Software Requirements Specification
For
Banking Application

[9/15/2023]
[1.0]

Prepared by:
Author1, Author2, etc.

Table of Contents

Revision History

Version	Date	Name	Description
1.0	10/02/23	Michael Jochens	The first version of the Application Developers' Requirements Document for Banking Application.

1 Introduction

1.1 Overview

This document will describe the requirements that are needed for the Banking Application. This will help the team to develop the project and will give clear guidelines to reach the crucial goal posts on the timeline that we have set. The clients' finances and safety of their information are the most important part of the development of this project.

This application will be an all-around web-based banking platform. It will be created for anybody who wants to start an account with this particular bank.

Example:
This document defines the requirement for the Innovative Publishing system that is being developed for UMKC Faculty. The purpose of this document is to represent the system requirements in a readable way so that clients and stakeholders can understand them and verify them for correctness but with enough detail that developers can design and implement a software system from them.

This document doesn't address *project* issues such as schedule, cost, development methods, development phases, deliverables and testing procedures. Those are addressed in a separate project document and quality assurance test plan.

The Innovative Publishing system is a web-based tool for publishing content on the Internet. It provides a way for authors to get direct and specific feedback from readers while imposing minimal additional work for authors and readers.

1.2 Goals and Objectives

While the team is working through this project, we are focusing on three main issues:

1. First and foremost, security. When working with financial information it is important to keep all the information that runs through this app, the clients' or the bank's, secure. This is to prevent any loss of data to malicious hackers.
2. Ease of use is a focus of this project as well. The usability of this app will be the main attraction to our clients. We want to make sure the app is efficient and carries all the information that the client is looking for.
3.

Example:
The three main goals of the innovative publishing product are:

1. Provide a simple mechanism for readers to offer feedback to authors on content. The rational is that more readers will offer feedback if the process is easy.
2. Use feedback from readers to add value to the content and improve the experience of other readers without intervention from the author. For example, book reviews on Amazon are instant free content that increases the value of the site.
3. The system shouldn't require any extra effort on the part of the author or other publishing agent.

1.3 Scope

The general banking application will be able to show the user's financial information in a clear and concise manner via the dashboard. There will be several actionable items on the dashboard that will give more information about the information on the dashboard. These items will include Accounts, Activity, Profile, Installments (debts). The user will not be able to open a new account online or open a new form of credit online.

1.4 Definitions

This section defines potentially unfamiliar or ambiguous words, acronyms and abbreviations. If terms such as "shall", "should" and "may" are used to indicate importance the meaning of these terms should be defined here.

Example:

Use case – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

Scenario – one path through a use case

Actor – user or other software system that receives value from a use case.

Role – category of users that share similar characteristics.

Product – what is being described here; the software system specified in this document.

Project – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

Shall – adverb used to indicate importance; indicates the requirement is mandatory. "Must" and "will" are synonyms for "shall".

Should – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

May – adverb used to indicate an option. For example, "The system may be taken offline for up to one hour every evening for maintenance." Not used to express a requirement, but rather to specifically allow an option.

Controls – the individual elements of a user interface such as buttons and check boxes.

1.5 Document Conventions

This section describes presentation conventions use in the document.

Example:

Portions of this document that are incomplete will be marked with TBD. Each TBD item will have an owner and estimated date for resolving the issue.

1.6 Assumptions

In this section list any assumptions on which the requirements, as they are described here, depend. Assumptions are conditions, usually outside the control of the performing organization, that are taken for granted.

Be careful to only document assumptions that are outside the control of the performing organization. For example, the following is not a valid assumption for the requirements document: “We assume that all requirements will be documented.” You might be *assuming* this but there is no point in documenting it as an assumption here because it is something that is primary within the scope and control of the performing organization. The purpose of this section is to document assumptions that are outside the control of the performing organization—conditions that should be noted because someone will be taking responsibility for ensuring they hold.

A distinction can also be made between assumptions that pertain to the requirements and those that pertain to the project as a whole. The software project management plan is the most logical place to document assumptions that pertain to the project as a whole.

Example:
It is assumed that the client has an ODBC compliant database installed and this database will be accessible from the machine where the system will run.

It is assumed that the web hosting ISP will allow server-side scripts to access the file system.

2 General Design Constraints

2.1 Product Environment

The General Banking Application will be an independent web application, constructed with Javascript and HTML. The database will be accessed through mySQL.
Example:
The innovative publishing system will be a component of the existing online course management system used in the computer science department at UMKC. The online course management system has built-in support for authentication. The innovative publishing system will use the existing course management system to identify and authenticate readers. The online course management system also includes a relational database which will be used through the JDBC/ODBC interface.

2.2 User Characteristics

It’s important for developers to have a complete and accurate image of the end users. Even when the requirements of the user interface are described in detail the developer will still make many tiny design decisions during design and implementation. Knowing the general characteristics of the end users will help the developer make better decisions.

If the specific users of the system are know list them here. More likely there will be user roles or categories of uses. For each group of users list their responsibilities, characterize their knowledge of the domain and describe their characteristics including technical sophistication, background and education.

Prioritize users if necessary. This is especially important when user characteristics conflict. For example, if the system must accommodate experience users as well as

novices it is important to know which should be given priority in case it's not possible to accommodate both groups.

2.3 Mandated Constraints

Ideally requirements will be specified in terms of functionality needed and developers will have free rein to design and implement a solution. In practice there are constraints on the eventual design and implementation.

Constraints may be mandated technologies. For example, the client may specify that a specific database management system, programming language, and/or operating system be used.

Constraints limit design and implementation options.

Constraints might be absolute, desirable or optional. If constraints aren't absolute the motivation for the constraint should also be given.

2.4 Potential System Evolution

The resulting software system should be maintainable and extensible. Knowing the types of anticipated changes aids significantly in establishing an architecture that will accommodate the types of expected changes. This section suggests ways the system is likely to be extended or modified in the future.

3 Nonfunctional Requirements

Nonfunctional requirements are properties the system must have. Nonfunctional requirements tend to be orthogonal to functional requirements. For example a system may have the nonfunctional requirement that it be offline no more than 15 minutes at a time and not more than ½ hour each week. The realization of this requirements isn't limited to one spot in the code. This nonfunctional requirement crosscuts some or all functional requirements.

3.1 Usability Requirements

It's hard to image a software system that doesn't have usability as one of its highest nonfunctional quality requirements. It's not enough to just say that the system should be usable though. Usability requirements must be stated in a quantifiable and testable way.

One method of specifying usability requirements is to specify efficiency, effectiveness and satisfaction goals for specific scenarios of use (section 4) carried out by representative users (section 2.2). A simpler alternative is to design a survey to measure user satisfaction and get consensus on who will take the survey and what will be considered an acceptable aggregate score.

3.2 Operational Requirements

This section describes the general characteristics of the physical environment for the product.

Example:
The users' environment is noisy so the system shouldn't depend on the user hearing audible output.

3.3 Performance Requirements

The main performance characteristics are speed and capacity (memory). Performance requirements are usually stated as a function of the number of concurrent users. Use this section to state the performance requirements of the system as a whole. If specific transactions have their own performance requirements state these requirements below along with the description of the feature.

Example:
System startup time should be less than 3 seconds. With 30 concurrent users no operation should take more than 5 seconds and 95% of the operations should take less than 2 seconds.

3.4 Security Requirements

Access to data and features may be limited to specific users. There may also be a requirement to keep an audit trail of system use. This section describes the security requirements including the levels and what needs to be protected.

3.5 Safety Requirements

The system may affect the safety of the larger environment. For example, there are limits on the intensity of stray electromagnetic radiation from electronic devices used in hospitals. Potential safety concerns should be investigated and documented in this section.

3.6 Legal Requirements

Users' information will not be viewable publicly and their information will not be
Example:
Student social security numbers will not be visible to other students.

3.7 Other Quality Attributes

There are specific sections above for non-functional quality attributes such as security, performance, etc. In this section describe any other non-functional quality attributes such as portability, availability, etc.

3.8 Documentation and Training

An important part of the total system is the documentation and training that is provided with the system. This section should describe the types and quantity of documentation and training that will be provided with the product.

3.9 External Interface

GUI using HTML

3.9.1 User Interface

The requirements document shouldn't contain design or implementation details. The *logical* user interface should be described here. The description here shouldn't unnecessarily constrain design and implementation options.

The general personality of the interface should be described here. For example, should the interface convey a conservative, professional, authoritative or fun attitude? What is the look and feel? Style? User characteristics were given above in section 2.2 but it may be helpful to characterize the average user here as adult, teenager or child.

User interfaces may contain a mixture of media types. It may be helpful to describe the desired/permissible user interface in terms of media elements.

Should the interface be intuitive or will training be provided. If training is required what types of training will be provided (online help, separate user manual, formal classroom training).

Ease-of-use requirements should be stated in a way that can be verified. For example, "the product should be easy to use" isn't verifiable because it's impossible to define "easy" in a measurable way. Must better is "75% of users will be able to use 80% of the features within 20 seconds without prior training".

3.9.2 Software Interface

The back end of the main application will be written in Java, while the database will be written in mySQL. The front end of the application will be written in HTML.

4 System Features

The core requirements of the system are listed in this section. This template recommends organizing requirements by features rather than use cases. Features are system behaviors from the user's point-of-view. The requirements of a feature are described by one or more use cases plus any additional narration that is necessary

Features should be ranked and listed in priority order. Priority is determined by cost, risk and value. To prevent arguments over the exact values of these measures this template recommends using the values: high, medium and low. There should be a written understanding how the priorities listed here are used to determine what order features are delivered and what determines essential features, desirable features and optional features.

If you are following a time-boxed or incremental development process, any formula used to consider what order to implement features should consider not only the priority defined by cost, risk and value but the features impact on the design and architectures of the system. It may be beneficial to implement a low priority feature before a higher priority feature if it is an important architecture component.

4.1 Feature: Login

4.1.1 Description and Priority

The next two sections will describe efforts to keep our project low cost, as we have no funds; the risks associated will be measured, with security being a focus of the team, as well as reliability from the moment someone begins to enter their username; the value of this project will surpass the projected efforts of the team.

Cost: Very Low
Risk: Low
Value: High

4.1.2 Use Case: User enters correct username and password.

If the user enters a username that exists and a password that matches the username, then the user will be able to access their account dashboard and the functions associated with it.

4.1.3 Additional Requirements

Functional requirements:

- Two text boxes, one for password, one for username.
- Login button.
- If a username or password is entered wrong too many times the account will lock for 5 minutes.
- When a username is entered that is incorrect, a catch will happen that will present a message warning that it is incorrect.

Non-functional requirements:

- General UI cleanliness.

4.2 Feature: Dashboard

4.2.1 Description and Priority

The dashboard is where most of the information on the web application will be found.
Cost: Very Low
Risk: Low
Value: High

4.2.2 Use Case: The user wants to check their account activity.

If the user interacts with the button on the dashboard labeled “Activity”, then the dashboard will redirect the user to the page where the information for the activity of their account is held.

4.2.3 Use Case: The user wants to move money from one account to another.

If the user interacts with a button called “Transfer” on the dashboard, it will take them to a page where their accounts are shown, and they are able to make a request to transfer money from their savings to their checking or vice versa.

4.2.4 Additional Requirements

Functional requirements:

- Button to redirect the user to the transfer page.

Non-functional requirements:

- Data must be shown in a clear and concise manner

4.3 Feature: Account Activity

4.3.1 Description and Priority

Account activity is a page where all previous withdrawals, deposits, and transfers will be recorded.

Cost: Low
Risk: Low
Value: High

4.3.2 Use Case:

4.3.3 Additional Requirements

Functional requirements:

Non-functional requirements: