Browser based room booking system

Project:	Bachelor project
University:	ITU 2011
Advisor:	Peter Sestoft

Kristian Klarskov Marquardsen	kkma@itu.dk	220387-1611
Frederik A. Wordenskjold Noerregaard	fawn@itu.dk	140987-1727

Contents

Chapter 1

Background

1.1 Introduction

In this project we have designed a user-friendly room booking experience for the IT University[1], and provided the concept for a solution to the complex task of assigning rooms to courses. As described later, the current process in which all room assignments - both short and long term - is both tedious and inefficient. With the ever increasing focus on web-based technology, we found it only logical to bring this part of university life online in a user-friendly manner by providing a simple overview for both day-to-day tasks such as reserving a room for group work, and the larger, harder to grasp tasks such as completing the room assignment for a whole semester.

[1] IT University will henceforth be refered to as 'ITU'

1.2 Current situation

As of writing, room management at the ITU is split between the Study Administration and the Facilities Management. Within those departments there is another division of labour which all in all adds to the confusion of handling the limited rooms efficiently.

1.2.1 Facilities Management

The information desk, controlled by FM, is currently in charge of the day-to-day use of the meeting rooms. When a student or teacher wishes to reserve or use a room, the FM employee will refer to a binder with a paper-overview of current status, and if they succesfully find an available room a form is filled out, photocopied and archived.

Aside from this, another part of FM is in charge of assigning offices and workspaces to teachers, teaching assistants and researchers. This is done through a locally stored Microsoft Access database, and is handled primarily by one person.

1.2.2 Study Administration

In regards to room management, the Study Administrations biggest challenge is the planning of a semester. First step is collect data from the course database to see which courses will be held in the coming semester. When a list of courses have been compiled, a system called Tabulex[2] is used to create a draft schedule so no overlaps of teachers and courses required by several tracks[3]. The output from Tabulex is finally used to manually create the room assignments for a standard week of the semester.

The other function of the Study Administration is to assign rooms required by exams, for both the examnation itself and for the prepation-time required by some exams. Unfortunately, not all courses have ended by the time exams begin, so apart from rooms being occupied the problem of privacy and noise presents itself.

TODO:**Perhaps reformatting and elobaration**

[2]http://www.tabulex.dk/

[3]'Track' is the term used to describe a semesters worth of courses in a specific direction, eg. the 'Software Engineering' direction of the 'Software Development and Technology' MSc education

1.3 Scope

Our approach to designing this system has been primarily focused on usability and all that entails. Tedious and de-centralized management of a single resource is both time consuming and highly error-prone. To exemplify a solid solution to this problem, we have narrowed our vision to the most used area (day-to-day booking) and the most tedious area (semester planning).

The simple booking have been succesfully prototyped, whereas the semester planning have only been designed and developed theoretically, and not implemented. Doing the full scale implementation of the semester planning would of course be a possibility, but it is not crucial to prove our point of data presentation and usability and we have therefore chosen to perform all the analytical work on it instead.

As mentioned in the original problem definition, we have also elected not to include an administration panel for super-users, as it would serve little purpose without all the features of the full system.

TODO:**Limitations/inclusions of our project work goes here**

Chapter 2

Analysis

2.1 Design decisions

TODO:**How to implement, discussion of for and against our choices of platform, language, etc etc. Possibly split this in to several sections**

2.1.1 Concurrent room selection

When user x and y tries to book rooms concurrently:
Possible solutions:

- Make all rooms that fits x, unavaliable for y while x is deciding on the proper room the use. Cons: All rooms fits x, NO rooms avaliable for y. Pros: Easy to implement, no dead ends. (Extremely unlikely).

- The screen, which shows avaliable rooms, of user y should update(update rate should increase strongly) when user x makes a booking, to make sure y doesnt hit a dead end.

Cons: Increased number of requests to the server, dead ends possible. (Tradeoff between probability of dead ends vs. refresh-rate)
Pros: High usability, and responsivenes. More flexible; allows several users to have many choices at once.

How to handle dead ends?

Chapter 3

User interface

TODO:**general introduction to this section, and why it warrents a whole chapter**

3.1 Challenges

TODO:**What challenges must our user interface overcome?**

3.2 Use cases

The booking of a room have four parameters; the week of the year, the day of the week, the time of day and of course, the room. With these four in mind, the searching, browsing and organising of rooms pose some relevant questions.
To assist us in the process of deciding which parameters in a search or overview should determine the presentation, we postulate the following use cases based on number of 'locked' parameters.

Legend	
W	Week
D	Day
T	Time
R	Room
Main case	High occurance
Edge case	Rare occurance

With one parameter locked

Main cases:
W - "I need a room sometime next week"

Edge cases:
D - "I need a room on a wednesday sometime"
T - "I need a room at 12 o'clock sometime"
R - "I need Aud. 4 at some point"

With two parameters locked

Main cases:
WD - "I need a room at thursday next week"
WT - "I need a room at 9 o'clock next week"
WR - "I need Aud. 1 in 2 weeks"
DT - "I need a room mondays at 8 o'clock" (W free, so meaning a span of weeks needed here)

Edge cases:
TR - "When is Aud. 4 available at 9 o'clock?"

With three parameters locked

Main cases:
WDT - "I need a room at 12 o'clock on friday next week" (Expecting this to be the most used search of them all)
WDR - "I need room 4a14 this friday"

Edge cases:
WTR - "I need Aud. 3 at 9 o'clock next week sometime"
DTR - "I need room 2a09 on thurdays 11 o'clock" (Probably to be used along with a semi-locked W (span))

3.3 Mockups

TODO:**Add all/some of the pictures from our mockups. Currently located on github. Explan the iterations.**

3.4 Usability

TODO:**Usability discussion and documentation for findings from our tests**

3.4.1 Test af mockup1

Tasks to perform
1 - simple: "Book a room today"
2 - simple: "Book a room tomorrow"
3 - special case: room not available current day: "Book room x for use as soon as possible"
- ...

Questions to ask
- "cancel" or "back" ? : "what do you think this button does?"
- Calendar - is it obvious?

Lead-in
1. You and your group have met, and need to find a room to conduct group-work.

2. Your group has finished work today, and decided to reconvene tomorrow. You have been assigned to arrange for a place to work tomorrow.

3. Your group will be having it's exam in auditorium 3, and you wish to prepare for it in the same setting as the actual exam.

11 - 11.20: Jakob 27 DMD 2. semester

1. Find a room
Trykker på Find me a Room siger trykker vel her. Vælger tid fra og til.

2. Rum i morgen
Trykker på find me a specific room. Finder hurtigt ud af at han nok skulle have valgt vha. kalenderen.

3. Eksamen i auditorie3 en anden dag.
Trykker på auditorie3 på kortet. Trykker på another day.

4. Find rum næste fredag.
Find me a specific room. Vælger "projektor", og er nice. Spørger om man skal vælge tid her.

5. Find et lokale til mere end 50 mennesker på en onsdag.
Find me a room, cancel, find me a specific room. Task failure. Går tilbage til kortet. Trykke på et af auditorierne. Siger at han nok i stedet ville trykke på kalenderen.

6. Komsammen for førsteårsstuderende.
Trykker på aud1 på kortet. Bruger kalenderen til at vælge datoer, for at se ledigheden.

Jon, 23, SWU 6. semester

1. Find me a room, OK.

2. Bruger kalendervælgeren til at finde datoenm find me a room, OK.

3. Vælger dato gennem kalenderen, find me a specific room, OK.

4. Som 3.

5. Find me a room. Cancel. Trykker på w på kalenderen. Task failure, specific room. Vælger "auditorie" og ikke capacity. OK.

6. Trykker på aud1 på kortet. Trykker på datoen, trykker på start-tid og slut-tid. Trykker på book.

3.5 Proposed design

TODO:**Explain and illustrate the proposed design**

Chapter 4

Implementation

4.1 Datastructures

TODO:**Introduction to this section**

4.1.1 Datamodel

TODO:**Explain our database design and datamodel**

4.1.2 Algorithms

TODO:**Outline the important algorithms in our program, and why they are awesome**

4.2 Server-Client relationship

TODO:**Explain the design; what we do, and what lift handles for us.**

4.3 Lift

TODO:**This section probably doesn't belong here, but it needs to be somewhere.**

4.4 Graphical design

TODO:**Explain how the graphical design was done, and how it helps support our usability. Tell why it is one of the very vital functions of a program like this.**

Chapter 5

Conclusion

5.1 Reflection

TODO:Reflect on the process.. hurp derp

5.2 Perspective

TODO:Take a step back and look at our work as a part to a whole. Tabulex, FM/Study admin etc.

5.3 Possible improvements

TODO:If this was a commercial product, what would we change/add/fix etc.

5.4 Expansions

TODO:Consider if this section is needed. Perhaps it could fit into the above. If not, we could talk about ridding the need for tabulex, centralizing everything in our system and so forth.

5.5 Conclusion

TODO:*drumroll* Round off the paper in an elegant, humble, and yet awesomely awesome way.