Requirements Analysis Document (RAD) for 121Calendar

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

September 15, 2014

Contents

1 Introduction

1.1 Purpose of the system

We are still thinking about the unique purpose of our system. While figuring it out, we will describe a pretty generic calender-system. The main focus will be making it easy to create appointments with other people, perhaps people you don't know.

1.2 Scope of the system

1.3 Objectives and success citeria the project

1.4 Definitions, acronyms and abbreveations

1.5 References

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:	90% of the 18-25-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. - Should be accessible whenever a 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 calender 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. - The system should not conflict with Danish laws (Unrealistic to test).
Ethical:	- Users information will not be used or sold.

3.4 System models

3.4.1 Scenarios

Scenario 1: Create Appointment: -

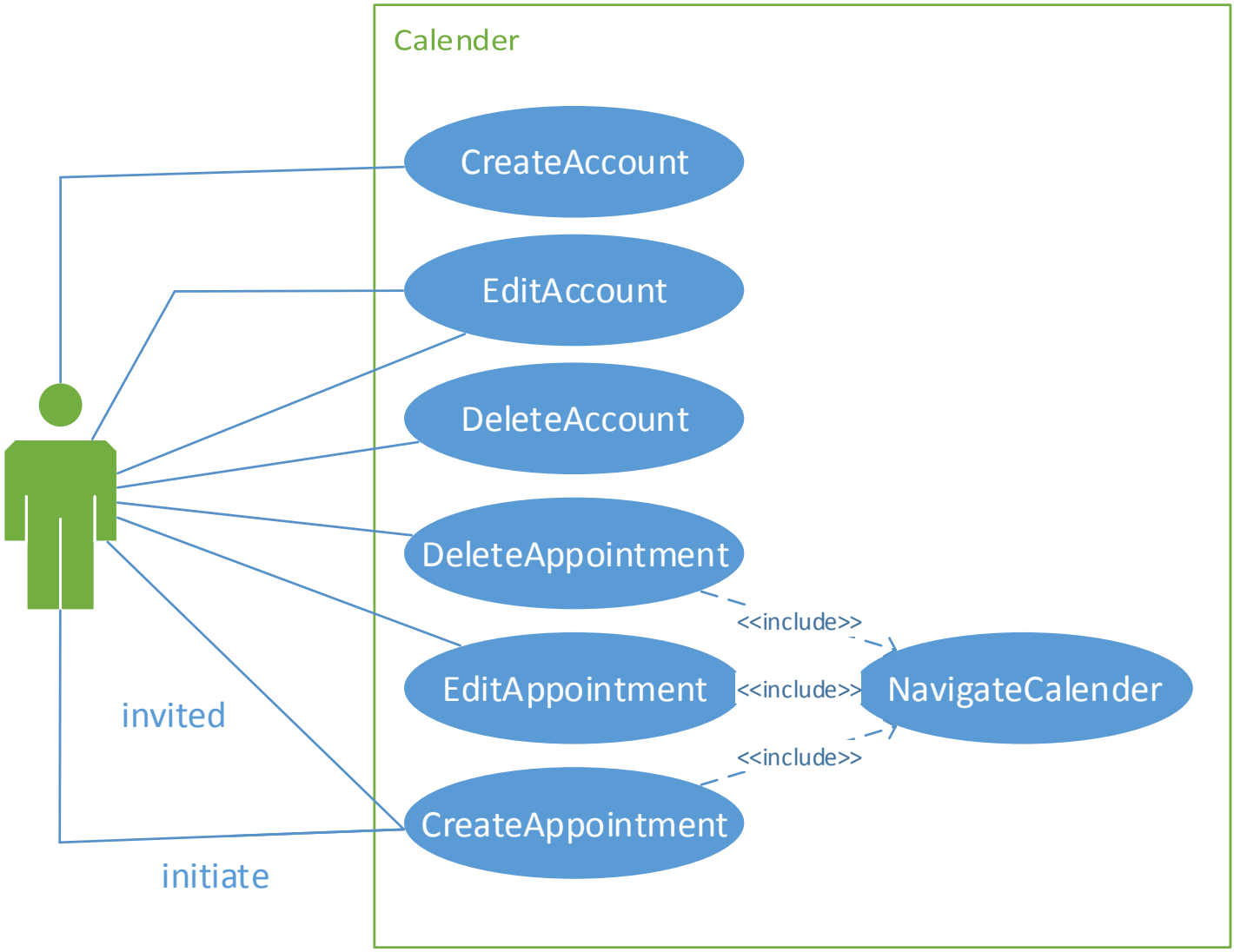
Scenario name:	HelleCreatesAppointmentWithLars
Participating actors:	Helle (Std. User) <initiator> Lars (Std. User) <participant>
Flow of events:	1. Helle wants to have a meeting with Lars on Tuesday 10:00 AM 2. Helle selects Create Appointment 3. Helle enters time, place and description. 4. Helle receives 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
DeleteAppointment
EditAppointment
CreateAppointment
NavigateCalender
<<include>>
<<include>>
<<include>>
invited
initiate

Use case name:	EditAppointment
Participating actors:	User wants to edit time and date of an appointment.
Flow of events:	1. User opens Calender. 2. Calender shows the calender navigation. 3. User selects the appointment he wants to change. 4. Calender shows the appointment in normal mode that allows changes. 5. User enters submits the wished changes. 6. Calender updates the appointment and notify potential participants about the changes.
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.
Flow of events:	1. User opens the program. 2. Calender shows the calender navigation. 3. User selects edit profile. 4. Calender shows the profile edit. 5. User select delete account. 6. Calender asks about how to handle appointments with other participant (Delete / Leave). 7. User selects how potentiel appointments with other participants should be handled. 7. Calender 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
Note	- Find a smart way to handle events with other participant, perhaps someone owns an event

Use case name:	DeleteAppointment
Participating actors:	User wants to delete an appointment.
Flow of events:	1. User opens Calender. 2. Calender shows the calender navigation. 3. User selects the appointment he wants to delete. 4. Calender shows the appointment in normal mode that allows changes. 5. User selects delete appointment the wished changes. 6. Calender asks how to handle other potentiel participants (Remove / keep appointment). 7. User selects how he wants deletion to be handled. 8. Calender changes/deletes the event accordingly and notify other participants.
Entry condition:	- User is logged in.
Exit conditions:	- Appointment is deleted/user leaves event. - User close the system. - Connection lost.
Quality requirements	- None
Note	- It is possible deletion and leaving an appointment should be seperated. It might be easier for the user to understand.

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:
Std. User:	A person which owns an *account*, and thereby a *calendar*. He/she is able to create/delete/edit *appointments*
Account:	An account of setting and information on a *Std.User*. It acts as a gateway between the *calendar* and the user in the real world
Calendar:	An overview of *appointments* for one *account* after their respective *time*.
Appointment:	A digital representation of an appointment between 1 or more Std.Users. It contains a *time* and a *place* for the appointment, a title and a description for the event.
Time:	A digital representation for a date and time of that day. Relevant for *appointments* and *calendars.*
Place:	A digital representation for a place. Could be bookable or not
Participant:	How an *account* (and thereby *Std.User*) is represented in an *appointment*.