Requirements Analysis Document (RAD) for 121Calendar

Mads Frederik Madsen - mfrm, Holger Stadel Borum - hstb and Paw Høvsgaard Laursen - pawh

September 19, 2014

Contents

1 Introduction

1.1 Purpose of the system

The unique purpose of our system is to be replace the old fashioned paper calendar. Our system may at first glance look like a pretty generic calender-system, due to it's very simplistic design. It is however very original, since it is the only calendar-system that does not restrict the user in some way. We have focused on making it easy to create appointments with other people, and on giving the user an almost autonomous control over his or her calendar.

1.2 Scope of the system

1.3 Objectives and success citeria the project

1.4 Definitions, acronyms and abbreveations

1.5 References

0.1 - Assignment 37 Elementary design, use cases and generic calendar functionality.
Changes: https://github.com/Madsen90/BDSA/commits/master

1.6 Overview

2 Current system

3 Proposed system

3.1 Overview

3.2 Functional Requirements

3.3 Nonfunctional Requirements

Category	Requirements
Usability:	98% of the 15-30-year-olds users should be able create, delete, edit appointments and account without prior knowledge, reading or education.
Reliability:	- Crashes/loss of connection must not cause loss of neither account- or appointment information, none-submitted data may be lost. - It should always be possible to access the server, when the client has Internet connection. - Crashes should be rare - less than 1% of operations made by a user may lead to a crash.
Performance:	- The system shuold be scalable - there should only be a hardware limit to the number of appointments or accounts in the system. - The calendar should load fast - there should be a maximum 1-2 second delay on normal computers with 1 mbit connection. - Client should be able to run on a single core 500 MHz CPU.
Supportability:	- The system should be documented. - Updateable to new browsers and OS'.
Implementation:	- Requires internet-connection
Operation:	- None.
Legal:	- User should agree to terms of use.

3.4 System models

3.4.1 Scenarios

Scenario 1: Create Appointment: -

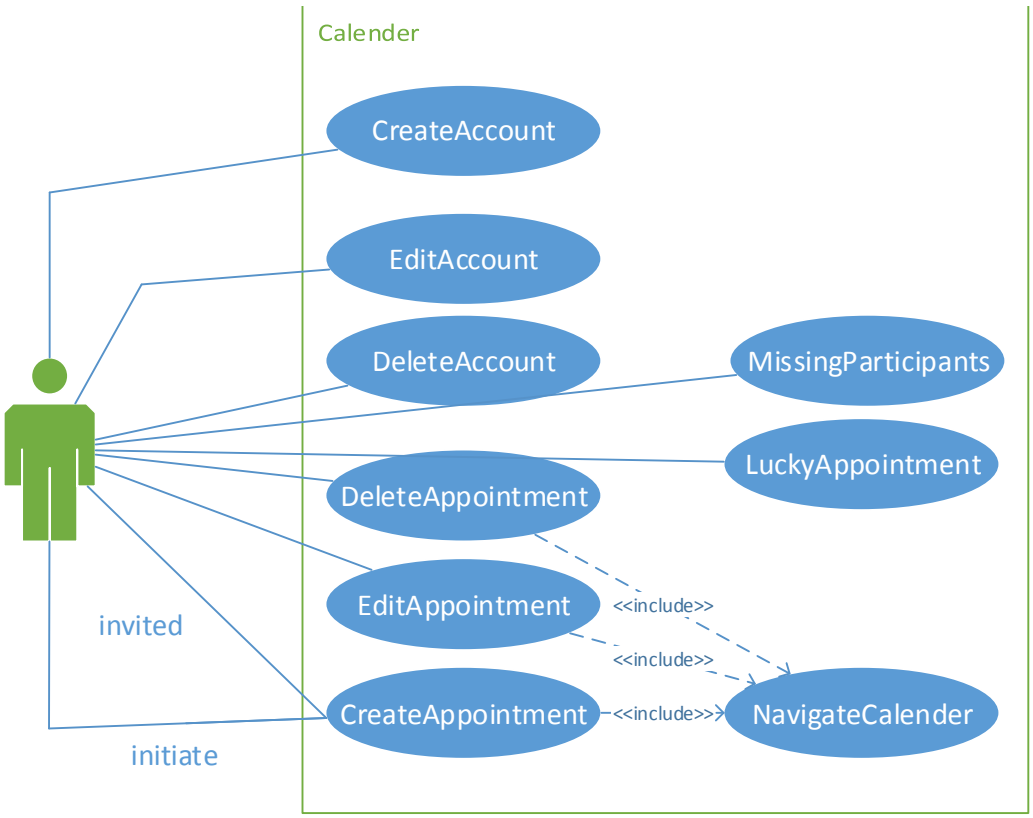
Scenario name:	HelleCreatesAppointmentWithLars
Participating actors:	Helle (Std. User) <initiator> Lars (Std. User) <participant>
Flow of events:	1. Helle wants to have a meeting with Lars on Tuesday 10:00 AM 2. Helle selects Create Appointment 3. Helle enters time, place and description. 4. Helle receives notice that appointment is successful. 5. Helle adds Lars as participant. 6. Lars receives notice that he has been added to appointment. 7. Lars reluctantly accepts invitation.

Scenario 2: Create User: -

Scenario name:	CreateHelleAsUser
Participating actors:	Helle (Std. User)
Flow of events:	1. Helle is a new employee at Statsministeriet. She needs a new calendar and accesses her calendar client 2. Helle selects Create User 3. Helle enters name and password, and confirms 4. Helle receives a notice of successful user creation.

Note: We need to figure out how to handle user-relationsship. It is unlikely everybody able to invite all users. Do we have user-friendship, groups.. etc?

3.4.2 Use case model


Calender
CreateAccount
EditAccount
DeleteAccount
MissingParticipants
LuckyAppointment
DeleteAppointment
EditAppointment
CreateAppointment
NavigateCalender
<<include>>
<<include>>
<<include>>
invited
initiate

Use case name:	CreateAppointment
Participating actors:	User wants to create an appointment with some friends and seeks additional participants. Users invited to the appointment.
Flow of events:	1. User opens Calendar. 2. Calendar shows the calendar navigation. 3. User selects the date and time of the appointment. 4. Calendar shows a form for entering description and place and participants. 5. User fills the form, adds participants through email and enters how many additional participants he seeks. 6. Calendar creates the appointment and notify participants that they have been added to the event.
Entry condition:	- User is logged in
Exit conditions:	- Appointment is changed. - User close the system. - Connection lost.
Quality requirements	- None

Use case name:	EditAppointment
Participating actors:	User wants to edit date of an appointment. Other users effected by changes.
Flow of events:	1. User opens Calendar. 2. Calendar shows the calendar navigation. 3. User selects the appointment he wants to change. 4. Calendar shows the appointment in normal mode that allows changes. 5. User selects change date of appointment. 6. Calendar shows the calendar navigation. 7. User selects a new date for the appointment. 8. Calendar updates the appointment, and notify potential participants about the changes and removes participants who sought appointment for the old date.
Entry condition:	- User is logged in
Exit conditions:	- Appointment is changed. - User close the system. - Connection lost.
Quality requirements	- None

Use case name:	DeleteAccount
Participating actors:	User wants to delete her account. Other users effected by deletion.
Flow of events:	1. User opens the program. 2. Calendar shows the calendar navigation. 3. User selects edit profile. 4. Calendar shows the profile edit. 5. User select delete account. 6. Calendar removes the user from all appointment. If the user was the only participant, the appointment is deleted. 7. Calendar deletes account, and notifies other users about changes made to their appointments.
Entry condition:	- User is logged in
Exit conditions:	- User account is deleted. - User close the system. - Connection lost.
Quality requirements	- None

Use case name:	DeleteAppointment
Participating actors:	User wants to delete an appointment. Other users effected by deletion.
Flow of events:	1. User opens Calendar. 2. Calendar shows the calendar navigation. 3. User selects the appointment he wants to delete. 4. Calendar shows the appointment in normal mode that allows changes. 5. User selects delete appointment the wished changes. 6. Calendar changes/deletes the event accordingly and notify other participants.
Entry condition:	- User is logged in.
Exit conditions:	- Appointment is deleted. - User closes the system. - Connection lost.
Quality requirements	- None

3.4.3 Object model

3.4.4 Dynamic model

3.4.5 User interface

4 Glossary

4.1 Initial Analysis Objects:

Object name	Description
User	User a person who owns an account and thereby a calender. He is able to create/delete/edit appointments.
Account	An account keeps track of user information, and may participate or create in an appointment.
Calendar	An collection of appointments for one account.
Appointment	A digital representation of an appointment between 1 or more Users. It contains a time and a place for the appointment, a title and a description for the event.
Place	A digital representation of a space, migt be none-existing.
Participant:	How an account (and thereby user). Is tied to an appointment.