Software Requirements Specification

For

[Commerce Bank Online Application]

[1.0.0]

Prepared by:

Ahmed Boukhousse, Jose Fonseca, Lyric Randle, Marcus Wong, Richard Schneider, Paula Salazar

**Table of Contents**

[**Introduction**](#_heading=h.ttjgobt3cdvd) **3**

[Overview](#_heading=h.3znysh7) 4

[Goals and Objectives](#_heading=h.hu4zjjqzus1j) 4

[Scope](#_heading=h.3qvewhg6z9s9) 4

[Definitions](#_heading=h.pixm7kp3huoj) 4

[Assumptions](#_heading=h.fbenetvqy95n) 5

[**General Design Constraints**](#_heading=h.mae6o2ac5nmz) **5**

[Product Environment](#_heading=h.17dp8vu) 5

[User Characteristics](#_heading=h.d2n2zgkvot47) 5

[Mandated Constraints](#_heading=h.ewb93ff76qnq) 5

[Potential System Evolution](#_heading=h.3ampa1yxnw4n) 6

[**Nonfunctional Requirements**](#_heading=h.o2stnt4uzlc) **6**

[Usability Requirements](#_heading=h.pstda4ljemdg) 6

[Operational Requirements](#_heading=h.kkwvryvsx2j1) 6

[Performance Requirements](#_heading=h.oi2d2h1ngfcp) 6

[Security Requirements](#_heading=h.xpuvc0sd0zbn) 6

[Safety Requirements](#_heading=h.5gulbkppnjqy) 6

[Legal Requirements](#_heading=h.hyodbraeub2z) 6

[Other Quality Attributes](#_heading=h.j4jk8b8kh9uf) 6

[Documentation and Training](#_heading=h.qr80f05dpo37) 7

[External Interface](#_heading=h.1xy9vknysact) 7

[User Interface](#_heading=h.8d8ely2ubhpp) 7

[Software Interface](#_heading=h.x87auni4x8qw) 7

[**System Features**](#_heading=h.ntvadn2k7lxt) **7**

[Feature: Display of Banking Information](#_heading=h.v04oi6pnmly7) 7

[Description and Priority](#_heading=h.i5esnhagrh5n) 7

[Use Case: See Account Transactions](#_heading=h.vax0zkjyulzi) 8

[Use Case: See and Receive Notifications](#_heading=h.3pu8h3dps90i) 8

[Use Case: Update Notification Types](#_heading=h.l7s7oynv4e7g) 8

[Additional Requirements](#_heading=h.b9dxzr5b2rdf) 8

[Feature: Backend Access of Data](#_heading=h.c3pf0oq8cd2f) 8

[Description and Priority](#_heading=h.1llln0gfeacf) 8

[Use Case: User Authentication](#_heading=h.3o7alnk) 8

[Use Case: API Layer](#_heading=h.dngg1oxibb6) 8

[Use Case: Export User Transaction History into Spreadsheet](#_heading=h.6dbme8g3uuaz) 9

[Use Case: Database](#_heading=h.imrq0cpez8jc) 9

Revision History

| **Version** | **Date** | **Name** | **Description** |
| --- | --- | --- | --- |
| 1.0.0 | 09/29/21 | Initial Plan | Wrote project requirements and outlined future features of the program. |
|  |  |  |  |

# **Introduction**

## ***Overview***

This document defines the requirement for the Online Banking system that is being developed for Commerce Bank by UMKC students. 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 (project charter), and quality assurance test plan.

The online banking system is a web application that pulls in transaction details ( preprocessed data was provided) and allows the user to set triggers for notification rules and receive notifications around them. The system should also save data to a database so recurring reports can be created.

## ***Goals and Objectives***

The main goals of this online banking system are :

1. Making online banking simple. Customers can login to an online banking system and see their transactions’ history.
2. Focus on human centered design to make the web application look user friendly and easy to use and navigate.
3. The system should be very simple and can be used by anyone regardless of their technical level

## ***Scope***

This online banking system will ask users to login first and then it will display their transaction details along with the number of times each notification rule has been triggered over the period of a month. The application will not allow users to sign up but it will allow for password reset. Users will have the ability to authenticate using a username and a password combination; they also will be able to log out!

## ***Definitions***

This section defines potentially unfamiliar or ambiguous words, acronyms and abbreviations.

**Notifications:** Messages that inform users of any changes made to their accounts.

**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.

**Controls** – the individual elements of a user interface such as buttons and checkboxes.

## ***Assumptions***

In this section we 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.

Users have internet access.

Users have a modern, internet browser.

Users have an account with the bank.

Users have an email address.

Users have javascript enabled on their browser.

Users will accept cookies.

# **General Design Constraints**

## ***Product Environment***

The online banking system is a client-server application built with MySQL database, .NET Core 3, and C#. It can be hosted on any major operating system using IIS or Nginx. The user will access the system using a modern internet browser and an internet connection.

## ***User Characteristics***

The system will be built for customers of the bank. These people will care about their finances, budget, and security. They are expected to have technical familiarity with opening a website and browsing information there.

## ***Mandated Constraints***

The only absolute constraints is to use one of the CSS and Javascript Frameworks and to build everything from scratch without the use of templates or content management tools such as Wordpress.

Optional constraints include the use of C#, React.JS, and Bootstrap; these are optional because they are the frameworks most familiar to technical help.

Desirable constraints are to remember user sessions, use an API layer, static code testing, and deploy the website.

## ***Potential System Evolution***

The system can be extended to allow other API requests, such as creation of users, extended to include Multi-Factor Authentication, resetting of user passwords, or other pages and displays for the user.

# **Nonfunctional Requirements**

Nonfunctional requirements are properties the system must have. Nonfunctional requirements tend to be orthogonal to functional requirements.

## ***Usability Requirements***

The user can understand the forms and information provided. The text is readable in all views and words are common language. This will be verified through a random survey of customers.

All date and time formats should follow U.S. standards.

## ***Operational Requirements***

The system should only depend on text-based interactions with the user.

## ***Performance Requirements***

Login time should be less than 3 seconds. Session tokens should expire after 15 minutes.

## ***Security Requirements***

Access to data and features may be limited to specific users. Account Numbers, Social Security Numbers, and Emails will be masked unless the user requests them.

There may also be a requirement to keep an audit trail of system use. Users should only use HTTPS. Passwords will be salted and hashed. Database will use parameterized queries to prevent SQL Injection.

## ***Safety Requirements***

None.

## ***Legal Requirements***

Disclosure of customer information should only occur to authorized people. Customers should be promptly informed of any data breaches.

## ***Other Quality Attributes***

Website should be available to anyone.

## ***Documentation and Training***

Product will be self-documenting on user pages. Developers have an internal tool and project guide.

## ***External Interface***

External interfaces may be user interfaces or software interfaces.

### **User Interface**

The style of the interface should be professional and informative. User interface should be entirely text or visual. The interface should be intuitive and no training is required to use.

### **Software Interface**

All user operations will occur over HTTPS. On the client-side, the website uses javascript to interact with an API that provides all necessary information. On the server, the website interacts with the SQL database to provide API outputs in JSON, and all webpages in C# HTML templates.

# **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 of 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.

## ***Feature:*** Display of Banking Information

### **Description and Priority**

This should allow the user to understand and interact with their financial history. This should display recent notifications to the user, their previous transactions, and their account information. Other details about summaries or setting notification types should be allowed.

Cost: high

Risk: low

Value: high

### **Use Case:** See Account Transactions

Users should be able to see transactions’ amount, the location and the time of transactions.

### Use Case: See and Receive Notifications

Users should have the ability to receive notifications of any transactions made on their accounts as well as the location and time of the transaction.

### Use Case: Update Notification Types

Users should have notifications types to choose from. They should be able to choose between in-app or email notifications and change their preferences as needed.

### **Additional Requirements**

Include in this section additional functional and non-functional requirements not specified in the use case(s) above.

## ***Feature:*** Backend Access of Data

### Description and Priority

The backend of the website should manage user access to data, maintain an API layer, and interact with a database. The API layer should manage all data transactions and provide an abstraction of the database. The user also needs the ability to export their transaction history into a spreadsheet. All interactions need to be secure.

Cost: high

Risk: low

Value: high

### Use Case: User Authentication

The user should be able to authenticate themselves by providing a username and password in HTTPS basic authentication method to the server.

### Use Case: API Layer

The users' web page should be able to receive banking data from the API in order to display the information to the user.

### Use Case: Export User Transaction History into Spreadsheet

The user should be able to request transaction history and have a spreadsheet file provided by the server.

### Use Case: Database

Store user’s data in DB on a MySQL Server. This data includes usernames, passwords, and transactions’ details. The database should only be securely accessed by the server.