Software Requirements Specification

For

Commerce Bank Web (CBW)

September 18, 2019

Version 1.1

Prepared by:

Scott Peery

**Table of Contents**

[Revision History 4](#_Toc20147747)

[1 Introduction 5](#_Toc20147748)

[1.1 Overview 5](#_Toc20147749)

[1.2 Goals and Objectives 5](#_Toc20147750)

[1.3 Scope 5](#_Toc20147751)

[1.4 Definitions 5](#_Toc20147752)

[1.5 Document Conventions 6](#_Toc20147753)

[1.6 Assumptions 6](#_Toc20147754)

[2 General Design Constraints 6](#_Toc20147755)

[2.1 Product Environment 6](#_Toc20147756)

[2.2 User Characteristics 6](#_Toc20147757)

[2.3 Mandated Constraints 7](#_Toc20147758)

[2.4 Potential System Evolution 7](#_Toc20147759)

[3 Nonfunctional Requirements 7](#_Toc20147760)

[3.1 Usability Requirements 7](#_Toc20147761)

[3.2 Operational Requirements 8](#_Toc20147762)

[3.3 Performance Requirements 8](#_Toc20147763)

[3.4 Security Requirements 8](#_Toc20147764)

[3.5 Safety Requirements 8](#_Toc20147765)

[3.6 Legal Requirements 8](#_Toc20147766)

[3.7 Other Quality Attributes 8](#_Toc20147767)

[3.8 Documentation and Training 9](#_Toc20147768)

[3.9 External Interface 9](#_Toc20147769)

[3.9.1 User Interface 9](#_Toc20147770)

[3.9.2 Software Interface 9](#_Toc20147771)

[4 System Features 9](#_Toc20147772)

[4.1 Feature: Database 9](#_Toc20147773)

[4.1.1 Description and Priority 9](#_Toc20147774)

[4.1.2 Use Case: Store Transaction 10](#_Toc20147775)

[4.1.3 Additional Requirements 10](#_Toc20147776)

[4.2 Feature: User Interface 10](#_Toc20147777)

[4.2.1 Description and Priority 10](#_Toc20147778)

[4.2.2 Use Case: Monitor Transactions 10](#_Toc20147779)

[4.2.3 Additional Requirements 11](#_Toc20147780)

[4.3 Feature: Notifications 11](#_Toc20147781)

[4.3.1 Description and Priority 11](#_Toc20147782)

[4.3.2 Use Case: Finding Pertinent Transactions 11](#_Toc20147783)

[4.3.3 Additional Requirements 11](#_Toc20147784)

[4.4 Feature: Administration 11](#_Toc20147785)

[4.4.1 Description and Priority 11](#_Toc20147786)

[4.4.2 Use Case: Editing Transactions 12](#_Toc20147787)

[4.4.3 Additional Requirements 12](#_Toc20147788)

[4.5 Feature: Auxiliary Functionality 12](#_Toc20147789)

[4.5.1 Description and Priority 12](#_Toc20147790)

[4.5.2 Use Case: Bank Statement 12](#_Toc20147791)

[4.5.3 Additional Requirements 12](#_Toc20147792)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.0 | 9-18-19 | Scott Peery | Initial Document |
| 1.1 | 9-22-19 | Nick Wiegand | Section 4, 3.2, 3.2.9 |
| 1.2 | 9-23-19 | Kenneth Harris | 3.3 and final proofreading/editing before submitting baseline requirements |

# Introduction

## Overview

The Commerce Bank Web application (CBW), will be a web app available to internet user bank members available on all browsers. The application will provide access to bank member accounts, which will display transactions, and allow the user to configure business rules for alerts and notifications. The app will allow users to view and manage their account.

This document provides information on the requirements for the Commerce Bank Web software application. It provides the project goals, scope and definitions in the introduction of the document. Design constraints and application environment are described in section 2 of this document. Non-functional requirements are outlined for verification. Functional requirements are given to show the system features and expected user interaction.

Project Constraints will be included in a separate document. The Software Project Management Plan will give specifics on project budget and schedule. A separate Test Plan document will be address test specifications and procedures.

## Goals and Objectives

The main objective of this project is to allow bank members a way to access their bank accounts from the internet. The Commerce Bank Web application is expected to:

1. Provide an internet interface with Commerce Bank to access account information
2. Function in a simple and intuitive manner
3. Provide member with transaction information
4. Provide ability to export or print transactions
5. Member has ability to Add/Edit/Delete business rules without assistance
6. Member has ability to receive notifications or alerts based of the business rules they create
7. Member has ability to review history of alerts and notifications, and hide any rules that have not had alerts or notifications
8. Member has ability to print/export their alert/notification history
9. Provide member with security assurance for their information

## Scope

The CBW application will allow users to access their account from the internet. It will allow the user to print or export transaction, alert, and or notification history. It will allow member to create business rules for alerts and notifications.

## Definitions

**Commerce Bank Web Application (CBW) -** the product that is being described here; the software system specified in this document.

**Project -** activities that will lead to the development of the CBW.

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

**Client –** Commerce Bank; the organization for whom the application is being built.

**User –** the person or persons who will actually interact with the application.

**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.

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

**Scenario -** one path through the use case.

**Role –** category of users that share similar characteristics.

**Controls –** the individual elements of a user interface such as buttons and drop-down boxes.

**Developer -** the person, persons, or organization developing the system, also sometimes referred to as the supplier.

**Stakeholder –** anyone with any interest in the project and its outcomes. This includes clients, customers, users, developers, testers, managers, and executives.

## Document Conventions

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. When these issues are resolved, this document will be updated and version number increased to reflect that changes have been made.

## Assumptions

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.

# General Design Constraints

## Product Environment

The Commerce Bank Web application will be an extension of commercebank.com site. It will use the existing web hosting ISP. The application will interface with a relational database management system that will store the users account information.

## User Characteristics

**Commerce Bank Web application users**: bank members are likely to be familiar with using the internet and navigating to a web site.

## Mandated Constraints

The application will be a web application. It will use a database platform of Microsoft Server 2012 or later. It will use .Net and be written in C#.

## 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.

The application is expected to potentially support the following at a later date:

1. Messaging center in the app
2. Notifications via email or text
3. Web API for backend interactions
4. Web API for bank associates and managers to review and edit member account information when helping the customer

# 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.

## Usability Requirements

99.5% of users will be able to set business rules with the application the first time without assistance.

The status of alerts and notifications should be able to be seen immediately upon opening the application.

The application will not use cryptic icons that are too abstract or meaningless to a user of any skill level.

The application will be intuitive in its layout, so the user does not need to guess or hunt for information.

The application will be as clutter free as possible for ease of readability. IE, no busy background or splash images that can interfere with font readability.

Spacing between columns of information will be such that it is easy to determine where one section ends and the other begins. (More than just outlining the cell, actual visible spacing separation shall be used).

The user will be able to undo certain actions easily and immediately upon realizing a mistake. (IE, business rule should have the edit/delete option right next to it).

Compatibility with printer and storage of reports should work effortlessly with out the need to install an app or extensive setup.

## Operational Requirements

The goal is to cut down all the frills usually found in banking apps to provide an experience that doesn’t lead the user down an endless trail of links to find what they need. But in doing this we want to make sure we are not depriving our users of any necessary functionality, so there is an important balance to be maintained.

This means that, operationally, all pertinent tasks need to be simplified down to the core. There should be little to no redirecting, and we want to reduce the amount of documentation the user would need to sift through. Any action the user is interested in doing should be but a one or two clicks and pop-up menus away.

## Performance Requirements

The system shall take no longer than 3 seconds to start up. The system shall handle concurrent queries by different customers, employees, or other users, with no one operation taking longer than 5 seconds to complete for up to 50 users. 95% of operations, but particularly page navigation, account data retrieval, and alerts/notifications should take less than 2 seconds to complete.

## 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.

Bank member account information security is provided by secure login to Commerce Bank Web application.

STRETCH- database will keep timestamp log of account access by user id (IE if member, associate or administrator)

STRETCH – associate can view records, but is not authorized to edit or delete information

STRETCH – administrator may edit information, but is not authorized to delete information

STRECTH – timestamp log of account changes by user id (IE member changes business rules, administrator edits record)

## Safety Requirements

NONE

## Legal Requirements

Members account information and personal information will not be visible by other members or non-authorized Commerce Bank personnel.

## Other Quality Attributes

NONE

## Documentation and Training

System documentation and user guide will be provide to the project stakeholders. The user will not receive training when product is released, but will have access to “ how to” documentation available via website.

## External Interface

### User Interface

The average user will be an adult. The interface will be intuitive. It will be streamlined and simple to use such that 99.5% of users will be able to use with no training or assistance. It will be visually appealing with a straightforward look and feel.

### Software Interface

Whilst there is no interfacing immediately planned, as previously mentioned, in product evolution a web API would be a useful addition to our product. However, until all the other requirements have been met, we will hold off on discussing exactly what interfacing would be useful with our product, and how we would go about implementing it.

# System Features

1. **Database for Transactions**
2. **User Interface to Review Transactions**
3. **Notifications Management**
4. **Administrator Oversight**
5. **Auxiliary Functionality**

These features have been ordered from greatest priority to lowest. That is not to say they are not essential, but without the former features, a majority of the latter would be of little use.

## Feature: Database

### Description and Priority

The database is essential to the entirety of this project’s functions. Without the database there would be no way for the user to keep track of their data. Nor would there be a way for them to interact with it. The database will store all of the pertinent transactional information that the user will need. The database will be the point of query for when business rules are set so that notifications for the user will be generated. It will also be the point of interaction for the administrator who will be overseeing said data.

**Cost**: medium

**Risk**: low, our team has sufficient skill to create such a database

**Value**: high, without the database there is no use for the application

### Use Case: Store Transaction

The user commits a transaction. The transaction is then sent to the database for storage along with all pertinent information. This transaction will then be reviewed to see if the business rules the user has set in place will trigger a notification. An administrator may later query said database for any transaction to review or correct any information found within. The user should be notified of this. The administrator should never be able to fully remove a transaction, we want a comprehensive transaction history.

### Additional Requirements

There are many additional requirements that may or may not be added further down the line as we develop a more comprehensive establishment of what features will be implemented. Some examples would be heightening database security, eliminating redundancy.

## Feature: User Interface

### Description and Priority

This will be one of the features that should entice users to use our application. The interface needs to be a simple, no frills layout that will allow the user to see the transaction history for their accounts as well as find any other info they were looking for intuitively. As the main target of this application is to create something simple and efficient for user interaction with their transaction history, it is second in priority only to storing that history properly.

**Cost**: low

**Risk**: medium, our team is experienced in UI but will be using a framework they are not previously familiar with

**Value**: This is the one of the main features that should attract users to our product

### Use Case: Monitor Transactions

Should the user want to find a transaction, they should be only a few clicks away from seeing what they need. The idea is to cut down on linking between pages or hiding information behind non-descript icons and sub-menus. We want our users to be happy with how quickly they can access account information even if they are manually searching their transaction history.

### Additional Requirements

All info listed up front needs to be pulled from database. Should have comprehensive but not overwhelming info right up front.

Collapsible menus so as to reduce clutter.

Clear design to minimize confusion and prevent excessive hidden functionality.

## Feature: Notifications

### Description and Priority

This feature is designed to help our users sort through their transaction data. With our notification system, the user will be able to define their own business rules. This way they can personally set when they want to see specific notifications. This will allow them to tailor our system to their specific needs.

**Cost**: medium

**Risk**: low, would increase as flexibility of business rules does, would require more thorough testing as rule constraint changes

**Value**: provide a feature which can be useful to a wide variety of users, providing them with a convenient, efficient transaction application

### Use Case: Finding Pertinent Transactions

With this feature our users can easily find transactions they concern themselves over. Even with a wide variety of transactions happening in their account, our user should be able to follow their notifications to find the specific transactions they have set rules to be notified about. This should prevent the user from having to sift through every transaction made in order to find the one they are looking for.

### Additional Requirements

Simplistic module to be used for creating business rules, and pushing them to the user account.

A section to keep track of active business rules, and allow removal/deactivation.

A notification history module.

## Feature: Administration

### Description and Priority

The administrator is brought in for oversight on transactions. They should be able to gain access to account history in order to properly oversee what is taking place when granted access. They should be limited in their power, but not hindered by said limitations. It is important to find balance in their role on our application.

**Cost**: low

**Risk**: high, a proper balance must be maintained, with rogue administration the whole application would go belly up and users would not trust using it

**Value**: high, customer satisfaction is of high import, we want to be able to swiftly and easily rectify faulty records

### Use Case: Editing Transactions

If the user has notified the administrators of a faulty record, or false transaction, the administrator should be able to come in and rectify the transaction. We don’t want transactions removed, even if they are fraudulent, so that the user has a full comprehensive history of their account. Fraudulent transactions should be nullified but remain in the account history with the proper documentation to show that they were taken care of.

### Additional Requirements

Integrated administration tools, so that not a lot of extra clutter or pop-up modules are needed for our administrators to moderate transactions.

Tracking of all changes that have been made by an administrator to prevent abuse of power.

## Feature: Auxiliary Functionality

### Description and Priority

Auxiliary functionality will be implemented further on in the project. Some of these features could include info sections to guide our users how to properly use features, as well as other info fetching services we have deemed of lesser concern than the core features of our product.

**COST**: low

**RISK**: low, the auxiliary functionality will only provide additional information to the user

**VALUE**: high, although non-essential, will provide greater customer satisfaction

### Use Case: Bank Statement

One example of this functionality would be an easy to export document when they request a bank statement.

### Additional Requirements

Information module with simplified documentation and instruction. Ideally not necessary, but useful for those who still need help with the interface.

Ability to search for specific transactions, whether by date, id, location, etc.

Ability to reorder transactions by chosen field.