REPORT

TDT4140 Spring 2014

Eivind Christenson

Patrik Fridberg Bakken

Kaj Andreas Palm

Trond Einar Snekvik

Leonid Zaikin

Contents

1 Time Usage **3**

1.1 Overview . . . 3

1.2 Description . . . 3

2 System test **4**

2.1 Test results . . . 4

2.1.1 Test 1 . . . 4

2.1.2 Test 2 . . . 4

2.1.3 Test 3 . . . 4

2.1.4 Test 4 . . . 5

2.1.5 Test 5 . . . 5

2.1.6 Test 6 . . . 5

2.1.7 Test 7 . . . 5

2.1.8 Test 8 . . . 5

2.1.9 Test 9 . . . 5

2.1.10 Test 10 . . . 5

2.1.11 Test 11 . . . 6

2.1.12 Test 12 . . . 6

2.1.13 Test 13 . . . 6

2.1.14 Test 14 . . . 6

2.2 Changes . . . 6

2.2.1 Test 3 . . . 6

2.2.2 Test 14 . . . 6

2.2.3 Other changes to design . . . 7

3 Project experience **8**

3.1 Things to avoid in future projects 8
3.2 Things to bring into future projects 8
3.3 How to prepare better for future projects 9

Chapter 1

Time Usage

1.1 Overview

In the table below we list the planned and actual time usage for each of the project parts. We define these project parts as given in the assignment text.

Project part	**Estimated time usage**	**Actual time usage**
Pt. 1: Project plan	12,5 hours	12 hours
Pt. 2: System test plan	18 hours	18 hours
Pt. 3: General design	80 hours	35 hours
Pt. 4: Implementation and testing	110 hours	74 hours
Pt. 5: Final reporting	10 hours	8 hours

Table 1.1: Time usage

1.2 Description

As we can see from table 1.1 our estimated time usage for part 1 and 2 are quite accurate compared to actual time usage. This is because our group had already completed most of part 1 and 2 when we made the time usage estimations for the entire project. If we take a look at part 3, we can see that we overestimate the time usage by 45 hours. The reason for this is most likely lack of information about the project, and limited experience with time estimation amongst group members. The implementation and testing phase was closer to our target estimation in percentage. However, we notice a significant deviation, possibly due to effective task delegation between group members.

Chapter 2

System test

2.1 Test results

In this section we will give the results from the tests specified in PU2. The test ID from PU2 corresponds to the subsections given here. Since we have tested all our code while implementing (i.e. unit testing), we are happy to see that almost all the test cases passed on the first try when going through the test plan. For details regarding the tests we refer to our documentation given in PU2.

2.1.1 Test 1

Test item: Logging in

Result:
Pass - User is logged in with correct password and can see calendars. Is not logged in with wrong password.

2.1.2 Test 2

Test item: Basic appointment planning

Result:
Pass - New appointment is created and visible in calendar to self and other users

2.1.3 Test 3

Test item: Manage appointment participants

Result:
1st try fail - Participants are not removed
2nd try pass - Participants are added, removed and can accept/decline appointment

2.1.4 Test 4

Test item: Appointment modification

Result:
Pass - All changes are saved and visible in application. Info messages are sent via email.

2.1.5 Test 5

Test item: Delete appointment

Result:
Pass - Appointment is deleted and is never shown in calendar again. Emails are sent to all participants.

2.1.6 Test 6

Test item: Room reservation

Result:
Pass - Room is successfully reserved

2.1.7 Test 7

Test item: Display 1 (appointment status)

Result:
Pass - Change is visible and statuses are updated successfully as well as clearly visible

2.1.8 Test 8

Test item: Status for participation

Result:
Pass - List of participants are visible to everyone invited and their statuses are shown.

2.1.9 Test 9

Test item: Decline meeting invitation

Result:
Pass - All participants are notified via email when a user declines a meeting. A user can also delete himself as a participant.

2.1.10 Test 10

Test item: Room reservation cancelling

Result:
Pass - A list of available rooms are automatically generated and user chooses which one to book. When the meeting is deleted later, the reservation is lifted and another user can book the room at that time.

2.1.11 Test 11

Test item: Display 2 (the calendar)

Result:
Pass - A clean calendar is shown with meetings shown on the appropriate days. Pressing next/previous successfully goes to the next/previous. We have a whole month instead of just a week to make it simpler for the user.

2.1.12 Test 12

Test item: Tracking participants

Result:
Pass - When participants change their statuses to "declined", "attending" or ignores it, these are shown in the list when entering an appointment with the names displayed.

2.1.13 Test 13

Test item: Multi-calendar display

Result:
Pass - All calendars are shown appropriately when the respective check boxes are marked. You will still see that there are appointments on that day in the calendar view, but you will not be able to se them in the appointments table.

2.1.14 Test 14

Test item: Alarm

Result:
1st try fail - We have overlooked this while implementing.
2nd try pass - Users can put in alarms and will get a message via email. Since this application is not running continually on a server, this will only happen when application is running.

2.2 Changes

2.2.1 Test 3

Change after 1st try: Fixed a mixup in the lists that shows the participants and the back bone list that holds the current participants.
Messages are not included here as the specification does not mention it in requirement 3.
A mail is however sent to external participants as the new requirement says it should

2.2.2 Test 14

Change after 1st try: Implemented possibility of putting in alarms

2.2.3 Other changes to design

1. We decided to move the save/read file from the abstract class Manager to the implemented parsing methods in the respective Managers (i.e. UserManager, CalendarManager and RoomManager). This to make the parse coding simpler.

2. When implementing we found need of several getters and setters for information from classes other than the one we were working in. These are not the simple getters and setters, but more complex ones like getting a user from UserManager through the ArrayList of User objects. Other than these we have also implemented methods for equality to make the code cleaner and simpler to implement (i.e. checking equality of two User objects).

Chapter 3

Project experience

We feel that this project has given us a good insight into how a larger project with several units should be executed. Even though we only had TDT4140 and didn't do the entire "fellesprosjekt" we feel that the planning and execution of our part of the project was a good representation of how we imagine a proper development process is executed. Although we feel that overall the execution of the project have been very smooth, there are parts which we realize that we should have done differently. This brings us to the next section.

3.1 Things to avoid in future projects

Overall we feel that the project have been a very smooth process for all of us, and we have all worked well together towards our common goals. Therefore there are not a great deal of things we wish to avoid in future projects, and those that we do wish to avoid are almost exclusively due to misunderstandings. For instance one thing we wish to avoid in future projects is misunderstandings concerning meeting times. A typical example is when we make a verbal arangement to start at 10, and some of us are certain that 10 is 10:15 rather than 10 sharp. This is a quite common misunderstanding at NTNU though, and not something that have lead to any problems.
Another important thing to avoid in future projects is lack of communication. This to ensure that we do not have problems with parts of the implementation not interfacing with each other due to differences in design. Again this did not create a real problem that was beyond a quick fix, but none the less an important part to have in mind for future projects.
Thirdly we could have made a more clear agreement on the distribution of the workload. This to avoid that a single individual is assigned a seemingly simple task that in reality turns out to be quite complex and time demanding.
Lastly we should have planned more ahead in terms of working environment. Since we consistently met at the same times we should have ensured that we had a good place to sit and work, rather than resorting to one of the larger computer labs. A proper room to work in would most likely result in a more focused environment with less noise which in turn would lead to a more efficient session.

So with these realizations it's time to reflect a bit on what we did well and should carry on with us into the next project we participate in.

3.2 Things to bring into future projects

One of the things we are most content with as a group is the use of a common platform for version control. We decided to use git via the github servers rather than svn. This choice enabled us to work independently over several different platforms. This was a necessity for us since all of us use rather

different operating systems and SDKs. It also ensured that changes were easily merged along the way rather than a manual merge in the end of the project, which often results in a lot of work before the project is functional. But version control in itself is not a magical solution, and as previously stated we did have some problems with compatibility even with version control.
Another good practice we used was to divide the project into smaller and more manageable parts. This made it fairly easy to assign manageable tasks to the members of the group. This breakdown of the project worked very well, even though we had a couple of parts that proved to be more extensive than anticipated.
This fragmentation of the project also allowed us to better utilize the different skills within the group. Since we (mostly) had small and manageable parts we could assign these in a more specific manner to the members best equip to handle those particular challenges. This ensured a more efficient way of working and a good utilization of our different skills.
Lastly we have, as mentioned, been working on the project at specific set times every week, which have ensured a steady progression of the project. This way we could handle challenges in the project in a timely manner and ensure that we were meeting our set deadlines.

3.3 How to prepare better for future projects

In this section we will reflect on how we can improve our preparations both in ways which we didn't think of earlier and to fix the problems listed in section 3.1. First of all we can better our communication. One way to do this is to have a meeting at the start of the project to set some ground rules to avoid misinterpretations. During this meeting a common system for communication can also be agreed upon, instead of ending up using different channels to reach different members of the group outside of the meeting times.
Secondly we should have spent a bit more time in the planning phase of the project to create a more detailed analysis of the workload. This is to prevent occasions when a seemingly small task suddenly reveals itself to be more complex and time demanding than previously anticipated. This would also enable us to better decide which member should be responsible for which task before we start with the implementation. This was done during the project, but we realize now that we should have spent more time on these preparations to save us time later in the project.
Thirdly we could also have spent more time on PU3 and made the design a bit more detailed. This would have saved us some time during the end of the implementation stage, where we on occasion realized that something rather should have been implemented in way X instead of Y. A more detailed design plan would probably prevent a few of those issues from arising.
Finally there are the demands to the system. These demands should have been discussed more among the group members. Almost all of the demands have been mentioned at some point to some extension, but there have been cases where we have made assumptions about demands without discussing them formally in the group. The result from this is that we have had occasions where the assumptions differ and two modules that are supposed to interact are designed with different assumptions in mind. This most often result in time being spent to first clarify the demands and then redesign some of the modules. With a more coherent list where the demands are clearly defined as we interpret them this extra work could have been completely avoided.
All in all we can conclude that to prepare better for future projects we need to spend more time in the planning stage of the project, thus saving time during the actual implementation.