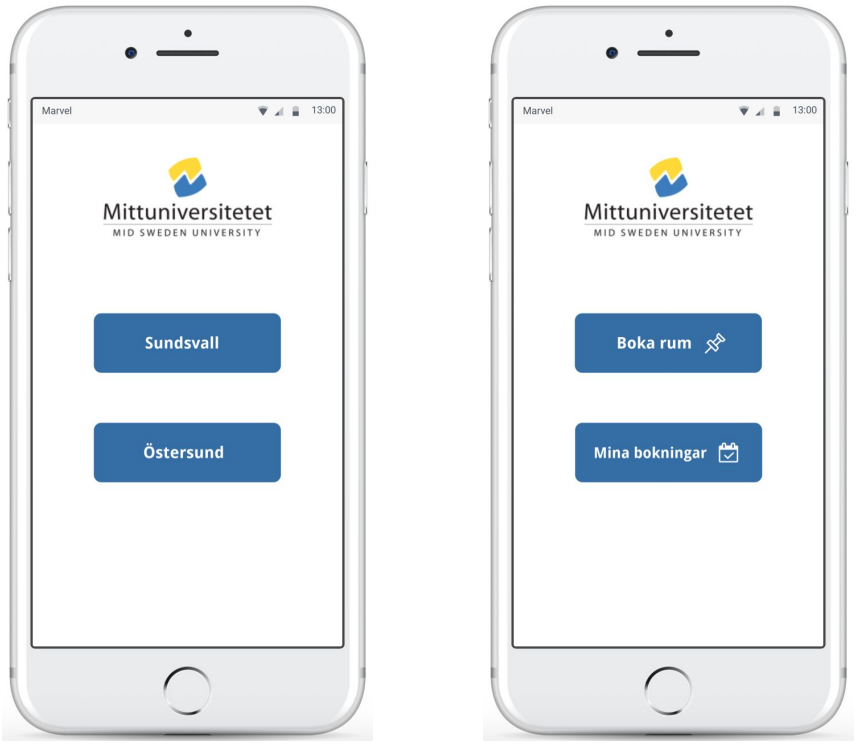
bokning under det givna tidsintervallet. Flödesscheman som beskriver algoritmerna och hur de används i vanliga operationer mot databasen visas i bilaga D.

5.4 Utformning av prototyp

För att genomföra användbarhetstester på gränssnittet skapades en prototyp med Marvel App vilket är ett design och prototypverktyg. Testpersonen skulle då utföra ett flertal uppgifter med prototypen som är av typen high-fidelity vilket innebär att den är likt den vanliga miljön för användandet av mobila applikationer och är möjlig att interagera med. Designen är skapad utefter de framställda och utvecklade funktionaliteterna och färgvalen är tagna från den nuvarande rumsreservationstjänsten som finns på Mittuniversitetet för att följa samma tema. Målet med tjänstens design var att göra den lätt att förstå när det kommer till navigering samt göra den enkel och stilfull. Designen följer användbarhetsmålen:

- Effektiv att använda (effectiveness)
- Effektiv att använda (efficiency)
- Säker att använda
- Har bra verktyg
- Lätt att lära
- Lätt att komma ihåg hur det används

Det första målet uppfylls genom att utveckla produkten så användaren kan utföra uppgifterna på ett effektiv sätt. Det andra målet är uppfyllt om användaren lär sig använda tjänsten med en hög produktivitet. Tredje målet innefattar de fel som kan uppstå och hur dessa kan rättas till vid t.ex. felklick. Nästkommande mål är uppfyllt om tjänsten innehåller alla funktioner som behövs. Det femte målet uppfylls om användaren kan lära sig gränssnittet på ett bra sätt genom utforskning och det sista målet ifall gränssnittet hjälper användaren att komma ihåg hur funktionerna används. De två sista målen går att koppla ihop med användandet av ikoner i utvecklingen av gränssnittet. Att använda ikoner istället för text anses vara en bidragande faktor till att det är lättare för användaren att lära sig och komma ihåg hur tjänsten används.[2]



(a) *(b)*

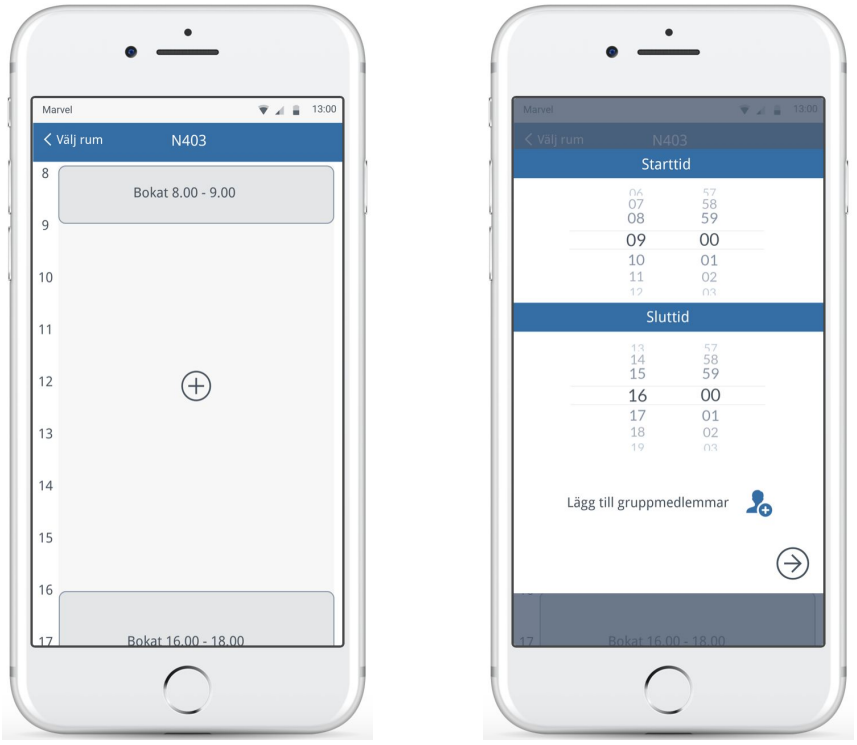
Figur 5.1. Prototyp a. Startsida. b. Sida två

I figur 5.1 (a) ovan visas startsidan med ett val i form av två knappar för att välja den stad som användaren studerar i. Det är ett val som även genomförs i nuläget då de olika skolorna erbjuder olika rum. Vid klick på någon av knapparna visar nästa figur 5.1 (b) att användaren kan välja mellan att boka rum eller visa sina redan befintliga bokningar.

(a) *(b)*

Figur 5.2. Prototyp a. Kalendervy. b. Vy för att välja rum

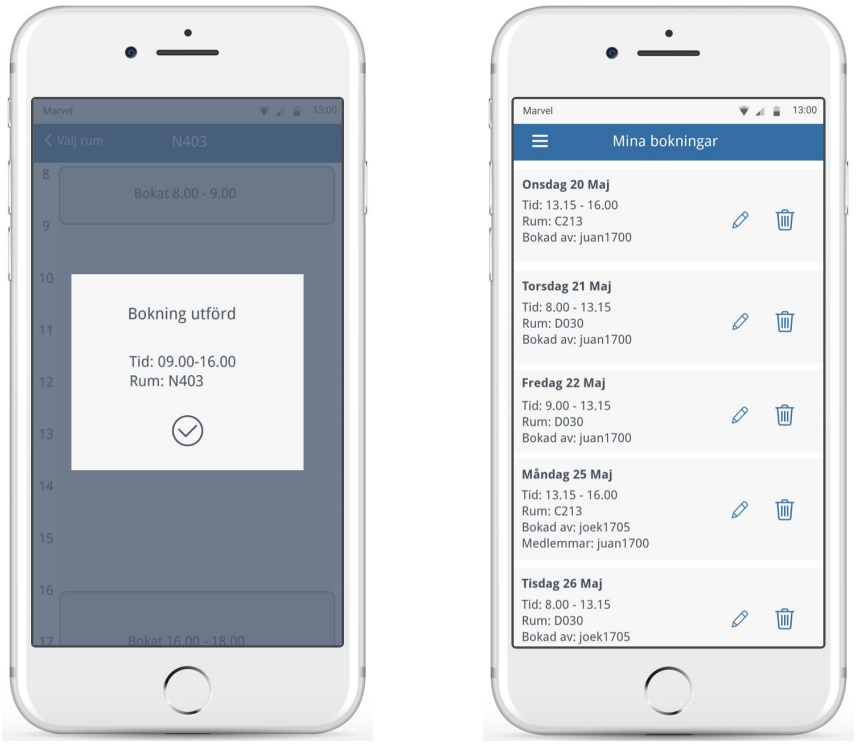
Ovan i figur 5.2 (a) visas en kalendervy där användaren kan välja vilket datum bokningen ska ske. Vidare tas användaren till vyn i figur 5.2 (b) för att välja ett rum att boka. För varje rum framgår vilken byggnad rummet befinner sig i, dess rumsnummer, antal platser och om det finns lediga tider att boka under den valda dagen.



(a) *(b)*

Figur 5.3. Prototyp a. Vy för att välja tid. b. Vy för att välja tidsintervall

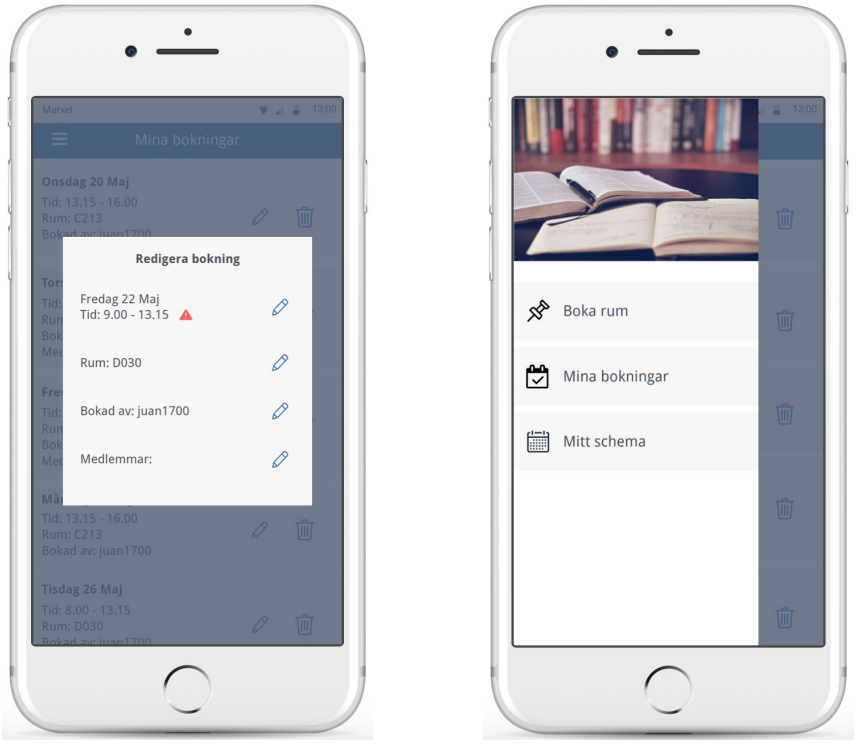
I figur 5.3 (a) ovan visas vyn efter att ett ledigt rum valts för bokning. Här framgår rumsnummer och vilka tider som är bokade under den valda dagen. För boka in en tid klickar användaren på plusikonen. Detta medför att vyn i figur 5.3 (b) visas. Här väljer användaren starttid och sluttid för bokningen samt lägger till eventuella gruppmedlemmar.



(a) *(b)*

Figur 5.4. Prototyp a. Popup bekräftad bokning. b. Vy mina bokningar

Figur 5.4 (a) ovan visar vyn för bokningsbekräftelser och i figur 5.4 (b) visas vyn för en användares befintliga bokningar. För en given befintlig bokning visas dess datum, starttid, sluttid, rumsnummer, bokare, möjliga gruppmedlemmar samt ikoner för redigering och borttagning.



(a) *(b)*

Figur 5.5. Prototyp a. Popup för redigera bokning. b. Sidomeny

Ovan i figur 5.5 (a) visar vyn ifall användaren vill redigera en befintlig bokning och genom att klicka på den ikon för respektive kategori som ska redigeras. I figur 5.5 (b) visas en sidomeny i form av en lista.

5.5 Implementering av algoritmer

De framställda algoritmerna har implementerats i programmeringsspråket Java för att illustrera hur de avses användas i förhållande till diverse operationer gentemot datakällan där samtliga data som ingår i systemet lagras. För implementeringen är datakällan en relationsdatabas som kan skrivas till och läsas ifrån genom diverse HTTP-metoder och formatet på informationen som skickas är antingen XML eller JSON. Netbeans har använts för implementationen med Glassfish som underliggande serverprogramvara.
Se Bilaga E för implementeringen av algoritmerna och hur de används i operationerna mot databasen. Hänvisning sker till [13] för hela källkoden och till [14] för dess dokumentation.

Koden för att skapa en bokning består utav två booleska variabler, variabler som antingen är sanna eller falska. Den ena anger huruvida någon användare registrerad på bokningen måste boka det angivna rummet vid ett senare tillfälle. Den andra anger om någon person registrerad på bokningen har för många bokningar registrerade i det angivna tidsintervallet. Trots att det finns en datastruktur innehållandes personer skapas här temporära personobjekt för att säkerställa att en koppling till det underliggande lagringsutrymmet finns.

För att redigera en bokning finns två instanser av typen reservation. Den ena representerar en bokning som blivit redigerad och den andra representerar samma bokning fast oredigerad. Även här finns två booleska variabler. Den ena anger om den givna bokningen får redigeras. Den andra anger om det för någon person registrerad på den redigerade bokningen finns för många bokningar i det möjligen nya tidsintervallet.

Borttagning av en bokning sker direkt om den inte börjat och om borttagningen inte sker alldeles för nära inpå dess början. Är detta däremot inte fallet så uppdateras den underliggande databasen för att återspegla att varje person som är registrerad på bokningen endast kan boka det givna rummet vid den borttagna bokningens sluttid eller senare.

Koden som kontrollerar huruvida en bokning har en tillåten starttid består av ett uppslag i databasen för att kontrollera om någon användare registrerad på bokningen är begränsad till att boka det angivna rummet vid ett tillfälle som inträffar efter bokningens starttid.

Trots att det finns en datastruktur innehållandes användare skapas temporära personobjekt för att räkna antal bokningar i tidsintervall. Syftet med detta är att säkerställa att en koppling till det underliggande lagringsutrymmet finns.

6 Resultat

I detta kapitel redovisas resultaten av enkätundersökningen, användbarhetstesterna samt enhets och-integrationstesten.

6.1 Enkätundersökning

I projektet har en enkätundersökning genomförts som hjälp för utvärdering. Enkäten skapades för att utvärdera det nuvarande rumsreservationssystemet samt utvärdera om funktionerna som föreslagits som förbättring var till värde för användarna.

Tabell 6.2: Svarsfrekvens enkätundersökning

Enkät	**Antal svar**
Utvärdering av tjänsten	52 st

Svarsfrekvensen på enkäten blev totalt 52 svar. Se bilaga F för exakta svar.

Av de 52 svar som inkommit visar det tydligt att majoriteten använder tjänsten för att boka rum. Angående hur användningen under de tre senaste månaderna varit som nästa fråga löd var det av olika anledningar många som inte använt tjänsten. När det kommer till om tjänsten anses vara lättanvänd är svaret väldigt delat men de flesta respondenterna anser att den varken är lätt eller svår att använda. Mobilanpassningen har varit en viktig del för oss då målet var att göra den mer lättillgänglig och enkäten visar på att totalt 40 personer i nuläget upplever att tjänsten inte är enkel att använda på sin mobila enhet. Bokningar som glöms bort och inte avbokas uppstår men större delen av respondenterna bokar av rummen om bokningen inte kommer nyttjas.
Resterande frågor i enkäten handlar om projektets föreslagna funktioner och mål för att förbättra tjänsten som finns i nuläget. En funktion för gruppbokning och att bokningar som inte nyttjas tas bort från systemet visade sig ha stor uppskattning.

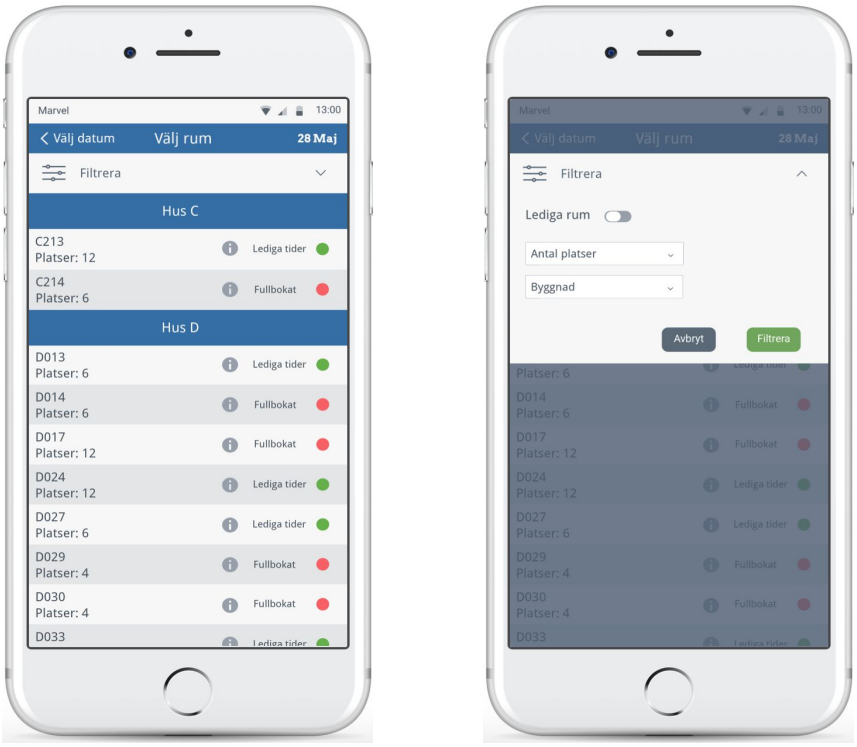
När det kommer till att begränsa tiden för ombokning var resultatet väldigt jämnt de flesta upplevde att den funktionen varken var mindre bra eller mycket bra. En annan funktion som skulle uppfylla målet för projektets ramar var att begränsa användarna så det inte är möjligt att boka flera rum under samma tidsintervall och enkäten visar tydligt att många håller med om att det vore en bra begränsning. En påminnelse innan bokningens början uppskattas av större delen av de totala svaren. Slutligen när det kommer till projektets

mål till ytterligare förbättrad mobilanpassning att tjänsten skulle utvecklas till en app. Resultatet visar att en större andel respondenter ansåg det som mindre bra men en stor del av svaren befinner sig på den mer positiva sidan av den linjerade skalan, att en app hade varit mycket bra.

6.2 Användbarhetstest

Som utvärdering av gränssnittet på prototypen utfördes 5 st användbarhetstester. Testerna utfördes på studenter som har olika inriktningar och varierande tekniska kunskaper. En enkät med uppgifter och frågor tillsammans med prototypen navigerade testpersonerna sig fram för att utvärdera och lämna synpunkter på möjliga förbättringar. Uppgifterna består av svarsalternativen 1-6 på en linjär skala där 1 står för svårt att utföra uppgiften och 6 för enkelt att utföra. I frågorna ges utrymme till testpersonerna att motivera sitt svar. I figurerna nedan visas frågorna som ställdes i användbarhetstestet. Uppgifterna som utfördes av testpersonerna finns i bilaga G.

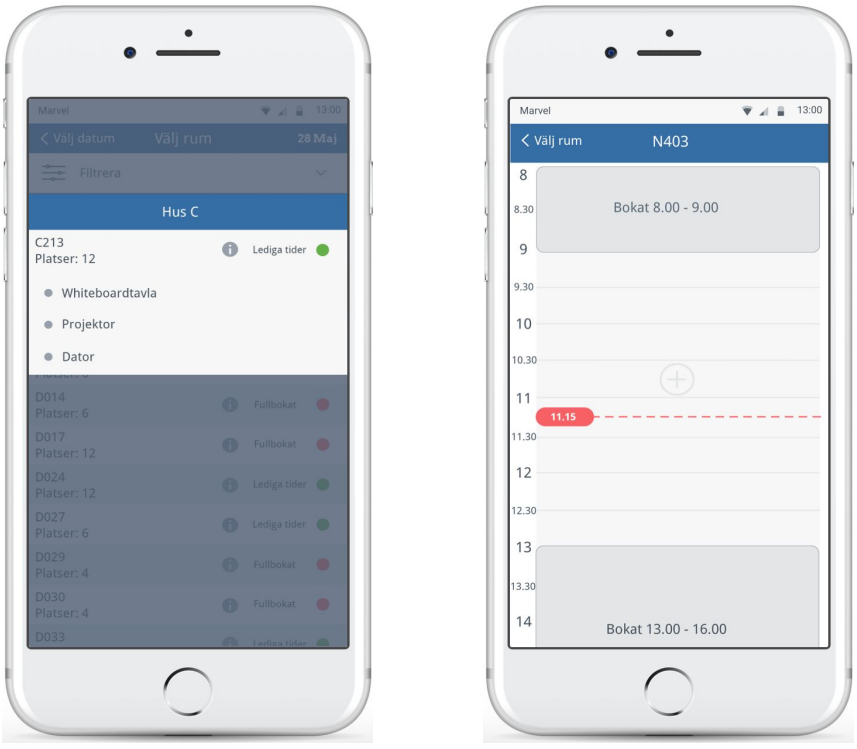
Alla testpersoner har lämnat ett högt betyg på att prototypen var enkel att använda och att tjänsten hade haft en positiv påverkan på den fortsatta användningen av systemet. Trots den positiva feedbacken då uppgifterna utfördes uppges olika utvecklingsmöjligheter i frågorna gällande användning samt funktionalitet. Nedan i figur 6.11-6.14 presenteras förbättringarna utifrån testpersonernas synpunkter.



(a) *(b)*

Figur 6.1. Prototyp a. Filtreringsmöjlighet. b. Popup filtreringsvy

I figur 6.1 (a) ovan visas den förbättrade designen på en filtreringsmöjlighet för att endast visa lediga rum, filtrera på en viss byggnad eller antal platser i rummet. I figur 6.1 (b) visas filteringsvalen i ett popupfönster.



(a) *(b)*

Figur 6.2. Prototyp a. Information om rum. b. Välj tid i rum

Figur 6.2 (a) ovan visar att varje rum utökades med mer information så vid klick visas ett popupfönster med mer information om vad som finns i rummet. Figur 6.2 (b) visar vyn för att kunna boka ett specifikt rum som gjordes tydligare med hjälp av tidslinjer och en röd markering för att se den nuvarande tiden ifall rummet ska bokas samma dag.

6.3 Enhetstest och Integrationstest

Det skapade datorprogrammets korrekta funktion säkerställs genom att genomföra enhetstester och integrationstester. Testplanen visas nedan och för varje test beskrivs uppgiften som ska genomföras och programmets förväntade beteende. Programvaran Postman har använts för att genomföra de olika testerna. Notera att testplanen innefattar båda typer av test.

6.3.1 Testplan

Skapa bokning

1. Skapa bokning utan begränsning – förväntar meddelande om godkänd bokning.

2. Skapa bokning där någon person ej får boka rummet vid den givna starttiden – förväntar meddelande om ej godkänd bokning.

3. Skapa bokning där någon person har för många bokningar i det givna tidsintervallet – förväntar meddelande om ej godkänd bokning.

Redigera bokning

4. Redigera bokning utan begränsning – förväntar meddelande om godkänd redigering.

5. Redigera bokning som börjar om mindre än 30 minuter – förväntar meddelande om ej möjligt att redigera.

6. Redigera bokning så att den bryter mot antalet möjliga bokningar i tidsintervall – förväntar meddelande om ej möjligt att redigera.

Ta bort bokning

7. Ta bort bokning som börjar om mer än 30 minuter – bokningen ska enbart försvinna från systemet.

8. Ta bort bokning som börjat – begränsning i avseende starttid för att boka det givna rummet ska aktiveras för samtliga personer registrerade på bokningen.

9. Ta bort bokning för att påtvinga uppdatering av begränsning av starttid i förhållande till det givna rummet – begränsningen av starttid

för det givna rummet ska uppdateras för samtliga personer registrerade på bokningen.

Samtliga tester har genomförts flera gånger under programutvecklingen och de testresultat som visas här har tillkommit från en artefakt som testats upprepade gånger. Testresultaten redovisas i tabellform och varje rad i tabellen är numrerad i enlighet med det test i testplanen den motsvarar. Ett godkänt i högra kolumnen innebär att programmet uppvisade korrekt beetende och ett icke-godkänt innebär att det inte gjorde det. Se tabell 6.3 nedan för resultat.

Tabell 6.3: Resultattabell enhetstest och integrationstest

Testnummer	**Resultat**
1	Godkänt
2	Godkänt
3	Godkänt
4	Godkänt
5	Godkänt
6	Godkänt
7	Godkänt
8	Godkänt
9	Godkänt

7 Analys

Här presenteras en kritisk granskning av lösningsförslagen. Resultaten av enkätundersökning, användbarhetstest, enhetstest och integrationstest samt design science diskuteras.

7.1 Enkätundersökning

För att få en tydligare och mer sanningsenlig frekvens hade ett större antal respondenter önskats. Då enkäten spreds genom e-post når den ut till ett stort antal respondenter men många uteblivna svar har dock märkts av samt enstaka svar som varit mindre givande i vår mätning. Trots ett begränsat antal inkomna svar har det framkommit att de flesta använder den nuvarande tjänsten men är inte helt övertygad om att den är lättanvänd. Det är också 40 av totalt 52 personer som upplever att det inte är enkelt att boka via sin mobila enhet vilket är ett av målen i projektet. En bidragande faktor till varierande synpunkter kan till stor del bero på det varierande användandet av tjänsten men också att som användare krävs inte den bredd på funktionaliteten som föreslås. Efter analys av resultatet har det trots den varierande frekvensen framgått att de flesta funktioner skulle uppskattas av användarna för att göra tjänsten mer användarvänlig och det är en bekräftelse på att det finns ett missnöje i den nuvarande tjänsten.

Som tidigare nämndes uppstod uteblivna svar vilket medför att undersökningens resultat hade kunna sett annorlunda ut ifall fler respondenter hade deltagit. Resultatet återspeglar de respondenter som deltagit och visar ett förväntat resultat utifrån studiens problematisering.
När det handlar om undersökningens validitet anses den vara hög då de frågor som ställs handlar om användarens egna uppfattningar om tjänsten. Enkätundersökningens frågor är utformade på ett sätt så dess reliabilitet, alltså tillförlitlighet inte ska påverkas och de som deltar kan inte heller ta del av andras svar och på så sätt minska reliabiliteten.[2]

7.2 Användbarhetstest

Valet av testpersoner vid användbarhetstestet var viktigt då det finns varierande tekniska kunskaper hos användare och det kan resultera i olika synpunkter på användargränssnittet. Som testledare och observatör var det positivt att se hur den varierande kunskapen inte hade någon påverkan då prototypen hos alla testpersoner upplevdes enkel. Då testerna utfördes på en prototyp och inte på ett färdigt system upplevde testpersonerna inte fördröjning som exempelvis kan uppstå vid nätverksanrop. Det är något att ha i åtanke vid utveckling och användbarhetstest på ett färdigt system.

7.3 Analys av lösningsförslagen

7.3.1 Begränsa redigering av bokning

I beskrivningen av algoritmen som kontrollerar om det är tillåtet att redigera en bokning och i dess implementation var det antaget att begränsningen ska tillträda från och med 30 minuter före bokningens början. Detta val gjordes utan vidare undersökning om vad som är lämpligt och kan korrigeras om nödvändigt.

Vid eventuell användning av algoritmen för att implementera funktionalitet för att redigera en bokning är det viktigt att datan som bearbetas tillhör den oredigerade instansen av bokningen som ska redigeras. Detta beror på att om instansen som bearbetas är den redigerade varianten av bokningen finns det risk att kontrollen misslyckas. I synnerhet så misslyckas algoritmen att ge ett korrekt resultat om den avsedda bokningen ej tillåts redigeras men den redigerade instansen skiljer sig från originalet i att start och sluttiden har skjutis framåt i tiden. För att implementera funktionalitet för borttagning av bokningar sker rimligtvis samma koll om redigering är tillåten och detta är möjligt att göra med enbart en instans så problemet som uppstod med flera instanser existerar ej.

7.3.2 Begränsa antalet bokningar i tidsintervall

Här har ett antagande gjorts i förhållande till antalet bokningar som får existera i ett givet tidsintervall. Det bestämdes att fyra stycken bokningar är en lämplig siffra men ytterligare undersökning kan bidra med kunskap om det finns lämpligare siffror.

En korrigering som kan tänkas göras på algoritmen är att den tar hänsyn till vilken stad bokningarna sker i. I nuläget tas ingen hänsyn till stad när räkningen sker men det kan vara önskvärt att göra det om det är så att användarna av systemet huvudsakligen befinner sig i en stad. Gör användarna det däremot är det överflödigt att ta hänsyn till stad i vissa fall eftersom en given användare för det mesta kommer boka rum i endast en stad. En annan möjlighet är att utöver att ta hänsyn till stad att även ta hänsyn till byggnaderna där rummen befinner sig.

För att algoritmen ska vara användbar i praktiken krävs vissa åtgärder i förhållande till hur tidsintervallen bestäms. Eftersom beräkningen innefattar en jämförelse gentemot start-och-sluttid kan begränsningen kringgås om åtgärder inte tas. Det händer i synnerhet om en begränsning existerar för en användare som bidrar med en bokning vars tidsintervall skiljer sig det minsta

lilla från tidsintervallet som begränsningen avser. Till exempel om en ny bokning med en starttid eller sluttid som skiljer sig med endast 5 minuter från de redan existerande bokningarna tillkommer. För att åtgärda detta kan algoritmen korrigeras så att den inte gör en direkt jämförelse gentemot starttiden och sluttiden utan att den istället kontrollerar om tiderna för nya bokningar ligger inom tidsintervallen för befintliga bokningar.

7.3.3 Gruppbokning

Möjligheten att skapa gruppbokningar uppnås rimligtvis genom att för varje bokning som tillkommer skapa en koppling till samtliga personer som ska ingå i bokningen.

I detta sammanhang bör systemet vid eventuell borttagning av en bokning begränsa varje person registrerad på bokningen så att de kan boka det avsedda rummet från och med den borttagna bokningens slut. Motsvarande bör systemet kontrollera att varje person registrerad på en ny bokning får boka rummet vid den givna starttiden. Genomförs dessa kontroller inte för samtliga personer löses inte problemen de avser lösa. Om det enbart är bokaren av rummet som kontrolleras kan gruppen boka om rummet när det inte ska vara möjligt genom att låta någon annan gruppmedlem agera bokare. Det kan även vara så att olika personer har olika rättigheter vad gäller starttid för bokning av ett givet rum och vissa kan vara tillåtna att boka rummet när som helst. Sker det dock ett försök att boka ett rum där minst en person ej är tillåten att boka kommer försöket misslyckas. Samma argument kan göras för begränsning av antal bokningar i tidsintervall.

7.4 Enhetstester och integrationstester

Samtliga tester som specificeras i testplanen har genomförts på det utvecklade datorprogrammet upprepade gånger. Testresultaten påverkas av detta i att tidigare versioner av datorprogrammet som ej klarat ett givet test kan ha korrigerats för att klara testet. Anledningen till att alla tester resulterade i godkänt beror alltså på att korrigeringar av datorprogrammet har skett iterativt för att uppnå godkänt.

Testmetodiken som valts är begränsad i att det på grund av mänskliga faktorer inte är möjligt att testa alla möjliga scenarion som kan uppstå i programmet. Svårigheten ligger i att identifiera dessa scenarion. Det finns även en risk att den som genomför testerna antingen tolkar uppgiften att genomföra eller det förväntade beteendet av programmet fel. Resultaten av testerna avser att påvisa till vilken grad det utvecklade programmet fungerar korrekt. Svårigheten i att välja uppgifter för testerna kan leda till att

icke-korrekt beteende ej upptäcks. Vidare är tillämpningen av testmetodiken inte rekommenderad för större projekt. Resonemanget lyder att för större projekt tillkommer betydligt fler uppgifter att utvärdera. Det sätt som metodiken tillämpats under projektets gång innebär mycket "manuellt" arbete och för större projekt bedöms tidsåtgången för enbart testning vara för stor.

7.5 Design Science

Första fasen i Design Science innefattar problemidentifiering och specifikation av kriterier för utvärdering av den framställda artefakten. Under projektets gång har det visat sig att det är komplicerat att framställa sådana kriterier i samband med problemidentifikation. De kriterier som framställdes i början av projektet har behövts korrigeras efter att insikter om problem med dem tillkommit under faserna där problemen löses. Därav bör de kriterier som framställs i första fasen vara öppna för korrigeringar om insikter om implementation kräver det.

8 Slutsats

Här dras slutsatser om huruvida projektets mål uppnåtts, om frågeställningen besvarats och framtida arbete som kan genomföras.

8.1 Mål och resultat

Syftet med projektet var att genomföra undersökningar för möjliga förbättringar till Mittuniversitetets rumsreservationstjänst. Möjliga förbättringsområden har identifierats och en enkätundersökning genomfördes för att ta reda på hur studenter och medarbetare vid Mittuniversitetet uppfattar systemet samt de nya funktionaliteter som kan tänkas ingå. Resultatet av undersökningen visade sig stämma överens med vår uppfattning av det nuvarande systemet och påvisade även att våra föreslagna lösningar är lämpliga mål till ett förbättrat system.
En prototyp av användargränssnittet framställdes för att kunna genomföra användbarhetstester som utvärderar utseende samt utformning. Målet var att ta reda på hur ett lättanvänt användargränssnitt bör utformas och resultaten av testerna tyder på att det skapade användargränssnittet är lättanvänt. De som genomförde testerna fick komma med synpunkter och dessa har tagits i åtanke för vidare arbete.

Efter undersökningar har en metod för att begränsa antalet bokningar en användare kan göra framställts. Den baseras på att en användare inte kan registrera fler än ett bestämt antal bokningar som tar plats under samma tidsintervall. Framtida arbete kan dock krävas för att korrigera metoden samt utöka den så att beräkningen tar hänsyn till stad.

Funktionalitet för gruppbokningar har framställts och vidare förbättringar krävs ej här. Modellering av databasen som specificerats under projektet bidrar direkt med denna funktionalitet.

Den tillämpade testmetodiken har påvisat att det utvecklade systemet fungerar korrekt. Det finns däremot inte en garanti på avsaknad av fel eftersom det inte är praktiskt att identifiera och testa alla möjliga scenarier som kan uppstå i systemet.

8.2 Återkoppling till frågeställning

För att återkoppla till studiens frågeställning: - *Hur konstrueras rumsreservationssystem som säkerställer att bokade rum nyttjas samt att det inte uppstår några missförstånd om vilka rum som är bokade?*

Vi kan konstatera att goda resultat har uppnåtts men de är inte tillräckliga för att fullständigt lösa problemet med att bokade rum inte nyttjas. Den metod som framställts för att begränsa möjligheten till redigering av bokningar kan lösa problemet delvis genom att påverka användare till att vara mer noggranna med att ta bort bokningar som inte används. Förutsatt att den används för att begränsa möjligheten till redigering av bokningar samt att låsa rum för bokning till en viss starttid. En ytterligare undersökning som kan göras är att ta reda på om denna påverkan finns. Vidare anses ett aviseringssystem vara en lämplig lösning av problemet med ogiltiga bokningar. På så vis kan en användare få en avisering att en bokning snart börjar och har användaren avsikten att inte använda det bokade rummet utgör aviseringen en påminnelse att avboka rummet.

Missförstånden i förhållande till bokningar identifierades att uppstå då bokningar tas bort efter att de börjat men före 30-minutersregeln gjort bokningen ogiltig. Anledningen till att göra detta är att kringgå 30-minutersregeln. Problemet har i princip lösts genom att låta konsekvensen för en sådan borttagning vara att det avsedda rummet ej kan bokas om förrän vid den borttagna bokningens sluttid. Ett undantagsfall där missförstånd fortfarande kan uppstå är när en bokning varar 30 minuter eller mindre. Om en bokning exempelvis börjar 08:00 och slutar 08:30 är det möjligt att ta bort bokningen 08:25 och boka om rummet så att bokningen börjar 08:30. Är det fallet att bokningar i genomsnitt varar längre än 30 minuter förekommer undantagsfallet inte särskilt ofta och för att bli av med det helt och hållet kan en minsta tidslängd på bokningar påtvingas. Vidare har resultaten av de genomförda användbarhetstesterna bland annat visat att möjligheten att enbart visa lediga rum är önskvärd. Ett rimligt antagande är att detta inte medför några missförstånd om vilka rum som är bokade.

8.3 Framtida arbete

Detta kapitel innehåller arbete för framtida utveckling av tjänsten. Autentisering är en nödvändig del och framtida arbete innefattar att utforma en bra lösning för det. Framtida arbete och utveckling är även att koppla ihop de funktionerna som utvecklats till ett användargränssnitt och möjliggöra hämtning från databas. Det är även önskvärt att korrigera algoritmen som

används för att begränsa antalet bokningar i tidsintervall så att dess syfte inte kan kringgås genom att genomföra små förändringar på ett givet intervall.
Ett fortsatt arbete skulle även kräva mer insamlat material i form av information från studenter och medarbetare som använder tjänsten för att skapa en tjänst som uppfyller alla möjliga funktionskrav.

Referenser

[1] Git-scm, "Information om Github", https://git-scm.com/about, Hämtad 2020-05-29

[2] H. Sharp, Interaction Design Beyond human-computer interaction. 5th ed. Indianapolis, Indiana: Wiley., 2019

[3] Coyette, A., Kieffer, S., Vanderdonckt, J., (2007). Multi-fidelity Prototyping of User Interfaces, i: Baranauskas, C., Palanque, P., Abascal, J., Barbosa, S.D.J. (Eds.), Human-Computer Interaction – INTERACT 2007. Springer Berlin Heidelberg, Berlin, Heidelberg, ss. 150–164

[4] XML, "Varför XML", http://www.xml.se/xml/varfor.html, Hämtad 2020-05-29.

[5] JSON, "Information om JSON", https://www.json.org/json-en.html, Hämtad 2020-05-29.

[6] Oracle, "Information om glassfish-server", https://www.oracle.com/middleware/technologies/glassfish-server.html, Hämtad 2020-05-29.

[7] J Kurose, K Ross, Computer Networking: A Top-Down Approach. 6 uppl. Pearson, 2013

[8] Postman, "Sending-the-first-request", https://learning.postman.com/docs/postman/launching-postman/sending-the-first-request, Hämtad 2020-05-27

[9] Doxygen, "Index", https://www.doxygen.nl/index.html, Hämtad 2020-06-15

[10] Vaishnavi, V., Kuechler, W., and Petter, S. (Eds.) (2004/19). "Design Science Research in Information Systems" January 20, 2004 (created in 2004 and updated until 2015 by Vaishnavi, V. and Kuechler, W.); last updated (by Vaishnavi, V. and Petter, S.), June 30, 2019. URL: http://www.desrist.org/design-research-in-information-systems/

[11] Android Developer, "Testing Fundamentals", https://developer.android.com/training/testing/fundamentals, Hämtad 2020-06-15

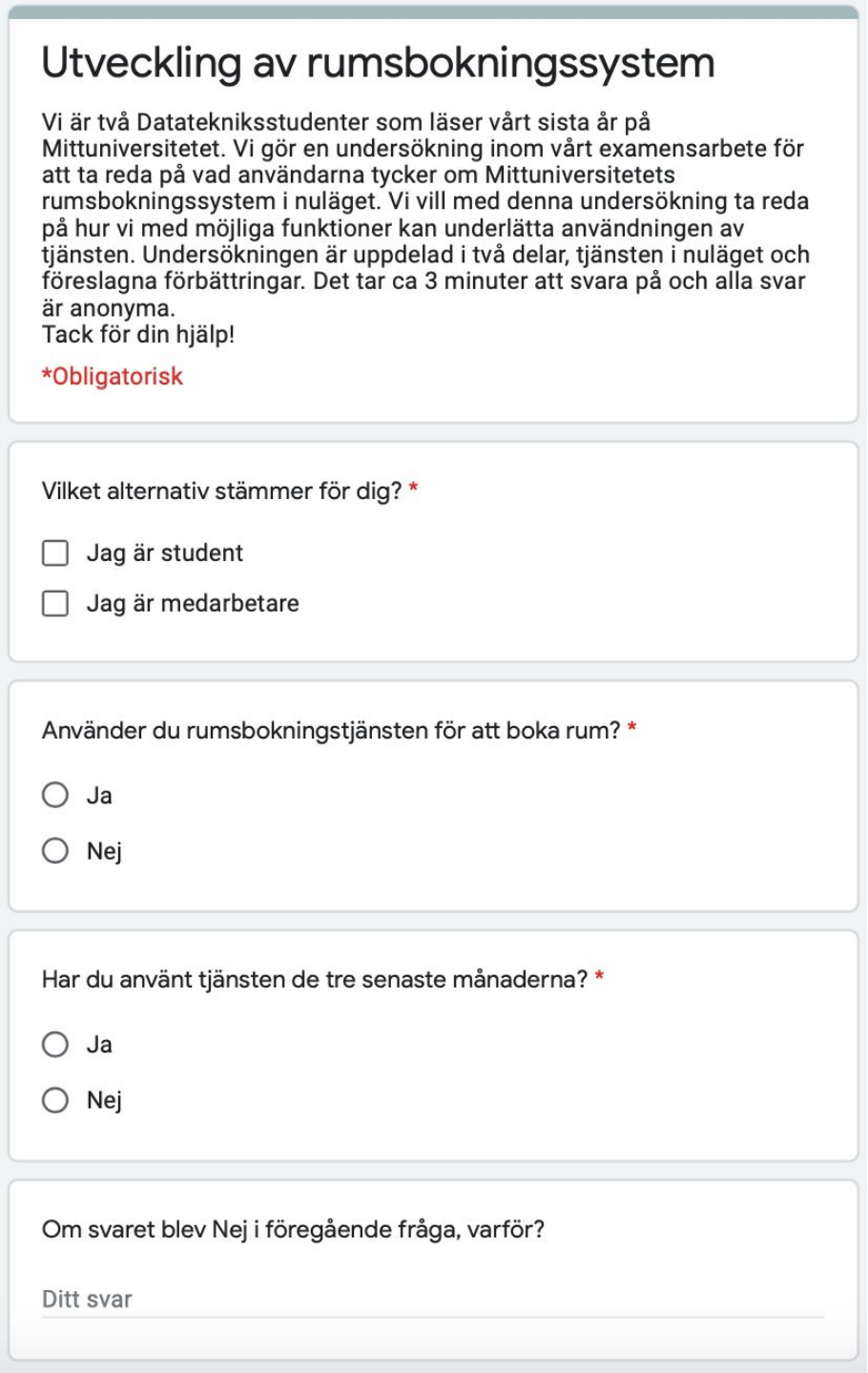
[12] U.Eriksson, Kravhantering för IT-system, 2:6. uppl. Studentlitteratur AB, Lund

[13] Github, "Hänvisning till källkod", https://github.com/DT099G-room-reservation/room_reservation_dataservice, Hämtad 2020-06-04

[14] Github, "Dokumentation av källkod", https://dt099g-room-reservation.github.io/html/index.html, Hämtad 2020-06-04

Bilaga A: Mall för enkätundersökning

I bilaga A nedan framgår enkätundersökningen som användes i projektet.



Utveckling av rumsbokningssystem

Vi är två Datateknikstudenter som läser vårt sista år på Mittuniversitetet. Vi gör en undersökning inom vårt examensarbete för att ta reda på vad användarna tycker om Mittuniversitetets rumsbokningssystem i nuläget. Vi vill med denna undersökning ta reda på hur vi med möjliga funktioner kan underlätta användningen av tjänsten. Undersökningen är uppdelad i två delar, tjänsten i nuläget och föreslagna förbättringar. Det tar ca 3 minuter att svara på och alla svar är anonyma.
Tack för din hjälp!

*Obligatorisk

Vilket alternativ stämmer för dig? *

- [ ] Jag är student
- [ ] Jag är medarbetare

Använder du rumsbokningstjänsten för att boka rum? *

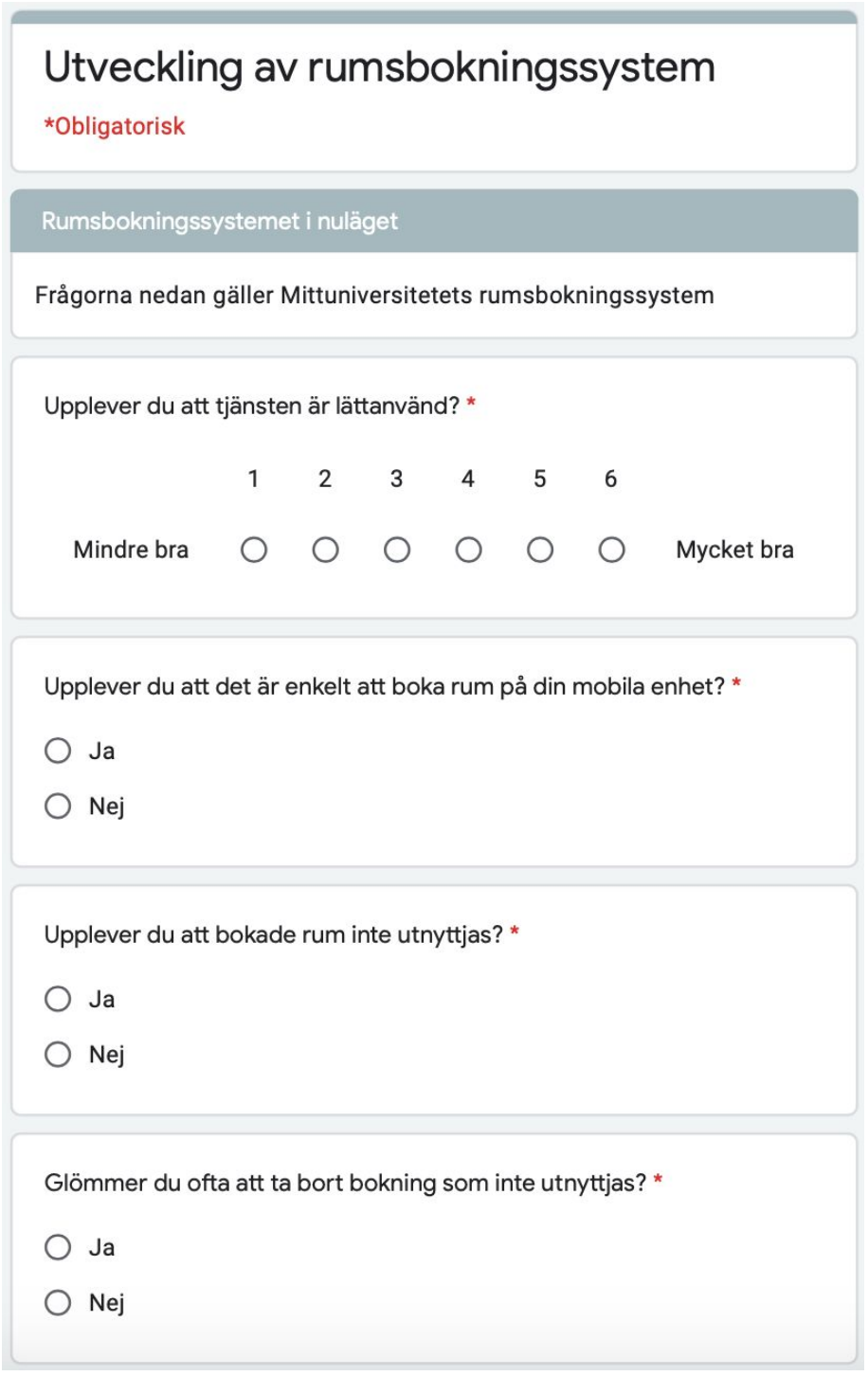
- ○ Ja
- ○ Nej

Har du använt tjänsten de tre senaste månaderna? *

- ○ Ja
- ○ Nej

Om svaret blev Nej i föregående fråga, varför?

Ditt svar



Utveckling av rumsbokningssystem

*Obligatorisk

Rumsbokningssystemet i nuläget

Frågorna nedan gäller Mittuniversitetets rumsbokningssystem

Upplever du att tjänsten är lättanvänd? *

	1	2	3	4	5	6	
Mindre bra	○	○	○	○	○	○	Mycket bra

Upplever du att det är enkelt att boka rum på din mobila enhet? *

- ○ Ja
- ○ Nej

Upplever du att bokade rum inte utnyttjas? *

- ○ Ja
- ○ Nej

Glömmer du ofta att ta bort bokning som inte utnyttjas? *

- ○ Ja
- ○ Nej

Utveckling av rumsbokningssystem

*Obligatorisk

Utvecklat rumsbokningssystem

Frågorna nedan gäller utökning av Mittuniversitetets rumsbokningssystem med fler funktioner

Funktion för att skapa en gruppbokning? *

När du bokar ett rum kan du ange fler användarnamn för att vara fler som står för bokningen. Då kan samtliga få meddelande om en aktuell bokning.

	1	2	3	4	5	6	
Inte viktigt	○	○	○	○	○	○	Mycket viktigt

En bokning som inte utnyttjas tas bort från systemet? *

Detta gör det möjligt för övriga att boka rummet

	1	2	3	4	5	6	
Inte viktigt	○	○	○	○	○	○	Mycket viktigt

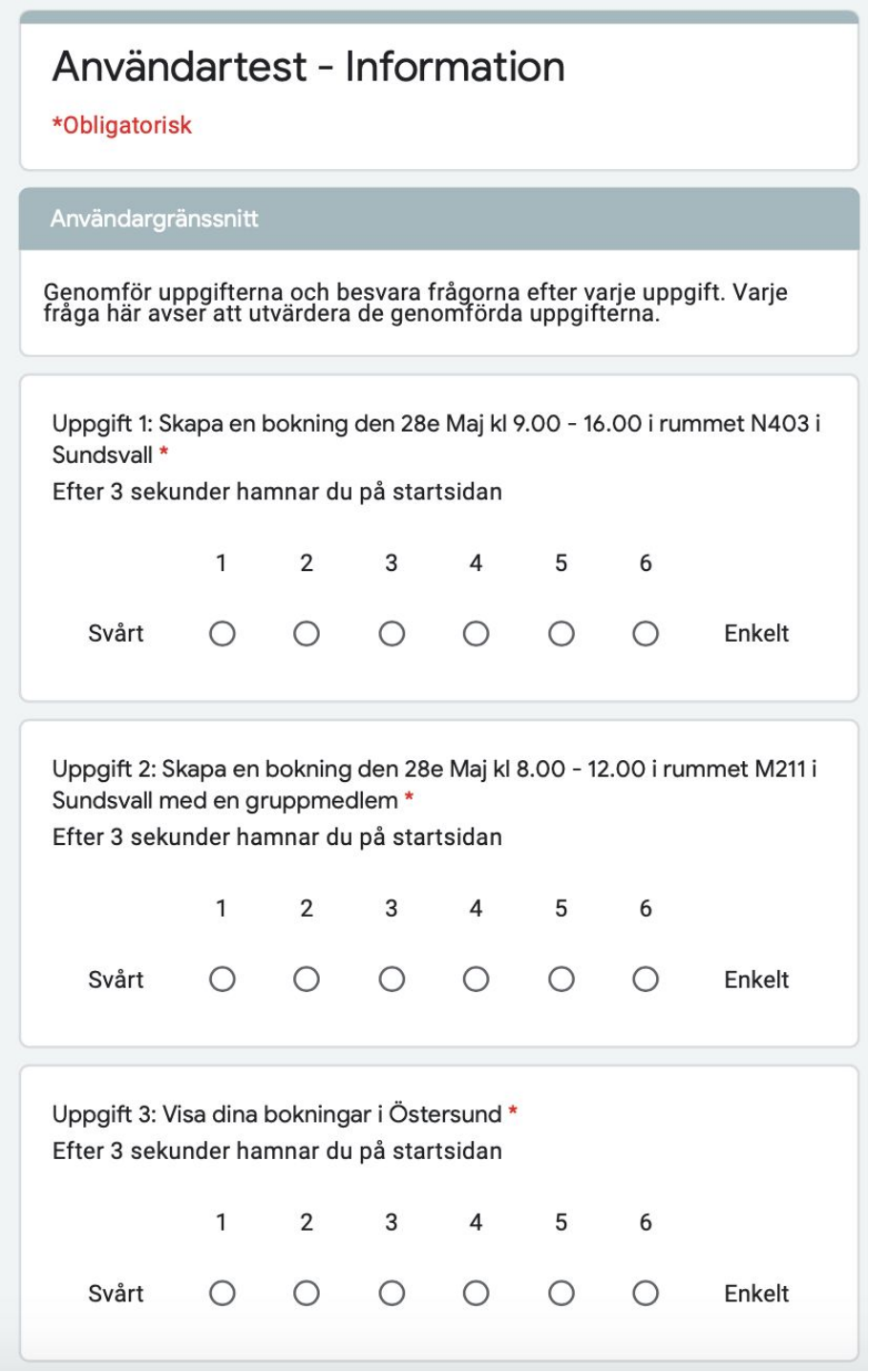
Möjligheten att omboka rum begränsas? *

Att ändra en bokning nära inpå dess början är inte möjligt

	1	2	3	4	5	6	
Inte viktigt	○	○	○	○	○	○	Mycket viktigt

Bilaga B: Mall för användbarhetstest

I bilaga B nedan framgår formuläret som användes i projektets användbarhetstest.



Användartest - Information

*Obligatorisk

Användargränssnitt

Genomför uppgifterna och besvara frågorna efter varje uppgift. Varje fråga här avser att utvärdera de genomförda uppgifterna.

Uppgift 1: Skapa en bokning den 28e Maj kl 9.00 - 16.00 i rummet N403 i Sundsvall *

Efter 3 sekunder hamnar du på startsidan

	1	2	3	4	5	6	
Svårt	○	○	○	○	○	○	Enkelt

Uppgift 2: Skapa en bokning den 28e Maj kl 8.00 - 12.00 i rummet M211 i Sundsvall med en gruppmedlem *

Efter 3 sekunder hamnar du på startsidan

	1	2	3	4	5	6	
Svårt	○	○	○	○	○	○	Enkelt

Uppgift 3: Visa dina bokningar i Östersund *

Efter 3 sekunder hamnar du på startsidan

	1	2	3	4	5	6	
Svårt	○	○	○	○	○	○	Enkelt

Uppgift 4: Ta bort en bokning som inträffar “Torsdag 28 Maj” i Sundsvall *

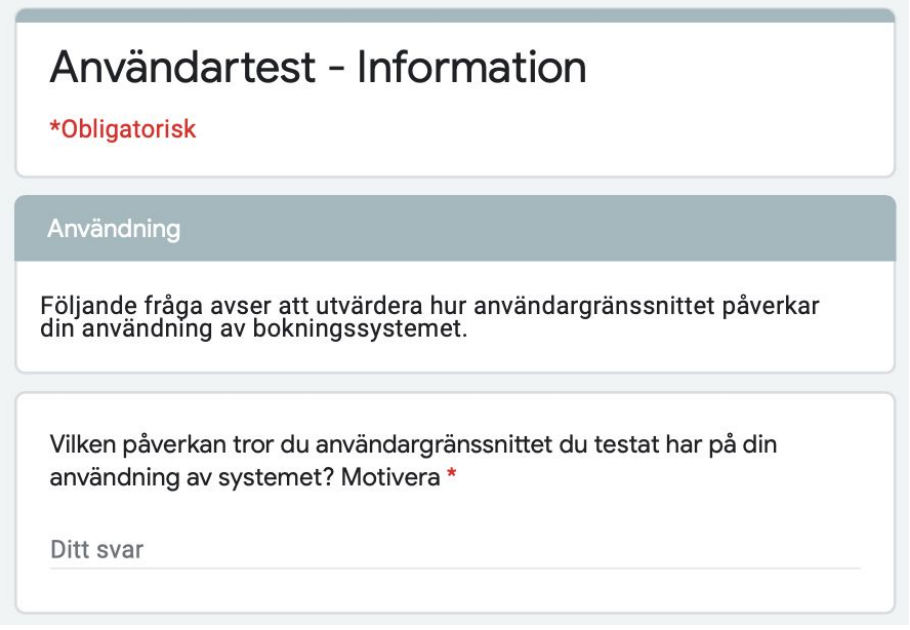
Efter 3 sekunder hamnar du på startsidan

	1	2	3	4	5	6	
Svårt	○	○	○	○	○	○	Enkelt

Uppgift 5: Redigera bokningen i Sundsvall som inträffar "Fredag 22 maj" genom att lägga till en medlem *

Efter 3 sekunder hamnar du på startsidan

	1	2	3	4	5	6	
Svårt	○	○	○	○	○	○	Enkelt



Användartest - Information

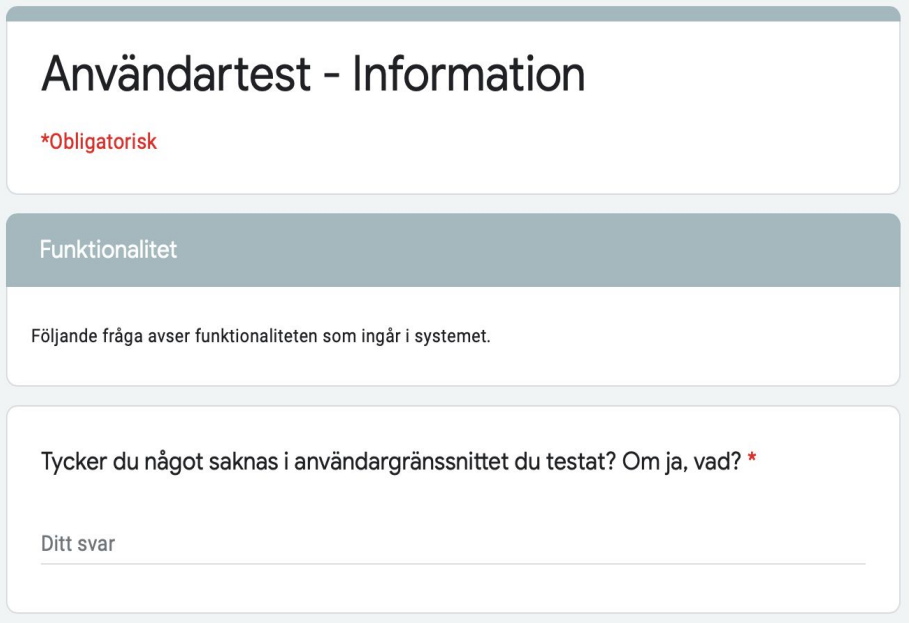
*Obligatorisk

Användning

Följande fråga avser att utvärdera hur användargränssnittet påverkar din användning av bokningssystemet.

Vilken påverkan tror du användargränssnittet du testat har på din användning av systemet? Motivera *

Ditt svar



Användartest - Information

*Obligatorisk

Funktionalitet

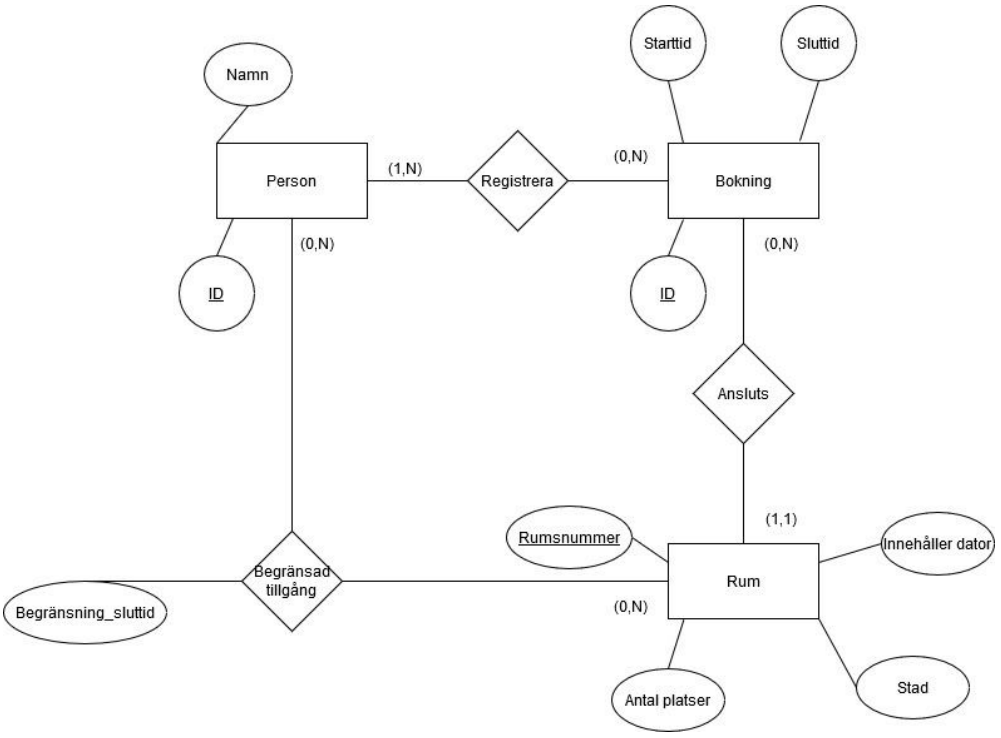
Följande fråga avser funktionaliteten som ingår i systemet.

Tycker du något saknas i användargränssnittet du testat? Om ja, vad? *

Ditt svar

Bilaga C: Datamodellering

Bilden nedan visar ER-diagram



Bilden nedan visar basrelationer

Person

ID	Namn

Bokning

ID	Starttid	Sluttid	Rumsnummer(FK)

Rum

Rumsnummer	Antal_platser	Dator?	Stad

Registrerade_bokningar

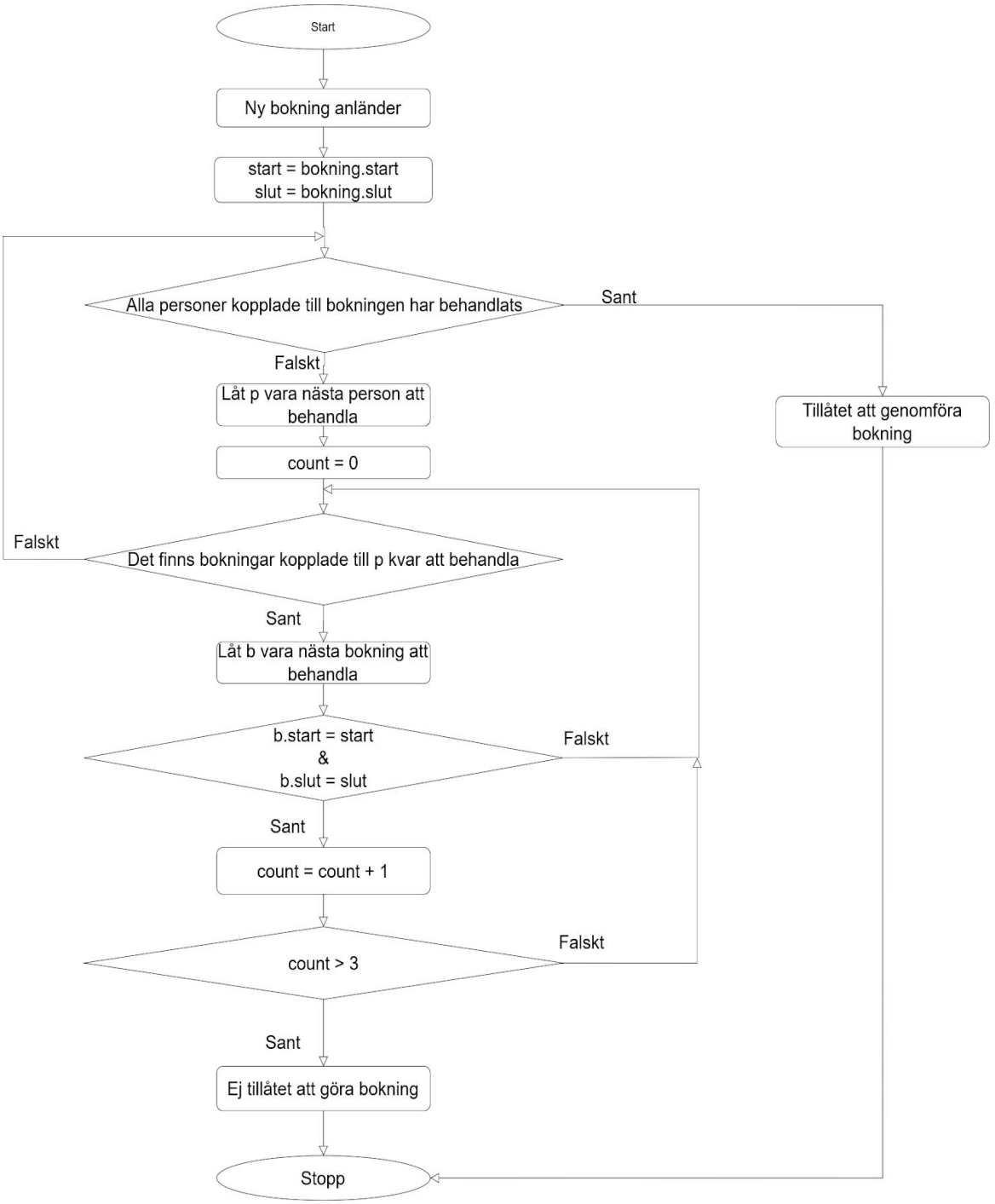
PersonID(FK)	BokningID(FK)

Begränsad_tillgång_till_rum

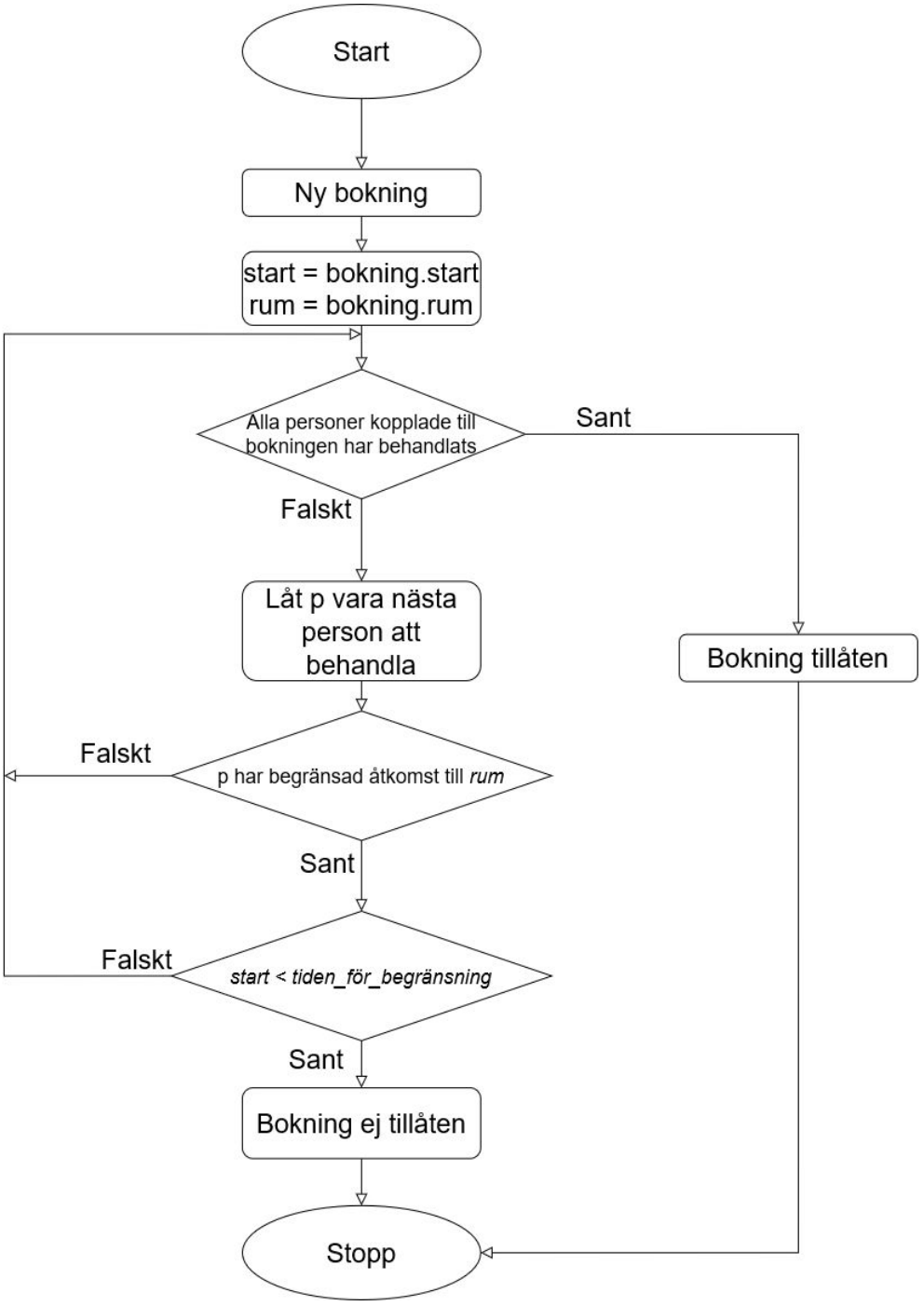
PersonID(FK)	Rumsnummer(FK)	Sluttid

Bilaga D: Flödesscheman

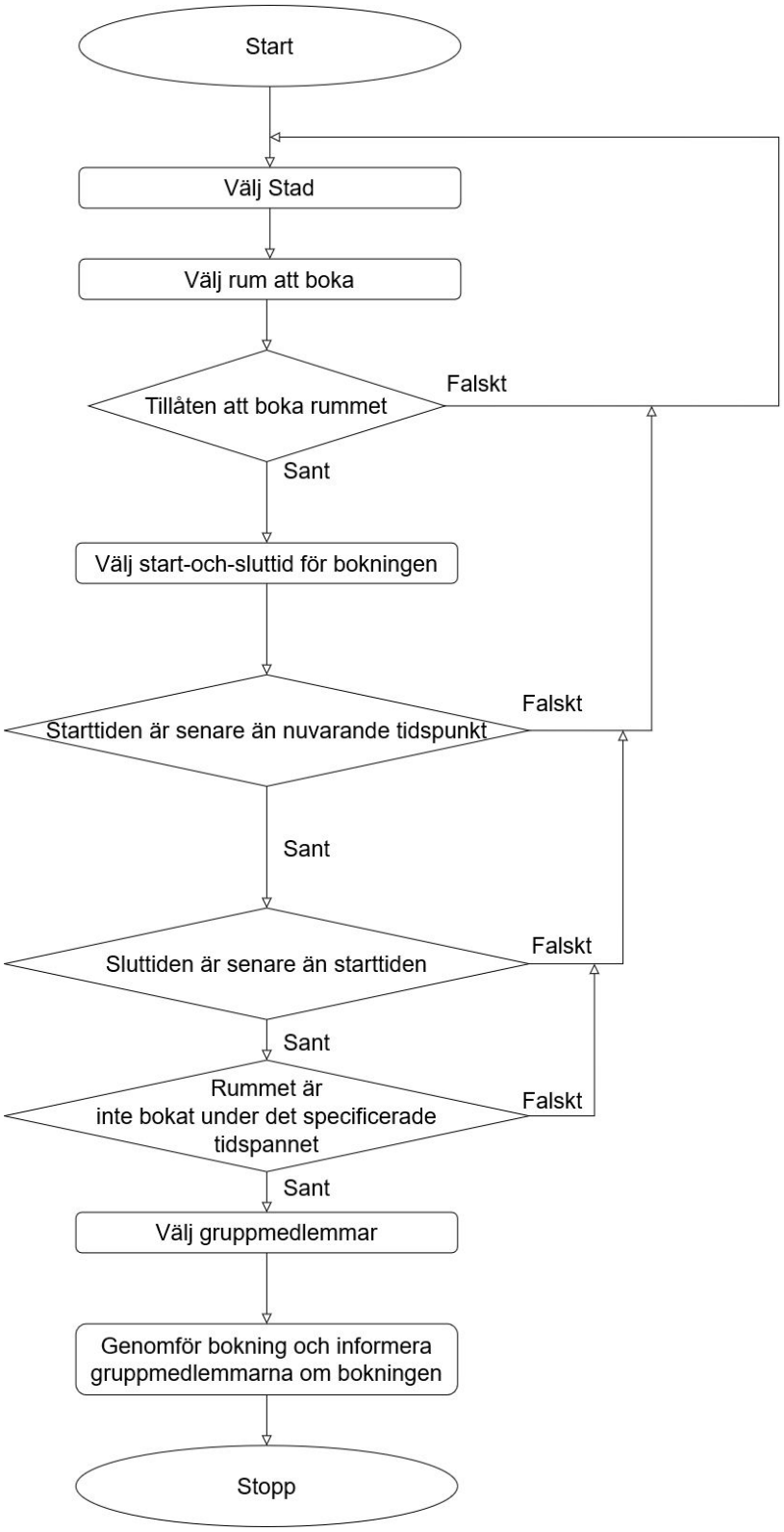
Flödesschema: Begränsa antal bokningar i tidsintervall



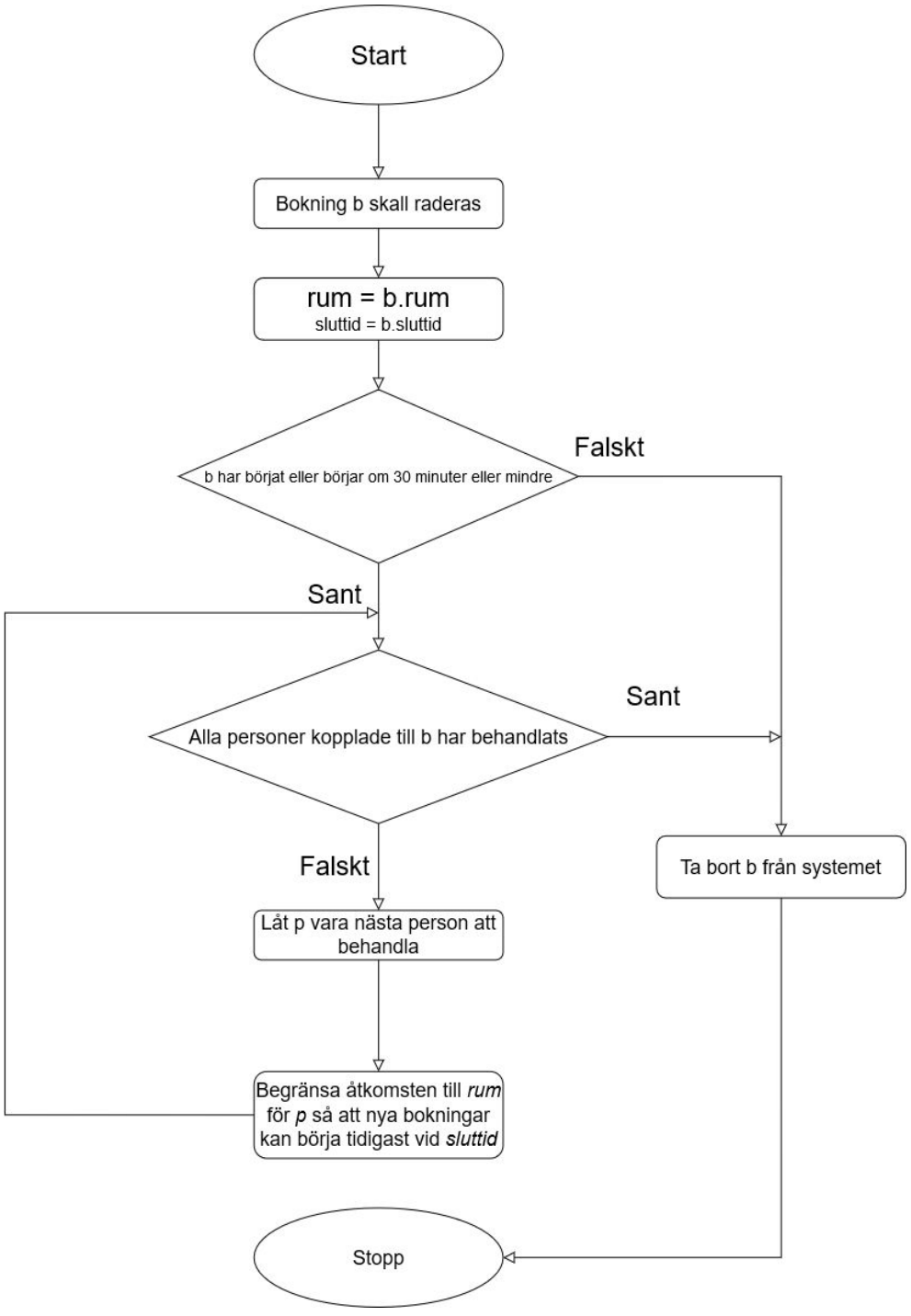
Flödesschema: Begränsa starttid för bokning



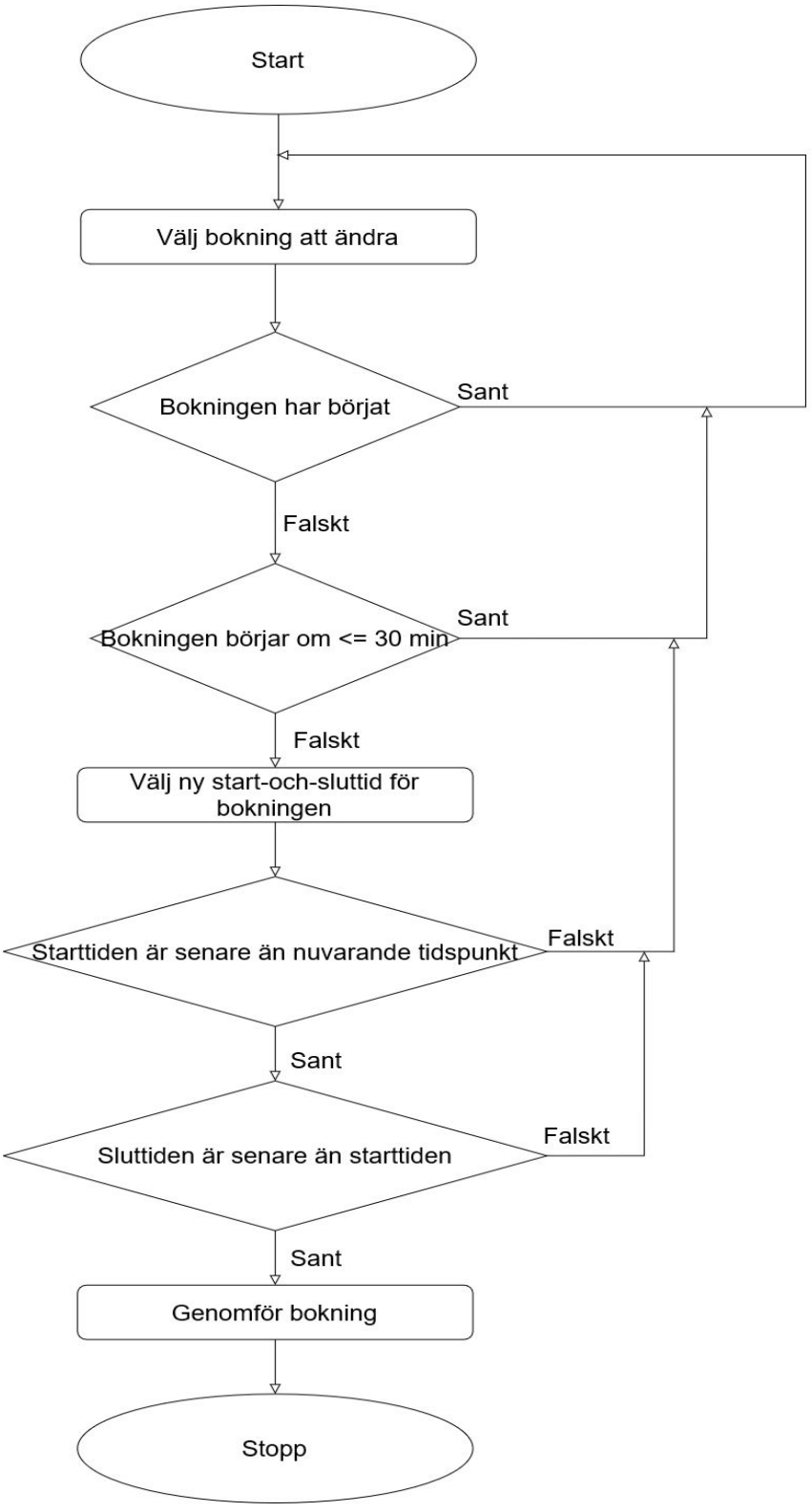
Flödesschema: Genomför bokning



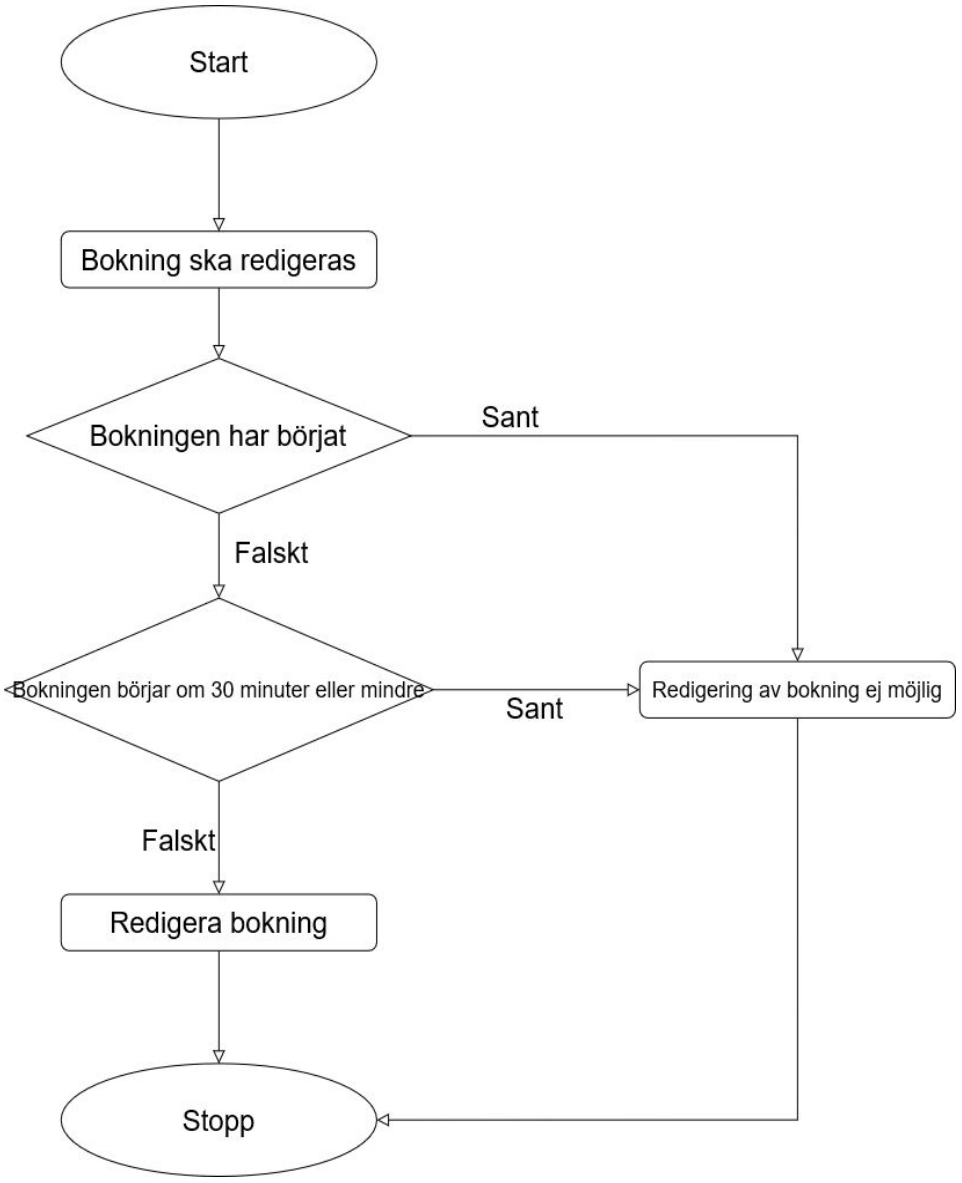
Flödesschema: Borttagning av bokning



Flödesschema: Ombokning



Flödesschema: Redigera bokning tillåtet



Bilaga E: Källkod

Implementation: Lägg till bokning

```
@POST
@Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
@Override
public Response create(Reservation entity) {
    Boolean allowed = allowed_reservation(entity);
    DateTime start = new DateTime(entity.getStartTime());
    DateTime end = new DateTime(entity.getEndTime());
    Collection<Person> people = entity.getPersonCollection();
    Boolean tooMany = tooManyReservationsInInterval(people, start, end);
    if (allowed && !tooMany) {
        for (Person p : entity.getPersonCollection()) {
            Person per = getEntityManager().find(Person.class, p.getPersonId());
            per.getReservationCollection().add(entity);
        }
        return super.create(entity);
    }
    return Response.status(Response.Status.FORBIDDEN)
            .entity("Not allowed to make this reservation").build();
}
```

Implementation: Redigera bokning

```
@PUT
@Path("{id}")
@Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public Response edit(@PathParam("id") Integer id, Reservation entity) {
    Reservation reservation = super.find(id);
    Boolean isStrict = modificationIsStrict(reservation);
    Collection<Person> people = entity.getPersonCollection();
    DateTime start = new DateTime(entity.getStartTime());
    DateTime end = new DateTime(entity.getEndTime());
    Boolean tooMany = tooManyReservationsInInterval(people, start, end);
    if (!isStrict && !tooMany) {
        super.edit(entity);
        return Response.ok().build();
    }
    return Response.status(Response.Status.FORBIDDEN)
            .entity("Not allowed to edit this reservation").build();
}
```

Implementation: Ta bort bokning

```
@DELETE
@Path("{id}")
public void remove(@PathParam("id") Integer id) {
    Reservation reservation = super.find(id);
    Boolean isStrict = modificationIsStrict(reservation);
    Collection<Person> people = reservation.getPersonCollection();
    if (isStrict) {
        for (Person p : people) {
            LimitedRoomAccessPK pk
                    = new LimitedRoomAccessPK(p.getPersonId(),
                            reservation.getRoomNumber().getRoomNumber());
            LimitedRoomAccess access
                    = new LimitedRoomAccess(pk, reservation.getEndTime());
            access.setPerson(p);
            access.setRoom(reservation.getRoomNumber());
            getEntityManager().merge(access);
        }
    }
    for (Person p : people) {
        p.getReservationCollection().remove(reservation);
    }
    super.remove(reservation);
}
```

Implementation: Begränsa starttid för bokningar

```
private boolean allowed_reservation(Reservation entity) {
    Collection<Person> people = entity.getPersonCollection();
    CriteriaBuilder cb = getEntityManager().getCriteriaBuilder();
    javax.persistence.criteria.CriteriaQuery cq = cb.createQuery();
    Root<LimitedRoomAccess> r = cq.from(LimitedRoomAccess.class);
    for (Person p : people) {
        cq.select(r);
        cq.where(
                cb.equal(r.get(LimitedRoomAccess_.person), p),
                cb.equal(r.get(LimitedRoomAccess_.room), entity.getRoomNumber())
        );
        List<LimitedRoomAccess> lst
                = getEntityManager().createQuery(cq).getResultList();
        for (LimitedRoomAccess a : lst) {
            DateTime end = new DateTime(a.getEndTime());
            DateTime startTime = new DateTime(entity.getStartTime());
            if (startTime.isBefore(end)) {
                return false;
            }
        }
    }
    return true;
}
```

Implementation: Kontrollera om bokning kan redigeras

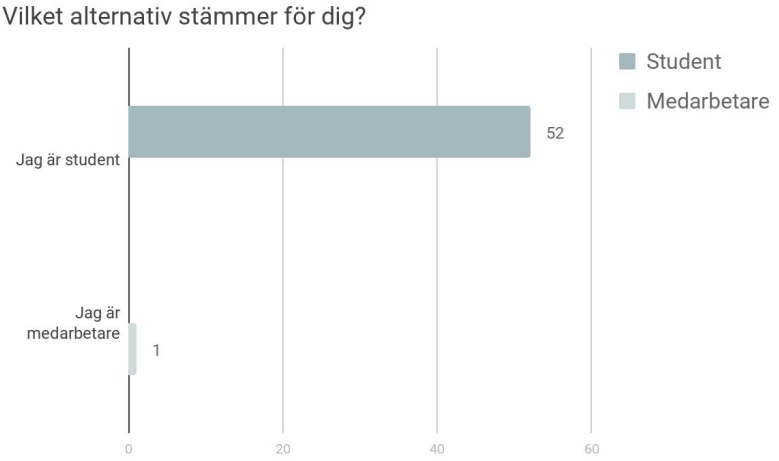
```
private Boolean modificationIsStrict(Reservation reservation) {
    DateTime reservation_start = new DateTime(reservation.getStartTime());

    //reservation has not started yet
    if (reservation_start.isAfterNow()) {
        DateTime now = new DateTime();
        Interval difference = new Interval(now, reservation_start);
        long duration = difference.toDurationMillis();

        //note: 1800000 milliseconds is 30 minutes
        //reservation begins in 30 minutes or less
        if (duration <= 1800000) {
            return true;
        }
    } else if (reservation_start.isBeforeNow()) {
        return true;
    }
    return false;
}
```

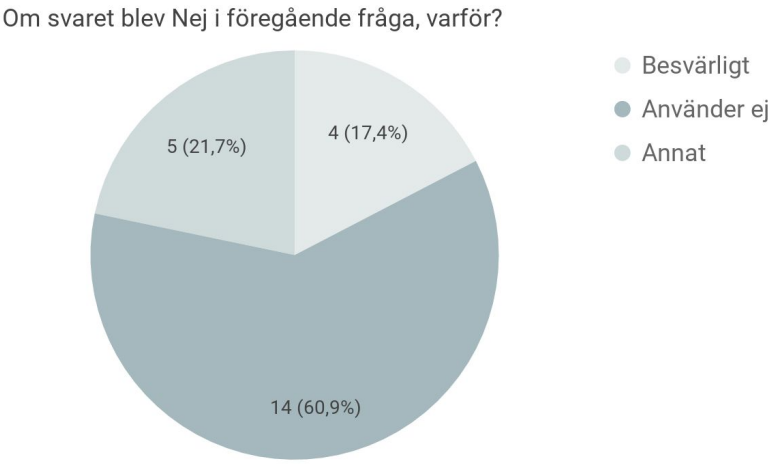
Implementation: Räkna bokningar i tidsintervall

```
private Boolean tooManyReservationsInInterval(Collection<Person> people,
        DateTime intervalStart, DateTime intervalEnd) {
    for (Person p : people) {
        int count = 0;
        Person temp = getEntityManager().find(Person.class, p.getPersonId());
        Collection<Reservation> reservations
                = temp.getReservationCollection();
        for (Reservation r : reservations) {
            DateTime reservationStart = new DateTime(r.getStartTime());
            DateTime reservationEnd = new DateTime(r.getEndTime());
            if (intervalStart.equals(reservationStart)
                    && intervalEnd.equals(reservationEnd)) {
                count += 1;
            }
        }
        if (count > 3) {
            return true;
        }
    }
    return false;
}
```

Bilaga F: Svar för enkätundersökning

I bilaga F nedan presenteras respondenterna som deltog i enkätundersökningens svar.





Om svaret blev Nej i föregående fråga, varför?
23 svar

Svarade ja
Inte varit aktuellt då kurserna varit praktiska på lab och skrivandet gått att göra hemma.
Finns sällan tider och är generellt ett krångligt system
De bästa rummen är alltid bokade långt i förväg.
Har till stor del varit hemma den senaste tiden, i den tidigare kursen skötte en gruppmedlem bokning. En mer generell anledning för varför jag inte brukar boka rum är för att många rum tenderar att bokas upp för att sedan inte användas vilket gör det lättare att bara leta efter ett ledigt rum.
Jobbar hemifrån
För att jag flyttat till annan stad

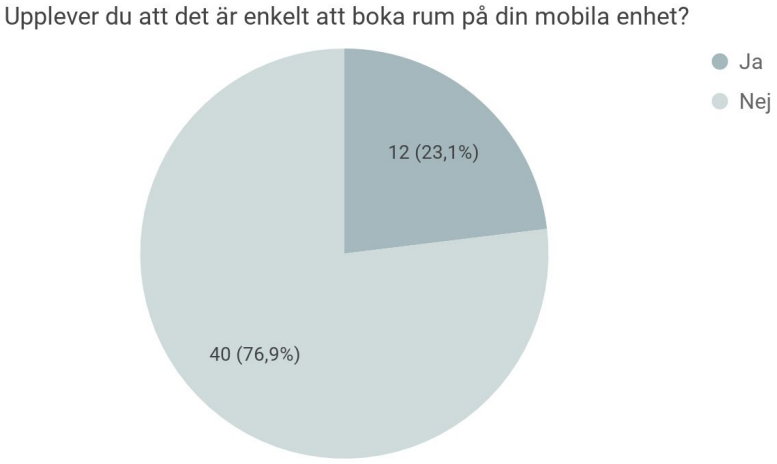
Om svaret blev Nej i föregående fråga, varför?
23 svar

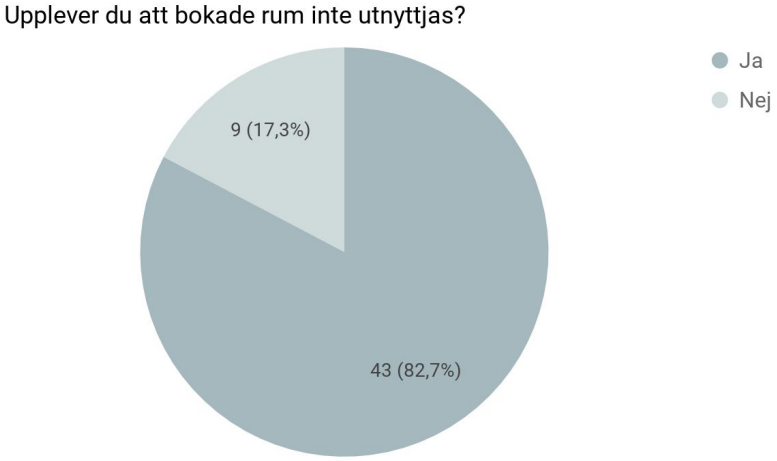
- Har tillgång till grupprum tillägnat det program jag går.
- Sitter ensam och pluggar så behöver ej rummen i denna period
- Därför
- Har inte bokat pga covid-19
- För jag aldrig behövt använda mig av ett rum för att kunna studera.
- Corona
- Inget behov
- Gjort exjobb och inte varit på skolan
- Jag har inte varit på campus.

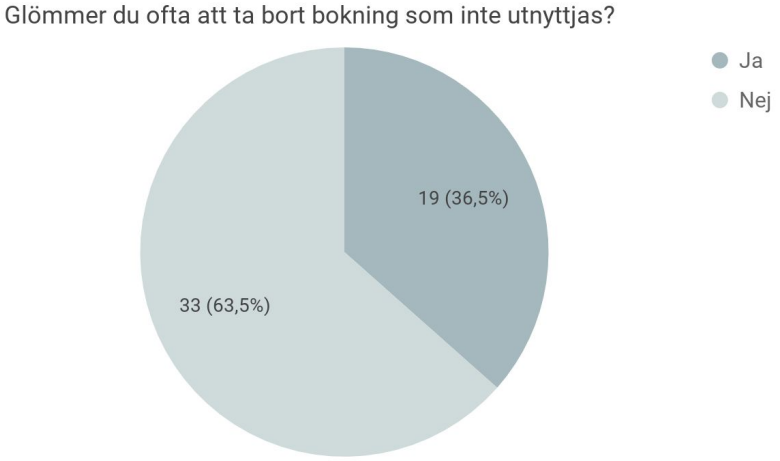
Om svaret blev Nej i föregående fråga, varför?
23 svar

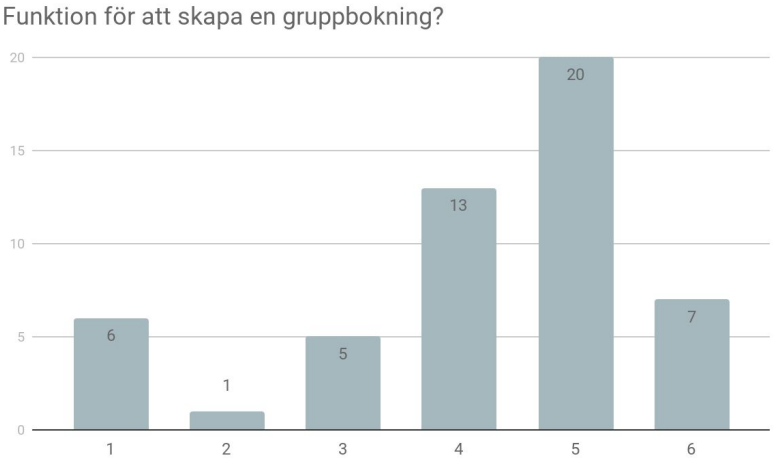
- Jag upplever det besvärligt då det inte går att boka via mobilen, jag kan ofta komma på att jag behöver boka ett grupprum då jag inte sitter vid en dator.
  När man väl bokat ett rum känner jag att det är lätt att glömma bort sin bokade tid då jag inte använder grupprummen så frekvent.
  Pga dessa två anledningar känner jag att det är mer besvärligt att boka ett grupprum än att faktiskt sitta hemma eller att sitta på en plats på skolan som inte kräver en bokning.
- Covid19
- Datorsalarna har varit tomma.
- ... Distansstudier... Arbetar hemifrån så har inte behövt boka grupprum.
- Har inte varit på skolan
- Det är i stort sett tomt i skolan, sitter sällan i grupprummen.
- Inte vart på skolan

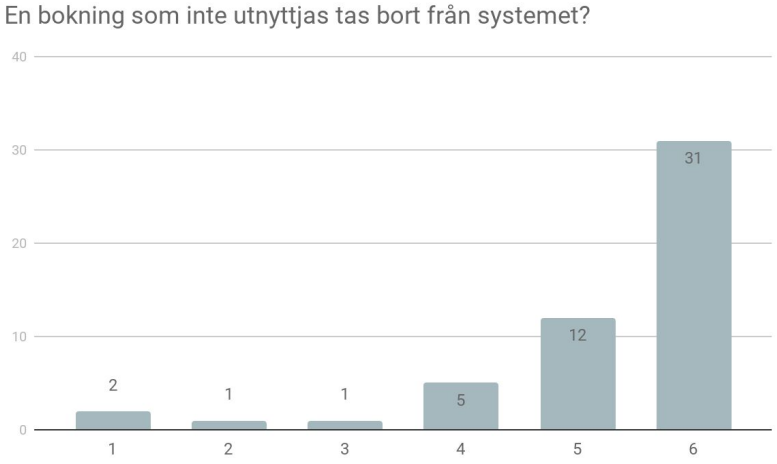

Upplever du att tjänsten är lättanvänd?
15
10
5
0
6
7
14
12
8
5
1
2
3
4
5
6


Upplever du att det är enkelt att boka rum på din mobila enhet?
Ja
Nej
12 (23,1%)
40 (76,9%)

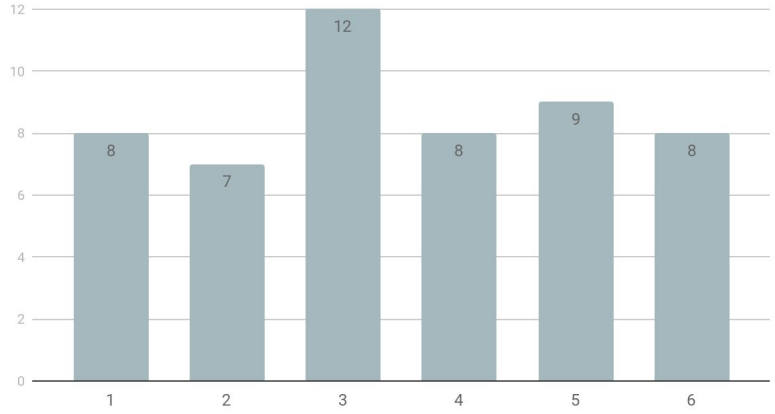

Upplever du att bokade rum inte utnyttjas?
Ja
Nej
9 (17,3%)
43 (82,7%)


Glömmer du ofta att ta bort bokning som inte utnyttjas?
Ja
Nej
19 (36,5%)
33 (63,5%)

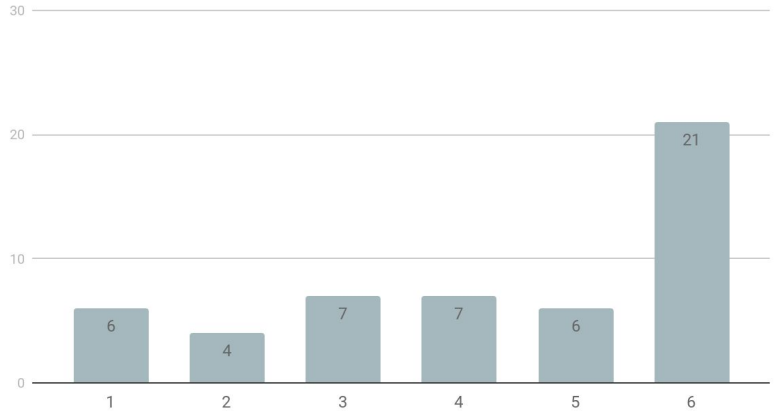

Funktion för att skapa en gruppbokning?
20
15
10
5
0
6
1
5
13
20
7
1
2
3
4
5
6


En bokning som inte utnyttjas tas bort från systemet?
40
30
20
10
0
2
1
1
5
12
31
1
2
3
4
5
6

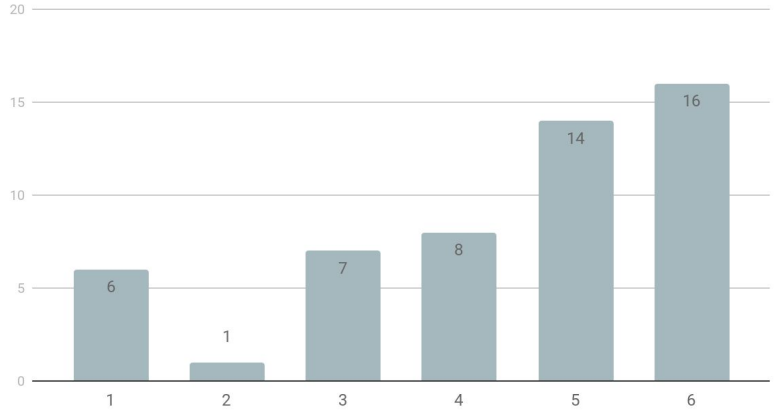
Möjligheten att omboka rum begränsas?

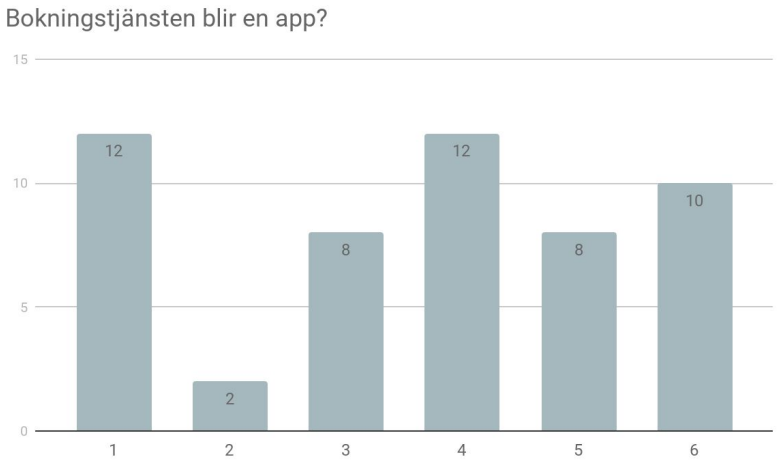


Begränsa antalet bokningar under samma tidsintervall?



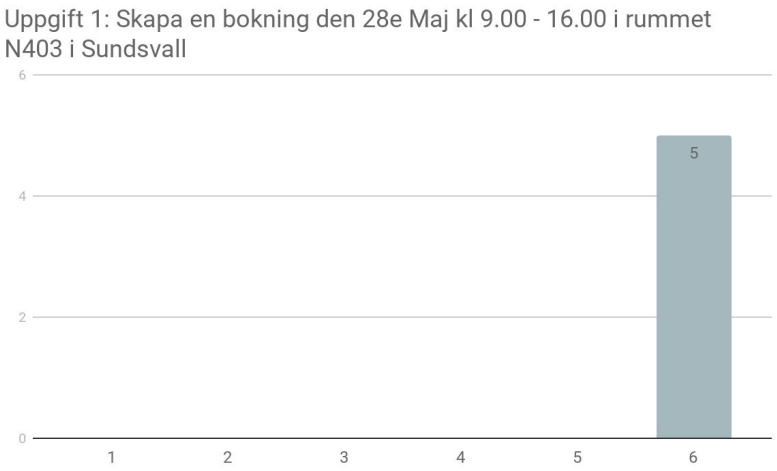
Du får en påminnelse innan bokningen börjar?

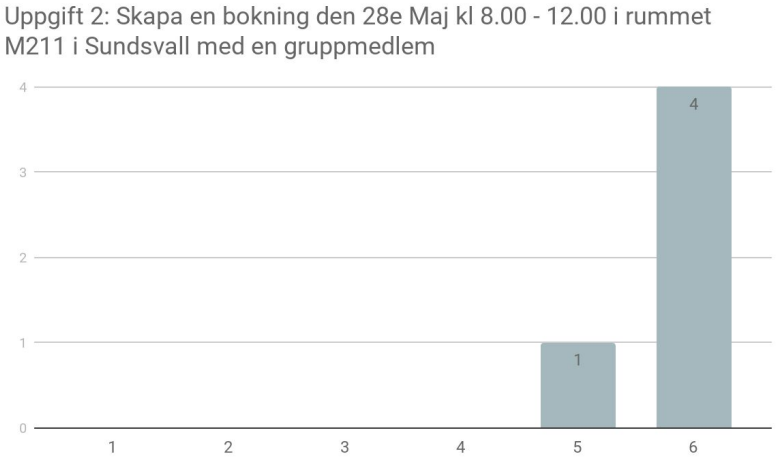



Bokningstjänsten blir en app?
15
10
5
0
12
2
8
12
8
10
1
2
3
4
5
6

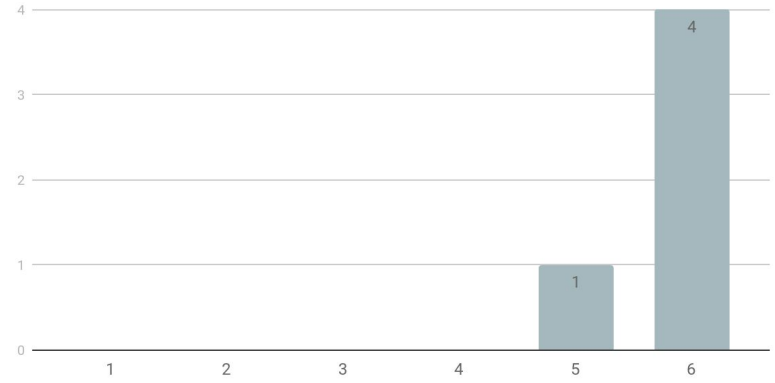
Bilaga G: Svar för användbarhetstest

I bilaga G nedan presenteras respondenterna som deltog i användbarhetstestets svar.

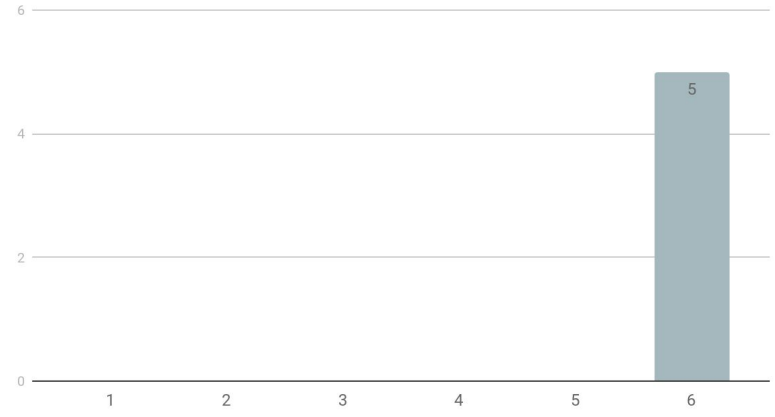




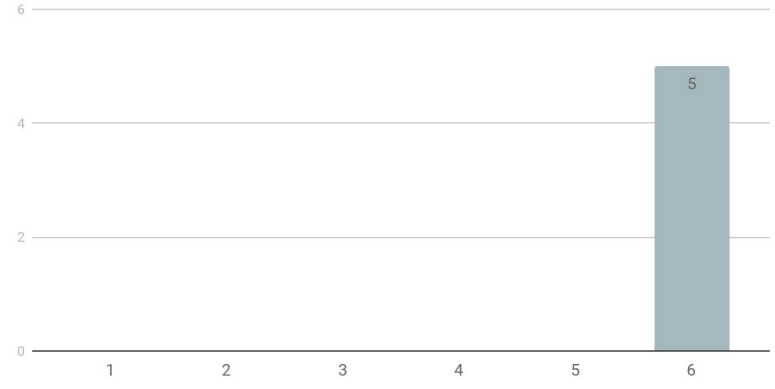
Uppgift 2: Skapa en bokning den 28e Maj kl 8.00 - 12.00 i rummet M211 i Sundsvall med en gruppmedlem



Uppgift 3: Visa dina bokningar i Östersund



Uppgift 4: Ta bort en bokning som inträffar “Torsdag 28 Maj” i Sundsvall



Användning

Vilken påverkan tror du användargränssnittet du testat har på din användning av systemet? Motivera

5 svar

- Programmet var lättöverskådligt med enkla vyer. Programmet var väl anpassat för nybörjare utan teknisk erfarenhet.
- tror att det skulle göra det enklare att boka samt boka av då det var väldigt lättanvänt
- Jag tycker att det underlättar att boka rum via en app där allt framgår tydligt, är lättbegriplig och logiskt. Kommer användas mer flitigt då bokningssystemet finns på en app i mobilen
- Jag känner att det kan underlätta bokningar som jag vill utföra på en mobil enhet då det inte går att göra det på ett effektivt sätt idag. Att kunna se tydligt och strukturerat vad jag har bokat för rum är positivt.
- Stor påverkan. Lätt att navigera sig och förstå vad som är vart. Enkla symboler. Simpel och "clean" design (bra)

Funktionalitet

Tycker du något saknas i användargränssnittet du testat? Om ja, vad?

5 svar

- Ett typ av filter som visar antalet lediga rum under en specifik önskad tidsperiod.
- nej
- Filter för att kunna sortera och enbart se vilka rum som är lediga samt mer info om vad som finns i rummen (projektor,dator,whiteboard etc)
- Tidslinjer där man ser vilka tider som är bokat för ett rum. Filter för att se t.ex. endast lediga tider, platsantal, hus, mm.
- Ingen tydlig tillbakaknapp, kugghjulet för "inställningar" är lite missvisande.