# Software Requirements Specification
# For
# Commerce Bank App

March 1, 2020
Version 1

Prepared by:
Tayler Singleton, Reed Heinrich, Connor Larson,
Samuel Feye, Landon Volkmann

# Table of Contents

## Revision History

| Version | Date | Name | Description |
|---|---|---|---|
| | | | |
| | | | |

# 1. Introduction

## *1.1. Overview*

The Commerce Bank App will be a web application available to users with an online account with Commerce Bank. The application will allow users to see their transactions, sort their transactions by date, export their transactions to a spreadsheet, add/edit/delete notification rules to display which transactions fit under a set criterion inputted by the user, pull/compare notification rules, see the number of times a notification rule has been triggered over the past month and year.

The document defined the requirements for the Commerce Bank web application being developed by group three. The purpose of this document is to represent the web application requirements in a readable way for the client to understand.

## *1.2. Goals and Objectives*

The main goals of Commerce Bank App product is to provide users an online summarization of their banking transactions and notifications based on their transactions:

1. Provide a web application for users to see their transactions and add transactions
2. The user should be able to set triggers for notification rules based on transactions.
3. The user should be able to receive notification based on default notification rules and created notification rules.
4. The user should be able to compare and pull different notification rules based on timeframes.

## *1.3. Scope*

Commerce Bank App provides users the ability to see their transactions and set triggers for notification rules notifying the user of when certain transactions fit within a defined criterion. The application also allows users to add transactions, though users can not delete transactions. The data will also be saved to a database for recurring reports to be created for the user to export their transactions to a spreadsheet.

## *1.4. Definitions*

**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.

**Trigger** - Tool should allow for configurable notification rules to be created to notify user when transactions fit into a set of criteria

**Notification Rule** – Determine which notifications (alerts) will be sent to the user.

### *1.5. Document Conventions*

Portions of this document will have certain actions in quotation marks, which means that action will be prompted within the web application by the user.

### *1.6. Assumptions*

It is assumed that the client will be able to operate and understand the web application using React and Redux frontend and MariaDB Database as the backend instead of .Net framework and SQLServer database.

## 2. General Design Constraints

### *2.1. Product Environment*

Commerce Bank App will be a web application built with React as the user interface, Redux as the state management tool for React and MariaDB Database as the backend.

### *2.2. User Characteristics*

Commerce Bank App Users: Commerce Bank user with an online banking account.

### *2.3. Mandated Constraints*

- .Net framework (optional)
- Must be a web application
- SQL server (optional)
- One CSS framework such as Bootstrap (optional)
- All meaningful data must be saved off in some accessible database. No references to external resources may be made.

The optional constraints are optional because we will receive limited support from the Commerce Bank team that has offered to help us if we opt not to follow those constraints.

### *2.4. Potential System Evolution*

Beyond the scope of the minimum requirements for the application, we have at least two stretch goals that must be completed. As a group, it is still an ongoing discussion in regard to which

stretch goals we will be implementing. One stretch goal we will assuredly be implementing will be using the Commerce Bank color scheme for styling our application. The second stretch goal is up for debate, however, we may very well end up implementing more than the minimum of two as our project develops naturally. We were tempted to create a Web API layer for backend interactions, to send notifications via email or text, and to use version control throughout the project, since we will be doing so through Github anyways.

# 3. Nonfunctional Requirements

## *3.1. Usability Requirements*

Task Efficiency: Users should be able to quickly see a summarization of their transactions and which notification rules have been triggered.

Ease of Learning: Users should be able to add/edit/delete notifications rules without technical assistance.

## *3.2. Operational Requirements*

The user's environment will be on their own mobile device or laptop, so the environment is up to the users themselves.

## *3.3. Performance Requirements*

The web application should be responsive being able transition from desktop layout to mobile version.

## *3.4. Security Requirements*

All user's password information should be masked. Users should only have access to transactions relating to their account number, which should be connected to their user account. The application will have data saved in the database to create recurring reports.

.

## *3.5. Safety Requirements*

The application will contain sensitive information of the user, so the main security risk is someone other than the desired user getting ahold of this information. Ideally, we need to do what we can to make sure that this can't happen.

## *3.6. Legal Requirements*

Users account numbers and transactions will not be visible to other users.

## *3.7. Other Quality Attributes*

- Simplicity where possible (login fields, overall design, transaction list).
- Easy to read

- Intuitiveness for new users
- Triggers don’t annoy user (ability to hide notifications)

## *3.8. Documentation and Training*

The application will have a user guide created explaining all features and how they will be used.

## *3.9. External Interface*

### 3.9.1. User Interface

The user interface will be simple and minimalistic using the Commerce Bank color scheme for styling. The user should be able to smoothly transition from area of the application to the next without trouble.

### 3.9.2. Software Interface

We will be using the React library for building our UI. React will be handy in making our UI smooth and look nice for the user.

# 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 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.

## *4.1. Feature: <Login Page>*

### 4.1.1. Description and Priority

This page will be resposible for handling user authentication into our system. It will hold two text input sections, a submit button and the commerce logo. After a user has been authenticated this page will redirect to the home page.

Cost: Med
Risk: Med
Value: High

### 4.1.2. Use Case: <Logging in>

1. User inputs text into the username field.
2. User inputs text into the password field.
3. User clicks the submit button
    a. User is authenticated
        i. User is redirected to Home page.

### 4.1.3. Additional Requirements

1. The password input field should mask the inputted text.
2. If user authentication fails

a. Login page is re-rendered and errors are displayed.

## *4.2. Feature: <DashBoard>*

### 4.2.1. Description and Priority

The dashboard will hold a summary of the notification triggers and their rules. This page will display counts for the number of times each notification rule has been triggered over a certain period of time.

Cost: Low
Risk: Low
Value: Med

### 4.2.2. Use Case: <Actor= user>

1. The user navigates to the dashboard.
2. A summary of triggered notifications will be displayed

### 4.2.3. Additional Requirements

1. User will have the ability to hide notification rules when the counts are zero.

## *4.3. Feature: <Home Page>*

### 4.3.1. Description and Priority

This page will hold a snapshot of data so a user can easily determine currect account balance and most recent transactions. This page needs to be clear, concise and overall easy to read. The home page will also hold the dashboard and a button responsible for exporting the transaction summary to a spreadsheet.

Cost: Med
Risk: Low
Value: High

### 4.3.2. Use Case: <Actor= user>

2. The user is redirected to the home page after logging in.
3. A daily snapshot will be displayed, showing current balance and recent transactions.
4. The dashboard will also be rendered.
5. The user clicks the “Export to Spreadsheet” button.
    a. A .xml file of transaction summary will be downloaded.

### 4.3.3. Additional Requirements

1. User has the ability to compare notification rules based on different timeframes.

## *4.4. Feature: <Add Notification Rule> CL*

### 4.4.1. Description and Priority

This tool will allow a user to add a new notification rule. A window will pop up and depending on the type of the rule they pick to add different fields will be rendered on the screen allowing the user to input the rules for it.

Cost: Med
Risk: Low
Value: High

### 4.4.2. Use Case: <Actor= user>

1. User clicks the “add notification” button within the notification page.
2. A window pops out onto the screen.
3. The first field is a drop down menu of the types of rules the user is able to add.
4. The user Selects “Transaction Amount alert”
    a. The field for “Amount” is rendered and the user enters a value.
    b. The fields for “start, end date, and indefinitely” Render and the user makes a selection
    c. The checkbox for active or not renders defaulted to active. The user can change this if they wish.
    d. The user clicks save.
    e. The rule is added to the database.
5. The user Selects “Balance alert”.
    a. The field for “Amount” is rendered. The user enters a value.
    b. The fields for “start, end date, and indefinitely” Render and the user makes a selection
    c. The checkbox for active or not renders defaulted to active. The user can change this if they wish.
    d. The user clicks save.
    e. The rule is added to the database.
6. The user Selects “Overdraft alert”.
    a. The fields for “start, end date, and indefinitely” Render and the user makes a selection
    b. The checkbox for active or not renders defaulted to active. The user can change this if they wish.
    c. The user clicks save.
    d. The rule is added to the database.
7. The user Selects “Transaction Name alert”.

    a. The field for “Transaction name contains” is rendered. The user enters a value.
    b. The fields for “start, end date, and indefinitely” Render and the user makes a selection
    c. The checkbox for active or not renders defaulted to active. The user can change this if they wish.
    d. The user clicks save.
    e. The rule is added to the database.
8. The user Selects “Recurring Alert”.
    a. The field for “How Many Occurrences before alert” is rendered. The user enters a value.
    b. The fields for “start, end date, and indefinitely” Render and the user makes a selection
    c. The checkbox for active or not renders defaulted to active. The user can change this if they wish.
    d. The user clicks save.
    e. The rule is added to the database.

### 4.4.3. Additional Requirements

1. The user can only add one rule at a time.
2. The user can cancel at any time by pressing the cancel button.
3. The end Date cannot be before the start date.

## *4.5. Feature: <Edit Notification Rule> CL*

### 4.5.1. Description and Priority

This tool will allow a user to edit an existing notification rule. A window will pop up to allow a user to edit the fields that go along with a notification rule. We will not allow them to change the type of rule just the fields that go along with one.

Cost: Med
Risk:  Low
Value: High

### 4.5.2. Use Case: <Actor=user>

1. The user clicks the edit button attached to every Notification rule.
2. A window pops up with the specific fields for each rule.
3. The user can change these fields
4. The users presses save. The notification rule is updated.

### 4.5.3. Additional Requirements

1. The user can only edit one rule at a time.

2. The user can cancel at any time by clicking the cancel button. No changes will be saved.
3. The user cannot change the type of notification after it has been created.

## *4.6. Feature: <Delete Notification Rule> CL*

### 4.6.1. Description and Priority

This tool will allow a user to delete a notification rule. This will live attached to each rule as a button on the main page and when clicked a window will pop up confirming they want to delete. Pressing yes will delete the notification rule from the database. Operation will not be undoable.
Cost: low
Risk: med
Value: high

### 4.6.2. Use Case: <Actor= user >

1. The user will click. delete button.
2. A window pops up and they can click yes delete or no to cancel
3. Pressing delete will delete the notification rule out of the database.

### 4.6.3. Additional Requirements

1. Confirming the delete is not undoable.
2. The user can only delete one rule at a time.

## *4.7. Feature: <Compare Notification Rules> CL*

### 4.7.1. Description and Priority

This tool will allow a user to take a notification rule and apply it to a past date and then return all the matching transactions that fit the rule.
Cost: med
Risk: low
Value: med

### 4.7.2. Use Case: <Actor=user>

1. This will be attached to notification with a button that says compare.
2. Clicking this will pop up a window with 2 date fields.
3. The user enters a past data range then clicks compare.
4. A table of past transactions will populate below allowing the user to see which transactions fit the rule.
5. The user can then press exit.

### 4.7.3. Additional Requirements

1. User can only test one rule at a time.
2. The user can only test rules that have been created.

## *4.8. Feature: <Hide Notification Rule> CL*

### 4.8.1. Description and Priority

This Feature will be a toggle attached to the notification rule. By clicking this toggle the notification will be either active or deactivated.
Cost: low
Risk: low
Value: med

### 4.8.2. Use Case: <Actor= user>

1. This toggle will live on the notification rule and also live in the edit tab.
2. The user toggles this on the rule itself.
    a. The database updates the rule as not active.
    b. The user toggles it again and it is active again.
3. The user toggles this on the edit tab.
    a. The database is not updated until the user presses save.

### 4.8.3. Additional Requirements

1. The user can only click 1 toggle at a time

## *4.9. Feature: Data Persistence with DB*

### 4.9.1. Description and Priority

This feature will ensure that account specific data persists between sessions. This data includes: Notification Rules, Account Information, and Notifications
Cost: low
Risk: low
Value: high

### 4.9.2. Use Case: Actor = user

1. Given the user logs in with valid credentials, the state of notification rules and notifications is maintained

### 4.9.3. Additional Requirements

1. None

## *4.10. Feature: Database API (stretch goal)*

### 4.10.1.Description and Priority

This feature will allow a developer to query the database for account specific data. This data includes: Notification Rules, Account Information, and Notifications
Cost: med
Risk: med
Value: low

### 4.10.2. Use Case: Actor = developer

1. Given a user supplies the api with proper credentials posts to the DB api with proper credentials and a valid query, they will receive a response fulfilling that query.

### 4.10.3. Additional Requirements

1. API will be Restful
2. Measures will be taken to prevent performance hits and other security risks from API based attacks