Software Requirements Specification
for
Commerce Bank

September 21 2020
Version 1

Prepared by:
David Johnson, Brittney MacLennan, Ashish Sharma, Tyler Wheaton, and Evan Wike

Table of Contents

Revision History

Version	Date	Name	Description

1 Introduction

1.1 Overview

This document will define the requirements for the Commerce Web application system that is being developed for bank customers. The purpose of this document is to ensure that system requirements are represented in a readable manner so that both clients and stakeholders can understand them and verify their correctness, while also ensuring that developers are able to design and implement a system from them.

This document will not address issues such as schedule, costs, development methods or phases, deliverables or testing procedures. A separate project document will be created to address these concerns in addition to a quality assurance test plan.

The Commerce Web application is a web-based tool that will allow bank customers to access their transaction details, as well as allow them to set specific triggers for notification rules and receive notifications based on these triggers. The system will also save this data to the database to allow recurring reports to be created. This web application will also allow bank customers to add transactions through the dashboard, which will then trigger any notifications based on triggers they have set.

1.2 Goals and Objectives

The goal of the Commerce Web application is to design an application that allows users:

- Function in an intuitive manner.
- Provide customers with an interface that allows them to easily receive alerts as well as manually add transactions.
- Configurable notification rules that notify users when transactions fit a set of criteria.
- Ability to pull/compare notification rules with different timeframes and be able to export to a spreadsheet.

1.3 Scope

At a minimum, the web application will:

1. be *responsive* and aesthetically pleasing.
2. allow customers to add, edit, and delete transaction notification triggers.
3. allow customers to add transactions.
4. allow customers to view their transaction history, and export it to a spreadsheet.

In order to accomplish these tasks, this project will require the building of a robust client. 70% of the work will go into building out the front end of the application (functionality, UI, etc.), another 10% will go into interfacing the client with the back end (data-access layer), another 10% will consist of setting up the database, database constraints, and user authentication systems on the BaaS (*Back End as a Service,* currently Firebase), and the remaining 10% will be dedicated to unit testing.

1.4 Definitions

actor	user, or other software system, that receives value from a use case.
BaaS	*"*Backend as a Service," a cloud service model that automates server side development and takes care of the cloud infrastructure.
customer	end user, the intended user of this software.
controls	the individual input elements in a user interface, such as buttons and checkboxes.
end user	customer, the intended user of this software.
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 to be used to express a requirement, but rather to specifically allow an option.
mobile-first	design philosophy that places mobile devices at the forefront of both design strategy and implementation; focuses on designing for the smallest screens first, before working back to laptops and desktops.
pagination	the process of splitting the contents of a website, or a section of its contents, into discrete pages.
product	the software system described in this document.
project	activities leading up to the production of the product described here. Project issues are described in a separate document.
RAD	"Rapid App Development," a form of agile software development methodology that prioritizes rapid prototype releases and iterations.
responsive	an approach to web design that allows web pages to render well on a variety of devices and screen sizes.
role	category of users sharing similar characteristics.
scenario	one path through a use case.
shall	adverb; used to indicate importance. Indicates the requirement is mandatory. Synonymous with *must* and *will* for the purposes of this document.
should	adverb; used to indicate importance. Indicates the requirement is desired, though not mandatory.

use case	describes a goal-oriented interaction between the system and actor, may define several variants, known as *scenarios*, that result in different paths through the use case, and usually in different outcomes.

1.5 Document Conventions

- Incomplete portions of this document will be marked 'TBD'. Each TBD item will have an owner and an estimated date for resolving the issue.
- Acronyms and *italicized* words used throughout the document will be defined in Definitions, section 1.4.

1.6 Assumptions

- It is assumed that the user is online and has a stable connection.
- It is assumed that the user has a web browser compatible with the web application.
- It is assumed the user has an account with the bank at the time of login.
- It is assumed the user has the valid login credentials.

2 General Design Constraints

2.1 Product Environment

Since this project will be on a standalone system that is not connected to Commerce during the duration of testing, we can only give details on the components that are available in our own system. We will be using Angular for web application development, SASS as a CSS preprocessor, and we will be using Firebase for our database. We are aware that support from Commerce can only be provided if we use .NET, and at the time of this document's creation, we have decided to use the two previously mentioned applications.

2.2 User Characteristics

For this document, we make the assumption that there will be two types of bank customers who use the system. These customers will fall into either a novice user or an experienced user.

Novice users are those who may not be technologically adept and have difficulty with computers or online web applications in general. Keeping these users in mind, we want to keep the user interface simple and aesthetically pleasing. It needs to be easy for them to navigate and instructions need to be clear so that they are not confused by the system.

Experienced users are those who are technologically adept and will easily understand the system as it is presented to them. While the system needs to be simple for novice users, experienced users will appreciate the more complex systems such as manually adding transactions to their transaction history.

2.3 Mandated Constraints

This application must:

1. be a web application (rather than a desktop application).
2. be built in a “newer” web development framework.
3. utilize at least one CSS framework.
4. ensure user passwords contain at least:
    a. 8 characters
    b. 1 uppercase letter
    c. 1 symbol
    d. 1 number
5. provide the following features:
    a. Home Page (including a Dashboard w/ notification summary)
    b. Login
6. achieve at least 10% code coverage for unit tests.

2.4 Potential System Evolution

We’re going with a modular design for the application, where it will consist of various different modules that can be swapped out or altered without affecting the structure of the overall application. Initially, it will include notification and transaction modules, but we would like to add more functionality; for example, the ability to set up automatic payments, support multiple bank accounts, and transfer money between them.

Currently, we’re using a *BaaS* as a means to get our application up and running quickly, using the *RAD* methodology; though, eventually we would like to create our own backend with ASP.NET and SQL Server. To allow for this future transition, we’ll be creating a data-access layer to interface the front and backends. This will allow us to swap out backends without having to rewrite any of the existing infrastructure of the application.

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 imagine 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.

The aim of the system is to not only be usable, but efficient for users that meet the goals and objectives (section 1.2). For the bank customers (section 2.2), we don't expect novice users to be efficient at using the system compared to the expert users. So the aim for novice users is a least a 70% efficiency rate and at least an 80% effectiveness in completing tasks. For expert users, we do expect them to be proficient in completing tasks as they have better understanding of how to use the system. Our aim for expert users is at least a 90% efficiency rate and at least a 95% effectiveness in completing tasks.

3.2 Operational Requirements

The user's environment is interacting with a electronic device (computer, phone, or tablet) connected to the web so the system depends on the user being able to use at least one hand.

3.3 Performance Requirements

While the main performance of the web application will depend on the end users system and internet connection, the project will be developed in such a way that the interface isn't overburdened with outside elements that will increase load times on the web page. Ideally, the web application should take no longer than 5 seconds to load while also allowing a margin of up to 10 seconds for slower internet connections. To help achieve and maintain this performance, the dashboard will have a minimal and simplistic design that is not reliant on images.

3.4 Security Requirements

Given its nature as a mobile banking tool, the application must:

- ensure the integrity of customer account information.
- behave correctly and predictably, through robustness and error prevention.
- abide by and promote industry standards when dealing with user data and user authentication.

3.5 Safety Requirements

The application must have a high degree of integrity when it comes to user account information and user data. To accomplish that, it must:

- restrict access to user data.
- prevent unauthorized access to user accounts through the promotion of strong passwords and passwordless authentication methods.
- prevent data loss and data corruption.
- have a consistent, predictable, and easily understood UI.

3.6 Legal Requirements

Bank account information is confidential and not visible to other users.

3.7 *Other Quality Attributes*

The web application at the time of release should be usable on all major browsers. It will require the user to be online in order to use the application. The application itself must be stable and responsive.

3.8 *Documentation and Training*

There are currently no plans to release the web application with any training instead the web application will be made to be intuitive and user friendly with minimal documentation to guide users through the interface.

3.9 *External Interface*

- **User Registration**
    - First time users shall be able to register for an account via various different types of *Authentication Providers*.
        - <u>Email and Password</u> - user creates an account with an existing email address and creates their own password. Verification emails will be sent.
        - <u>Federated Identity Provider</u> - users may create an account using either their *Google* or *Facebook* account.
- **Login**
    - Registered users shall be able to login to the application using multiple authentication providers.
        - Users are identifiable by the same *Firebase* user ID regardless of the authentication provider they used to sign in. For example, a user who signed in with an email and password can link a *Google* or *Facebook* account and sign in with either method in the future.
    - Users shall be able to <u>reset</u> their password, or <u>obtain their username</u>, by *email* or *answering security questions*.
- **Transactions**
    - Users shall be able to <u>view</u> their transaction history and <u>sort</u> it by date.
    - Users shall be able to <u>add</u> transactions, which will automatically trigger any associated notifications.
- **Triggers/Transaction Notifications**
    - Users shall be notified of transactions that fit into a set of criteria when they log in (i.e. “Transaction in Alaska”).
    - Users shall be able to <u>add</u>, <u>edit</u>, and <u>delete</u> notifications and triggers without technical assistance.

3.9.1 User Interface

In order to satisfy the various Security and Safety restraints necessary for a mobile banking application, the *User Interface (UI)* should be <u>consistent</u>, <u>predictable</u>, and <u>easily understood</u> for first-time users (who are presumed to be somewhere in the young-adult to adult age range). To that end, an intuitive interface will be the goal; where even a first-time user, regardless of age or prior experience, is able to navigate the core functionality of the application within a few minutes.

The theme of the application will be between Twitter (modern, fun) and a government website (authoritative, informative) -- offering a clean, polished, and professional experience to the user, similar to standing in the lobby of a bank -- meaning reduced clutter, authoritative fonts, and purposeful design. Overall, it should prioritize usability and accessibility to accommodate both the average and older user, while remaining informative.

Accommodating a variety of different users, across a multitude of devices, will be central to the design: navigation should be quick and easy to understand, with as flat of a hierarchy as possible, and the application will be *responsive*, utilizing a *mobile-first* approach. The application should render well on *at least* 75% of the most common screen resolutions (i.e. mobile, tablet, laptop, desktop).

The user interface should load efficiently and quickly, within 1 second of a user's request on an average internet connection. If necessary, user's with slower internet speeds should be accommodated when a lot of data is involved, via solutions such as *pagination*.

3.9.2 Software Interface

A data-access layer will interface with the Firebase SDK, which in turn, interfaces with both the Firebase Database and Firebase Authentication over HTTPS. Firebase is certified under major privacy and security standards, and supports General Data Protection Regulation (GDPR) in the EU, and the California Consumer Privacy Act (CCPA).

While we will be using a NoSQL database at first, we plan on creating our own back end in the future. The data-access layer should make the transition relatively seamless, though, we'll have to adapt it for a relational database.

4 System Features

4.1 Feature: Login Page

4.1.1 Description and Priority

Upon visiting the website, users will be met by the login page, where they'll be able to create an account if it's their first time, or login to their dashboard. Users may choose one of several different authentication options, including through the use of: an email/password combination, either their Google or Facebook account, or a link sent to their registered email. They'll also be able to reset their password here, should they happen to forget. Users will also be able to link their different login methods, so that they're able to login with any of them.

The login page will act as the main entry point of the application, making it a high value feature in and of itself. But, the most important aspect of this feature takes place behind the scenes. Given the nature of this application as a mobile banking tool, private user accounts are a necessity. All of their sensitive data will be associated with it, so security must be a high priority. Toward that end, we chose to promote several secure, passwordless options of signing in in the hopes of minimizing “P@ssword1.” The login also plays an important role as the “face” of the application, where first impressions are made.

The application also utilizes persistent authentication, where the user doesn’t have to login every time the visit. Their authentication will expire after a set period of time of inactivity, which will be a little shorter than normal for enhanced security.

Cost: *medium*
Risk: *low*
Value: *high*

4.1.2 Use Case: 1

User arrives at the website and:

1. Attempts to login, receives a message that there are no users registered with that email and is asked if they would like to create an account.
2. They then hit the “Create Account” button, and are taken to the “Registration” section.

4.1.3 Additional Requirements

TBD - no additional requirements at this time.

4.2 Feature: Registration

4.2.1 Description and Priority

Upon reaching the login page, first-time users, or users who wish to create another, separate account, will be able to navigate to the Registration page. Here, users will be able to sign up for an account using their email and password. Strict password requirements will be enforced for security. Upon registration, users will automatically be signed in to their new dashboards. They’ll also be sent a verification email to verify their accounts.

A method of account registration is necessary for the core functionality of the application, and is thus a high priority.

Cost: *medium*
Risk: *low*
Value: *high*

4.2.2 Use Case: 2

After attempting to login, the user is directed to the Registration page where they:

1. Are guided through the steps of creating an account.

2. The user is prompted to enter their email address, which is then verified as a real email.
3. The user is then prompted to enter a password.
4. But, “Passw0rd” is caught by the built-in password validation and they are told it isn’t strong enough.
5. After several different attempts at combining the number 1, an exclamation point, and their dog’s name, capitalized of course, they are able to move on to the password confirmation dialog.
6. Again, after a few attempts, they manage to type in the same password, and a green check mark appears - meaning they’re ready to roll.
7. The user then clicks the ‘Register’ button, and is taken to their new dashboard.

4.2.3 Additional Requirements

TBD - no additional requirements at this time.