Software Requirements Specification

for

Customer Relationship Management Framework

Version 1.2 approved

Prepared by Ing. Stanislav Jaša

ČVUT v Praze, Fakulta elektrotechnická

2012-10-13

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Stanislav Jaša	2012-10-07	Initial Edit	1.1
Stanislav Jaša	2012-10-13	E-R model	1.2

1. Introduction

1.1 Purpose

This document describes software requirements of the software project called Customer Relationship Management Framework. This is the first, initial version. The next version will also contain E-R database model as well as other remarks on the initial software design that will further develop the idea of the concept introduced in this document.

1.2 Document Conventions

This documents uses standard typographical conventions. Additionally, advanced economic and computer science terms are used. The shortened forms of terms are to be avoided. In case of frequent use, the term definition is described in the brackets after the first occurrence of the term, and in the last part of this document, references. Requirement priorities are stated in words, no typographical convention is used to emphasise one software requirement over the other.

1.3 Intended Audience and Reading Suggestions

The Intended Audience of this document is the evaluating teacher.

1.4 Product Scope

This product aims to cover basic conceptual issues of systems that are providing *application programming interfaces* (APIs), but are, in their current state, not able to cooperate with each other as no common standard exists. The practical application is therefore the interconnection protocole between a users database of unix mail solution, invoicing software, and help desk ticketing system that all need to be presented to the customers as one integral part. The scope of this product is to provide a bridging machanism between these components.

1.5 References

The references cover basic technologies and ideas that will be furthe made use of in this document.

- Enterprise JavaBeans Technology. (2012). Oracle | Hardware and Software, Engineered to Work Together. Retrieved from http://www.oracle.com/technetwork/java/javaee/ejb/index.html
- Java EE Technical Documentation. (2012). Oracle Documentation. Retrieved from http://docs.oracle.com/javaee/
- Online fakturace udělaná jednoduše správně | Fakturoid | Fakturoid. (2012). Retrieved from http://www.fakturoid.cz/
- Oracle, Inc. (2012). JavaServer Faces Technology. Oracle | Hardware and Software, Engineered to Work Together. Retrieved from http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
- OTRS Help Desk - Open Source Service Management | OTRS. (2012). IT Service Management Software - Free Open Source Help Desk - Problem Management System - Customer Interaction Software | OTRS. Retrieved from http://www.otrs.com/en/software/otrs-help-desk/
- Spring Framework | SpringSource.org. (2012). SpringSource.org |. Retrieved from http://www.springsource.org/spring-framework
- Zimbra - Community. (2012). Zimbra offers Open Source email server software and shared calendar for Linux and the Mac. Retrieved from http://www.zimbra.com/community/
- Hibernate - JBoss Community. (2012). Retrieved from http://www.hibernate.org/

2. Overall Description

2.1 Product Perspective

This product is a state-of-the-art solution made solely for the purposes of instruction in the school subject *Webové a podnikové aplikace* (WPA). It is not indended to replace nor complement any other existing solutions, yet its future convenient application may be encountered.

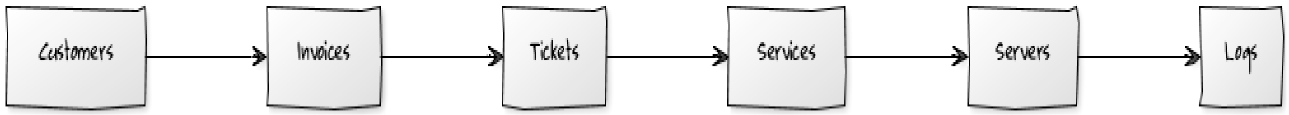

Figure 1: Main framework components workflow

In the figure 1, the basic components of the whole framework are illustrated. The main points are the scalability, and applicability of the system. Customer interaction, accounting interface, ticketing support, the actual software services, hardware layer, and logging are all bridged by one single design.

This product as it is does not aim at providing ultimate solution. It rather pursues a simple, and effective approach to model a conventional business process of an IT company.

2.2 Product Functions

The product will in its essence provide these functions:

- customer registration, login and interaction with the purchased services;
- invoicing API communication;
- help desk support via support tickets;
- service bridging interface – methods how to call API's of 3rd party products, and perform basic customer specific actions;
- hardware platform health realtime monitoring;
- logging of all customer interactions.

2.3 User Classes and Characteristics

This system will have in essence two basic user classes:

1. customer using the *client interface*;
2. provider using the *agent interface.*

Customers will access simplified web interface for downloading invoices, issuing new orders, filing new support tickets, and requesting service interactions like web service restart, new mailbox registration, password reset, etc.

Providers will be the technical personnel overseeing the actions performed by the customers. In the most possible number of cases, the providing side will not directly interfere with the customers so that the system can be merely fully automated.

2.4 Operating Environment

The target hardware platform will be Ubuntu linux operating systems with java platform installed. Java EE 6 will be used as a main standpoint, alongside with a servlet container of the providers' choosing.

2.5 Design and Implementation Constraints

Based on the specification above, the system itself will merely avoid to implement any heavy communication protocols. Also security constraints will be taken into account so any rather potential threats will be elimited by simply not implementing such operations.

2.6 User Documentation

This document will be the core of all the documentation. Further user documentation will be provided “on-the-go” on the customer requested basis.

2.7 Assumptions and Dependencies

The work will be likely limited by a very strict time frame, therefore the whole system is intended to simplify actual implementation steps. The modular structure however does not prevent from any further extentions.

3. External Interface Requirements

3.1 User Interfaces

All the users will use classic computers for accessing the system. The GUI itself will not be important for this project as common design layout will be used, and predefined components deployed. The technologies used for the user interface definition will be HTML5 / CSS / Javascript user interface accessible via any standard modern internet browser; being tested in linux google chrome browser only. Further on, Java Server Faces will be used as a rendering platform with a mixed Java Server Pages, and simple Java Servlet output.

3.2 Hardware Interfaces

Classic linux server installation is required, alongside with Java container, namely Glassfish in this implementation. As for the multiplatfom nature of java, the possiblity of running the application on any other java enabled plaform is not denied. Simple virtual server is certainly a possitive choice. Hardware monitoring data will be collected via simple XML or JSON interace that will not require any additional harware resources but the ones already available.

3.3 Software Interfaces

Recommended operating system, on which the software will be tested, is Ubuntu Server 12.04. Java Application Server will be Glassfish in the lastest version. Database functionality will be facilitated by MySQL Server 5. Additional java technologies employed will be Java Server Pages / Java Server Faces, JBoss Hibernate, Spring Core technologies, Java Persistence API, java.net package, XML and JSON parses provided by the libraries above.

3.4 Communications Interfaces

All the communication with the users will be conducted using the HTTP or HTTPS protocols. Passwords will be encrypted upon submission by the internal MySQL hasing function. The user will be confirmed of all the relevant operations progress via e-mail. The softwars assumes a standard server deployment with a full-duplex 100 Mbps connectivity uplink so a reasonable processing time can be guaranteed.

Moreover, the software will heavily rely on the third party APIs that will deliver the required functionality for invoicing, server monitoring, available services implementation, etc. In essence, this software is a core framework providing stub implementations for basic, and simple services, not trying to replace full functionality of full-featured administration interfaces. Therefore, for instance in case of standard unix mail users administration service is implemented from within the system, only user addition, password change, and user deletion use cases will be within the relevant scope of this project.

4. System Features

4.1 Customer Registration, Ordering, Invoicing, and Login

4.1.1 Description and Priority

Customer is able to register, order a product / service of his choice, and login to the client interface.

4.1.2 Stimulus/Response Sequences

Registration form has to be filled in. Upon the successful submission, the confirmation e-mail is sent. The customer is then offered a list of services for which he/she is able to pay for instantly via a third party payment solution provider. Lastly, the further customer interaction is facilitated by a login / login authentication routines. At this point, the system only facilitates login / logout actions.

4.1.3 Functional Requirements

REQ-1: Create customers database structure.
REQ-2: Generate registration form.
REQ-3: Facilitate customer details validation.
REQ-4: Send a confirmation e-mail.
REQ-5: Generate login form.
REQ-5: Create services database structure.
REQ-6: Create invoicing database structure.
REQ-6: Generate list of services available alongside with their payment options.
REQ-7: Communicate with an external payment gateway.
REQ-8: Process payment success information, ie. send an invoicing request.
REQ-9: Enable service actions access to the customer.
REQ-0: All these actions are logged by the logging facility.

4.2 Additional Service Purchase

4.1.1 Description and Priority

The customer is able to purchase additional services, prolong their service subscription.

4.1.2 Stimulus/Response Sequences

After successful login to the system, the customer is shown a list of services that he/she is able to obtain. After successful payment, the customer is enabled the access to the service.

4.1.3 Functional Requirements

REQ-1: Login / logout repeatedly functional.
REQ-2: Show list of available services minus the services the customer is already subscibed to.
REQ-3: Apply for a new service form generation.
REQ-4: Communicate with an external payment gateway.
REQ-5: Process payment success information, ie. send an invoicing request.
REQ-6: Enable service actions access to the customer.
REQ-0: All these actions are logged by the logging facility.

4.3 Service Cancellation

4.1.1 Description and Priority

The customer is allowed to cancel the enabled service. Based on the prepaid period, the service is not cancelled immediately but after the prepaid period is over. The customer is allowed to download their user data for archiving purposes.

4.1.2 Stimulus/Response Sequences

After successful login, the customer is shown a list of enabled services, any of which they are allowed to cancel.

4.1.3 Functional Requirements

REQ-1: Login / logout repeatedly functional.
REQ-2: Show the list of services the customer is paying for, and are enabled for them.
REQ-3: Select a service to cancel.
REQ-4: Show a cancellation warning confirmation, and inform of a cancellation revoke period.
REQ-5: Submit a cancellation ticket request.
REQ-6: Wait for the provider agent communication.
REQ-7: Wait for the service run-out period.
REQ-8: Cancel the service.
REQ-0: All these actions are logged by the logging facility.

4.4 Ticketing System Operation

4.1.1 Description and Priority

The customer has a right to conctact a provider's agent that will react on the support issues that the customer might encounter during their interaction with the services.

4.1.2 Stimulus/Response Sequences

The customer submits a support request ticket, the agent replies to it. The agent is therefore required to be constantly present within the agent interface. The automatic update facility will be provided, however no other than a standard web browser interface will be created.

4.1.3 Functional Requirements

REQ-1: Login / logout repeatedly functional.
REQ-2: Show the list of tickets previously submitted to the agent.
REQ-3: Generate a 'New Ticket Submission' form.
REQ-4: Create the ticketing database structure.
REQ-5: Save the ticket to the database.
REQ-6: Alert the agent of the new ticket submission.
REQ-7: Generate a ticket response form to the agent.
REQ-8: Show the agent response to the customer.
REQ-0: All these actions are logged by the logging facility.

4.5 Agent Interface

4.1.1 Description and Priority

The agents are able to interact with the customers via the ticketing system. At the same time monitoring interface is provided so that the agents are able to observe the server status.

4.1.2 Stimulus/Response Sequences

The agent interface is available to a restricted range of IP addresses. Each agent is assigned a unique identifier that they use to further communicate with a customer.

4.1.3 Functional Requirements

REQ-1: Check the agent is loggin in from an allowed IP address, and with a right credentials combination.
REQ-2: Show unprocessed ticket requests list to the agent.
REQ-3: Reply to the customer ticket request.
REQ-4: Update server health information in the database.
REQ-0: All these actions are logged by the logging facility.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application will be able to run effectively within any convenient JEE container.

5.2 Safety Requirements

No backup options are within the layer of this application predefined. Low level backup methods seem to be more convenient for this particular purpose.

5.3 Security Requirements

Severe security measures must be taken in account especially concerning the customer data. Today many solutions provide sufficient security, the discussion of which is out of the scope of this document. The application itself will be protected by the predefined combination of username / password, and IP restriction options.

5.4 Software Quality Attributes

The quality of the software will be assured by a JUnit testing facility. For each of the functional requirements, a satisfactory, fully automatic test suite will be designed.

5.5 Business Rules

All the work must be finished within the deadlines provided by the teacher of this univesity subject.

6. Other Requirements

Appendix A: Glossary

API	Application programming interface
TBD	To be determined
Java	Development platform used for this project
JEE	Java Enterprise Edition
JEE container	An environment capable of running the code code of this project
JUnit	Java testing facitlity used for the development of this project
IP address	Internet protocol address
WPA	Webové a podnikové aplikace; scope of this project

Appendix B: Analysis Models

6.1 E-R Model of the above described software

The E-R model is showed in the figure 2. The main components respect the above stated design. Especially, they strictly follow the key ideas introduced in the figure 1 (basic workflow of the software). The architectural design is based on the simple idea of cooperation, and at the same time independence of all the components.

The central part of the diagram showed in the figure 2 is the table customers. Each customer can purchase services that are invoiced to them. The time period from and until which they are made available is kept in the descriptive table ivoices_items. The sole purpose of this table is to gather all the information about the purachased services, and all future automatic renewal based on the customers' wishes.

The next complex structure in the figure 2 is the ticketing system. Each ticket can be related to another ticket, creating thus a conversation. Each ticket can also be assigned an agent responsible for the ticket administration. Agents are responsible for many tickets, yet at one time only one agent can be responsible for a ticket.

Third, and rather independent part of the figure 2 is servers monitoring table, intended to store the actual server health of all the infrastructure. This table is expected to alter depending on the requirements of each of the services provided. The basic monitoring of network, server load and free memory is however predefined.

Every record in all these tables is at the same time represented by a unifying structure in the objects table. The purpose of this table is mainly to allow mutual compatibility, and future extendability of the sytem --- especially for the logging purposes.

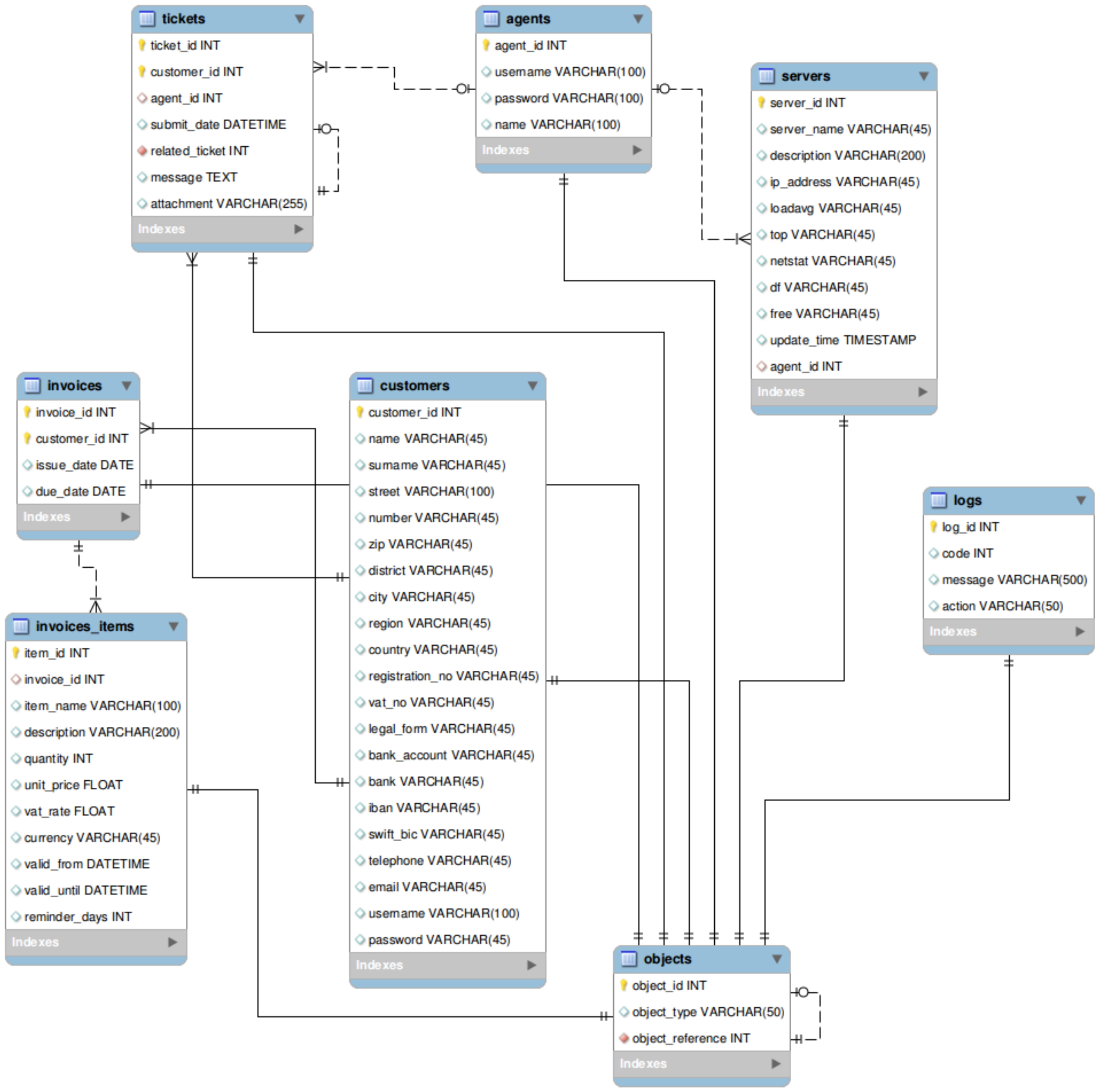

Figure 2: E-R model of the project in Crow's Foot notation generated by MySQL Workbench

Appendix C: To Be Determined List

Empty for now.