Project notebook

Customer management application

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project No.** | **Doc. ID** | **Version** | **Date** | **Author** | **Status** |
| 2018-001 | 1000 | 1.0 | 10.05.2018 | JoAn | Valid |

Table of contents

[1 Introduction 2](#_Toc513738596)

[1.1 Structure of document 2](#_Toc513738597)

[1.2 Related documents 2](#_Toc513738598)

[2 Deadlines 2](#_Toc513738599)

[3 Project team 2](#_Toc513738600)

[3.1 Josef Anzengruber (JoAn) 2](#_Toc513738601)

[3.1.1 Tasks in Project 2](#_Toc513738602)

[3.2 Miro Vuorisalo (MiVu) 3](#_Toc513738603)

[3.2.1 Tasks in Project 3](#_Toc513738604)

[3.3 Wilhelm Peter Lehner (WiLe) 3](#_Toc513738605)

[3.3.1 Tasks in Project 3](#_Toc513738606)

[4 Project standards and tools 4](#_Toc513738607)

[4.1 Development environment 4](#_Toc513738608)

[4.1.1 Programming language 4](#_Toc513738609)

[4.1.2 Text editor and debugger 4](#_Toc513738610)

[4.1.3 Version management system 4](#_Toc513738611)

[4.2 Coding standards 4](#_Toc513738612)

[5 Task documentation 5](#_Toc513738613)

[5.1 Task 1: Design 5](#_Toc513738614)

[5.2 Task 2: Testing 6](#_Toc513738615)

[5.3 Task 3: Detailed design and implementation 6](#_Toc513738616)

[5.4 Task 4: Versioning 6](#_Toc513738617)

[5.5 Task 4: Version 2.0: Internationalization 6](#_Toc513738618)

[5.6 Task 5: Final steps 7](#_Toc513738619)

# Introduction

## Structure of document

This document basically includes the deadlines, a short description of the project team and the task of each team member, the project standards and used tools and finally the requirements and tasks of the project.

## Related documents

For the documentation of the project two documents were required: a development handbook and an user documentation. The content of development handbook was split and is now included in the project notebook and in the development handbook.

The ***project notebook*** is a collection of important documents (i.e. deadlines, requirements, tasks, team members, project standards, etc.). The ***development handbook*** is a description of the solution and reasons for decisions (i.e. static structure, dynamic behaviour, component description, test case, etc.). The ***user documentation*** is an information for the user to get started, to reinstall application and to back up the stored data.

# Deadlines

There was only one deadline for all tasks:

***15.05.2018***

This is the date where the final version of both required versions must be handed in.

# Project team

## Josef Anzengruber (JoAn)

Josef studies Automation at the University of Applied Sciences of Upper Austria in Wels. He is in his sixth semester.

### Tasks in Project

* Design:
  + General design of application
  + Detailed design of project library
* Implementation:
  + Project library: Customer and Encrypt classes
* Testing:
  + Unit tests of project library
  + Static tests: Code review of GUI and project library
  + Integration tests
* Documentation:
  + Project notebook
  + Developer handbook: setup, design, implementation and testing
  + User documentation: setup

## Miro Vuorisalo (MiVu)

Miro studies Automation at the Helsinki Metropolia University of Applied Sciences. He is exchange student at the University of Applied Sciences of Upper Austria in Wels. Miro has only little experience in object-oriented programming and no experience in C#, therefore the workload was distributed (in agreement with the lecturer) in a way, that he did not need to implement any code.

### Tasks in Project

* Design:
  + General design of application
* Testing:
  + Development of test cases
  + System tests
  + Acceptance tests
* Documentation:
  + Developer handbook: introduction, design and testing
  + User documentation

## Wilhelm Peter Lehner (WiLe)

Wilhelm studies Automation at the University of Applied Sciences of Upper Austria in Wels. He is in his sixth semester.

### Tasks in Project

* Design:
  + General design of application
  + Detailed design of GUI
* Implementation:
  + GUI
* Testing:
  + Unit tests of GUI
  + Static tests: Code review of GUI and project library
  + Integration tests
* Documentation:
  + Developer handbook: introduction, design and Implementation
  + User documentation

# Project standards and tools

## Development environment

### Programming language

The task was to develop an application which is able to run on Windows 7 and has a graphical user interface (GUI). Therefore, we chose the programming language ***Microsoft Visual C# 2015***, because the Windows Forms class includes and supports the requirements. Furthermore, two project members (Josef and Wilhelm) had already experience with this programming language and with the Windows Forms class.

### Text editor and debugger

For the implementation of the code ***Microsoft Visual Studio Community (Version 14.0.25431.01)*** was used. It supports the programming language C# and has several tools for automated code generation for Windows Forms applications. Furthermore, this editor supports debugging and helps to find errors very fast.

### Version management system

For the versioning of the project the free software for distributed versioning ***git*** was used. The tool ***GitHub*** was used to share the project online. GitHub is a web-based online service, which is used for software development projects.

## Coding standards

The following code standards were defined (in class) during the startup phase of the project:

1. Class names, member variables and methods start with capital letter
2. Coding notation: Camel case is used, e.g. numberOfKrapfen
3. constants in ALL capital letters
4. Description for each class (using ///)
5. Description for each public method and critical private methods (using ///)
6. Meaningful variable names - and in English
7. Curly brackets { } in a new line and required with all loops (no lambda expressions possible)
8. One line ifs possible without { }, but must be written on a single line, e.g. if () ...

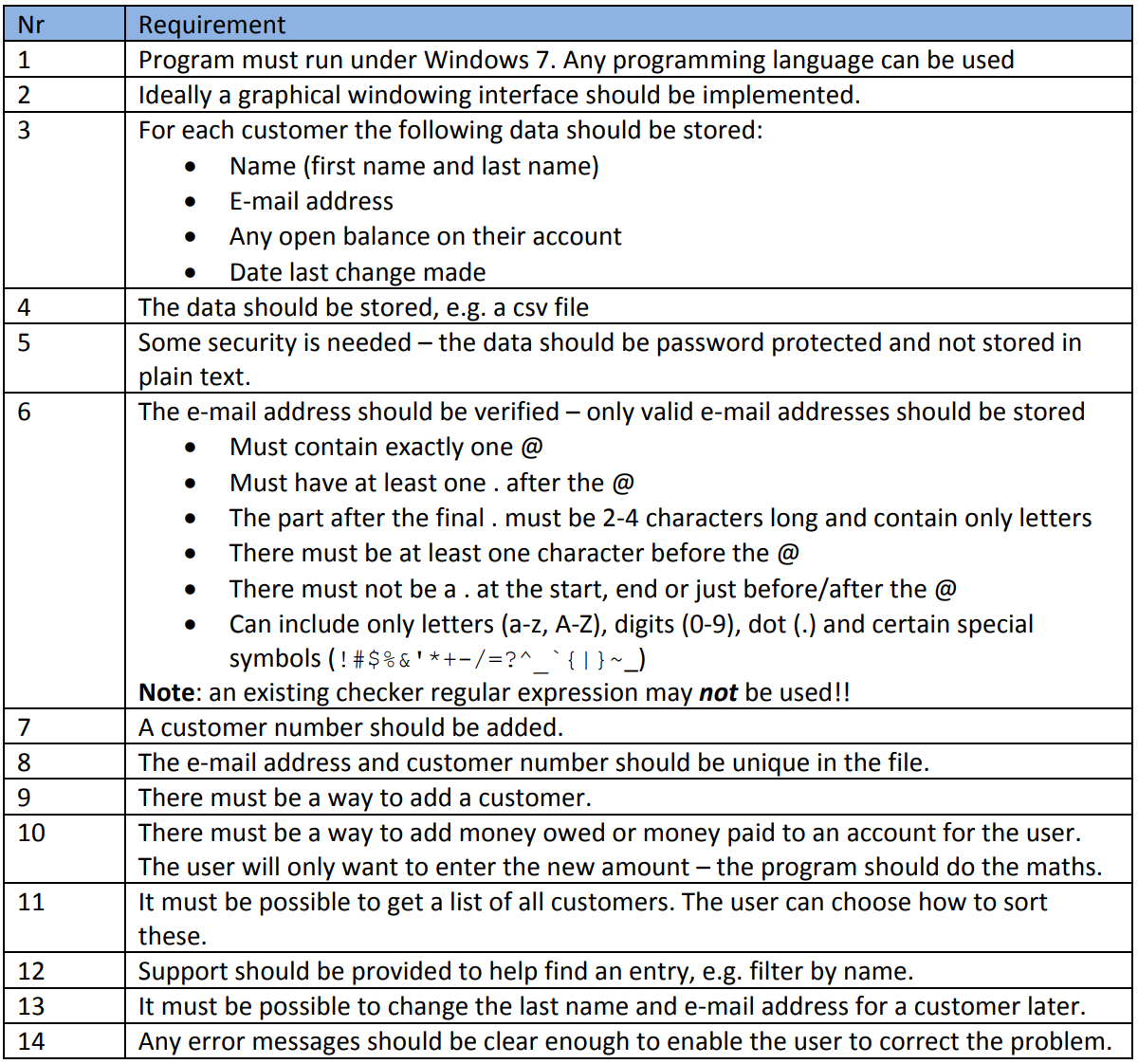
# Task documentation

In this section the tasks are listed.

## Task 1: Design

Think about how you will solve the following problem and how you would split the work relatively fairly ‐ BUT DO NOT IMPLEMENT IT! Then record all details you think you would need to do it assuming a) you won’t do it right away and b) you cannot talk to each other during the development. You do not have to use a tool to do this – paper is fine.

The program is meant to store customer data for a small company. You can assume the users speak English and have experience using Windows.



## Task 2: Testing

For your project/design, write a list of test cases. Since we have no code, these are “black box”. Add a column that shows which method helped you identify the test case.

1. First, write tests based on the requirements
2. Then, think about the equivalence classes and add additional test cases if necessary.
3. Finally, think about the extreme values and add additional test cases if necessary.
4. If you have an activity diagram, etc. you may see even more test cases based on paths.

Now, you have a fairly complete list. Mark those that are most important and need to be checked each time – and those you wouldn’t really do or rarely. Try to group them logically (i.e. if I found an error here, I’d also want to test…).

## Task 3: Detailed design and implementation

Please try to do this without discussing with your teammate(s) – see how well it works. If you have to get additional information, write down what information you needed to share.

Develop a first version of your part of the customer data program. Do unit tests and correct any problems you find.  If necessary, implement stubs so it runs independently. You can write automated tests for your own code if you want – otherwise make a list of tests you did.

Don’t forget to apply our coding standards! And remove all compiler warnings.

## Task 4: Versioning

* **Check in your part of the program:** Create a repository for your team. Each of you should check in the code you did as homework from your own computer.
* **Integrate your program (in class):** Try to integrate your code in your team to get it working together. Before running the tests, download the whole thing from the repository only one computer. When you are done, check‐in the changed files.  Tag this. This is your alpha version (i.e. ready for tests).
* **Spread know‐how in the team:** Do a code review (line‐for‐line inspection), so that another team member understands the algorithm for generating and checking passwords (ie. just this part).
* **Finalize your program (due later):** Finally, think about which tests from your list need to be done. Split these between you and make a list of the test results. If it is necessary, make corrections. Then run ALL of the tests again. When you are done upload the updated version.  Tag this (Version 1.0).

## Task 4: Version 2.0: Internationalization

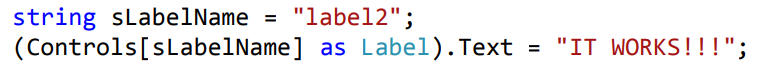
After you have finished the other version of your program and made a release.

Make at least one form in your project international, i.e. allow the user to change the language, and then change all labels/values. Also store what language they chose for next time the program is started.

Some options:

* File for each language (most common?)
* Csv 1 column for each language
* XML structure
* DB (less common)

One possibility for implementing:



Then check this in, re‐run the tests and make another release.

## Task 5: Final steps

* **Document for maintenance:** Write a developer handbook. For this, document information for future developers:
  + Coding standards
  + Versioning (e.g. where)
  + Class diagram (you may want to hide some details or use color to make it easier to get an overview)
  + File structure and naming – plus what you use to encrypt it
  + Important implementation details, e.g. when file is loaded, unencrypted, … (as far as possible, using diagrams)
  + How to test, where test cases are, …
* **User documentation:** Include the essential documentation:
  + “Getting started” guide for first‐time user
  + Operators information: What needs to be backed up, how to use old logfile when reinstall
  + Release notes for second release
* **Prepare a delivery:** Prepare a USB or DVD to use for installation. (Be sure to test it!)