Software Requirements Specification

for

EarthTeamProject- Banking Website

Version 0.2 approved

Prepared by:

Slav Radev

Radomir Kolev

Angela Teneva

Iliyan Iliev

Plamen Dimitrov

Vasil Arabadzhiev

Stefan Dimitrov

SoftUni

16 April 2016

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document provides an overview of a Banking Website - revision v1.0 from a functional perspective, highlights some specific considerations relating to the application, and also highlights some technical and organizational implementation issues.

## Document Conventions

1.2.1. Standards followed when writing this SRS:

* 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

1.2.2. Typographical conventions. The format of this SRS is simple. Bold face and indentation is used on general topics and/or specific points of interest. The remainder of the document will be written using the standard font, Arial.

## Intended Audience and Reading Suggestions

This SRS is intended for all related to the "Banking Website" project developers, project managers, QA experts, marketing staff, users, and documentation writers.

The recommended sequence for reading the rest of this document, according to its structure, is: beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

## Product Scope

The scope of this project is the creation of "Banking Website" system. Team "Earth" will develop a banking website to serve the "Royal Bank of SoftUni" - a modern commercial financial institution that accepts deposits and channels the money into lending activities both for business clients and for individuals. It provides an effective approach of managing bank clients communication, working for accounts information, withdrawal and deposit operations. Using this software system you can keep record for daily banking transactions. Integrity and security of data as important mandatory objectives are resolved in the banking system in a reliable way.

The application must maintain data and provide a user friendly interface for retrieving customer related details without time consuming, just in few seconds, with 100% accuracy.

## References

1. "Basics of Banking" - http://www.kesdee.com/pdf/BasicsofBanking.pdf
2. "Product – CRM PROVERS" Software Requirements Specification - SoftUni 2015, Sofia
3. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

# Overall Description

## Product Perspective

The product which is developed is called “Royal Bank of SoftUni”(RBSU) – banking solution. Currently this is the only product from the series of financial software for RBSU. It is a self-contained product which will be extended in the future by additional modules.

## Product Functions

Some of the major functions of the RBSU – banking solution are:

* RBSU - banking solution enables its customers and corporate clients to manage and monitor their banking transactions.
* RBSU – banking solution provides the ability to view history of transactions.
* It has unique credit calculator.

## User Classes and Characteristics

The two major groups that will use this product are:

* Customers – vast majority of the income come from this group. The focus of the product is to keep them satisfied.
* Corporate clients – small but growing percentage of the income of RBSU is generated from this group. One of the focuses of the product is the attraction of more clients of this group.

## Operating Environment

The product will be deployed on the main web servers of RBSU. It will has access to the internal systems and resources of RBSU, alongside that it will be accessible from the outside world.

Every customer and corporate client will be able to connect using just computer and browser.

## Design and Implementation Constraints

The budget for the realization of the product is 100 golemi zeleni ☺

The deadline of the project is the end of the week.

The resulting system must be available 24/7 and should be highly scalable and flexible.

To be able to provide high level of security the system will provide Single - Sign – On; it will protect the data form the data corruption; it will provide user management for action accountability.

## User Documentation

* Installation guide
* Administration guide
* Customer guide
* Corporate client guide

## Assumptions and Dependencies

We assume that our customers and corporate clients should have Internet connectivity. The RBUS – banking solution is expected to work on Mozilla Firefox or Google Chrome. RBUS – banking solution should have easy and secure access to the legacy systems of the Bank.

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>