Rent It

Second-Year Project,
Bachelor in Software Development,
IT University of Copenhagen

Group 12
Jakob Melnyk, jmel@itu.dk
Frederik Lysgaard, frly@itu.dk
Ulrik Flænø Damm, ulfd@itu.dk
Niklas Hansen, nikl@itu.dk
Jacob Claudius Grooss, jcgr@itu.dk

April 26th, 2012

Contents

Chapter 1

Preface

1.1 How to read this document

Chapter 2

Introduction

2.1 Problem analysis

2.2 Assumptions

Chapter 3

Cooperation

3.1 Cooperation

3.2 ITU group structure

3.2.1 Meetings

3.3 SMU cooperation

3.3.1 Meetings

3.3.2 Conflicts

3.3.3 What we could have done differently

Chapter 4

Design

4.1 Database

4.1.1 Analysis

Decisions

ER-model

SMU involvement

4.1.2 Tables

4.1.3 Entity Framework

4.2 Service

4.2.1 Analysis

4.2.2 Architecture

4.2.3 Interface

SMU involvement

4.3 Client

4.3.1 Analysis

4.3.2 Architecture

4.3.3 Graphical User Interface

Usability

Chapter 5

Implementation

5.1 Service

5.1.1 Architecture implementation

Issues, workarounds and fixes

5.1.2 Error handling

5.2 Client

5.2.1 Architecture implementation

Issues, workarounds and fixes

5.2.2 Error handling

Chapter 6

Manual

6.1 Client

6.1.1 Navigating the client

6.2 Service

Chapter 7

Testing

7.1 Strategy

7.1.1 Test types

Scenario-level tests

Service-level tests

Graphical interface tests

7.1.2 Regression tests

7.2 Test results

7.2.1 Code coverage

7.3 Reflection on test strategy

Chapter 8

Conclusion and reflection

8.1 Issues and potential fixes

8.1.1 Issues we would prioritise

Bibliography

[1] Blog: http://www.artima.com/weblogs/viewpost.jsp?thread=203994 (22nd of April, 2012)

Appendix A

Written Review

Written review

Layout

Even though the first draft is only ten pages, it feels odd not to have a table of contents (ToC). A ToC would have made it easier to navigate the document.

Section/subsection titles were not very distinct from the standard text. Either some form of numbers or letters could be used to show that a new section begins. Italics could also be considered.

The description of the data model and the web-service felt quite clustered and had no real distinction between when one ended and the other began. Again sectioning could alleviate this issue.

Bullet points (or some other "fancy" representation) of the methods in the web-service description would have been good. It was not very clear what was a method and what was not.

The communication section could have been improved by splitting it into subsections. There are three or four subjects discussed in the section and each of them could have had their own subsection.

Content

The ER-model notation is mentioned (and Søren Lauesens book referred to) in the communication section, but not in the data model section. It probably should be, as the data model section lacked an explanation of how to read the ER-models.

The test report section could have been improved by showing (and explaining) a template, then giving an example of an actual report log entry. Another option could be to just explain the report log snippet you have actually included, instead of just leaving it in there with no explanation.

The text has a lot of poorly argued for decisions. One of particular note is "Revised ER_model. If your smart you will notice that there is no distinction between admins and normal users, but just use your imagination and trust us n this one."

Another example of poor argumentation: "The way we have been testing out system is far from the most optimal way, but it is the only way to do it." A tool such as Pex is mentioned, but it would be nice if there were a more detailed explanation as to why your system is designed in such a way that only manual testing is possible. It seems weird because it is possible to make tests that do exactly what you are doing in your system. They can just do it automatically. Spamming could be avoided by just being sensible about how often/when you test.

Grammar and spelling errors: We assume this is because of no proofreading being done, but this should be a priority come the final hand-in. One example is the label for the revised data model as quoted previously. Just that label has two spelling errors.

The use cases could use some elaboration (in terms of describing each use case), but the illustration works quite well, and you could possibly just explain the more complicated ones (what does it mean to update a movie?).

You have a good problem introduction (besides the spelling and grammar problems).

Description is a bit lacking on the data models, but the way you present them is quite good.

Appendix B

SMU meeting logs

Appendix C

System Diagrams

C.1 Class diagrams

C.2 Service API

Appendix D

Test results

D.1 Result table

Appendix E

F# Handins

E.1 F# Handins - Frederik Lysgaard

E.1.1 HandIn 1

E.1.2 HandIn 2

E.1.3 HandIn 3

E.1.4 HandIn 4 & 5

E.2 F# Handins - Jacob Claudius Grooss

E.2.1 HandIn 1

E.2.2 HandIn 2

E.2.3 HandIn 3

E.2.4 HandIn 4 & 5

E.3 F# Handins - Jakob Melnyk

E.3.1 HandIn 1

E.3.2 HandIn 2

E.3.3 HandIn 3

E.3.4 HandIn 4 & 5

E.4 F# Handins - Niklas Hansen

E.4.1 HandIn 1