Software Requirements Specification
For
Commerce Bank Web App

February 28, 2021
Version 1.0

Prepared by:

Zach Gharst
William Keke
Benaiah Kilen
Atticus Parris
Andrew Poitras

Table of Contents

Revision History

Version	Date	Name	Description
1.0	2/28/2021	Gather, Analyze, and Specify	Initial rollout of requirements.

1 Introduction

1.1 Overview

This document defines the requirements for the Commerce Bank Web App that is being developed as a practice for the software engineering processes learned within COMP-SCI 451R Software Engineering Capstone at the University of Missouri-Kansas City. The purpose of this document is to represent the web application's requirements in a readable way for clients and stakeholders such that they can verify the requirements for correctness. The requirements within should also be written such that the developers of the web application can use them to create a concise and comprehensible strategy for the design and implementation.

This document will not provide details on the project issues: schedule, cost, development methods, development phases, deliverables, and testing procedures. Those are addressed within the Project Charter and Testing Plan.

The Commerce Bank Web App is a web application that has a home page, allows a user to log in to their banking account, displays banking transaction details and summary, allows the user to set notification rules of their transactions, receive said notifications, and save data to a database to maintain a persistent state.

1.2 Goals and Objectives

The main goals of this application include:

1. Display a login page when the user first visits the website that allows the user to input an email and password that is tied to an existing Commerce customer account in order to start a user session.
2. Show summary of accounts on a home page once a user session has been created. The accounts are clickable so that the user may advance to the specific page tied to that bank account.
3. Detail a set amount of transactions on a bank account's page so that the user may see the relevant details of each transaction: date, type (credit/deposit), amount, description, and the new account balance.
4. Allow the user to set notification rules about their accounts' transactions. When a transaction is posted that is within those rules, the user shall receive a text/email/browser notification.
5. Implement at least two of the optional features/stretch goals detailed in sections 4.X.

1.3 Scope

The Commerce Bank Web Application will give Commerce Bank customers the ability to check their banking transactions from any web browser. Users will be able to check their transaction details and set notifications that will go off after a transaction has been

posted that triggers the notification. Users will be able to do all this by logging into the application with their Commerce Bank login.

Some of the functions of Commerce Bank that the web application will not have within its scope would be creating new customer accounts, creating new bank accounts, or advertising details about Commerce Bank products.

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 web application 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.

Customer account – the digital representation of a person who is a Commerce Bank customer.

Bank account – the digital representation of a checking or savings account owned by one or more customer accounts and a customer account may own more than one bank account.

1.5 Document Conventions

This document is a revolving work in progress -- it's expected that the requirements will change as a project changes; however, that doesn't mean that the requirements Anything within this document that isn't complete must be marked as such with an owner and estimated resolution date (example: [TBD: Zach, 03/28]).

1.6 Assumptions

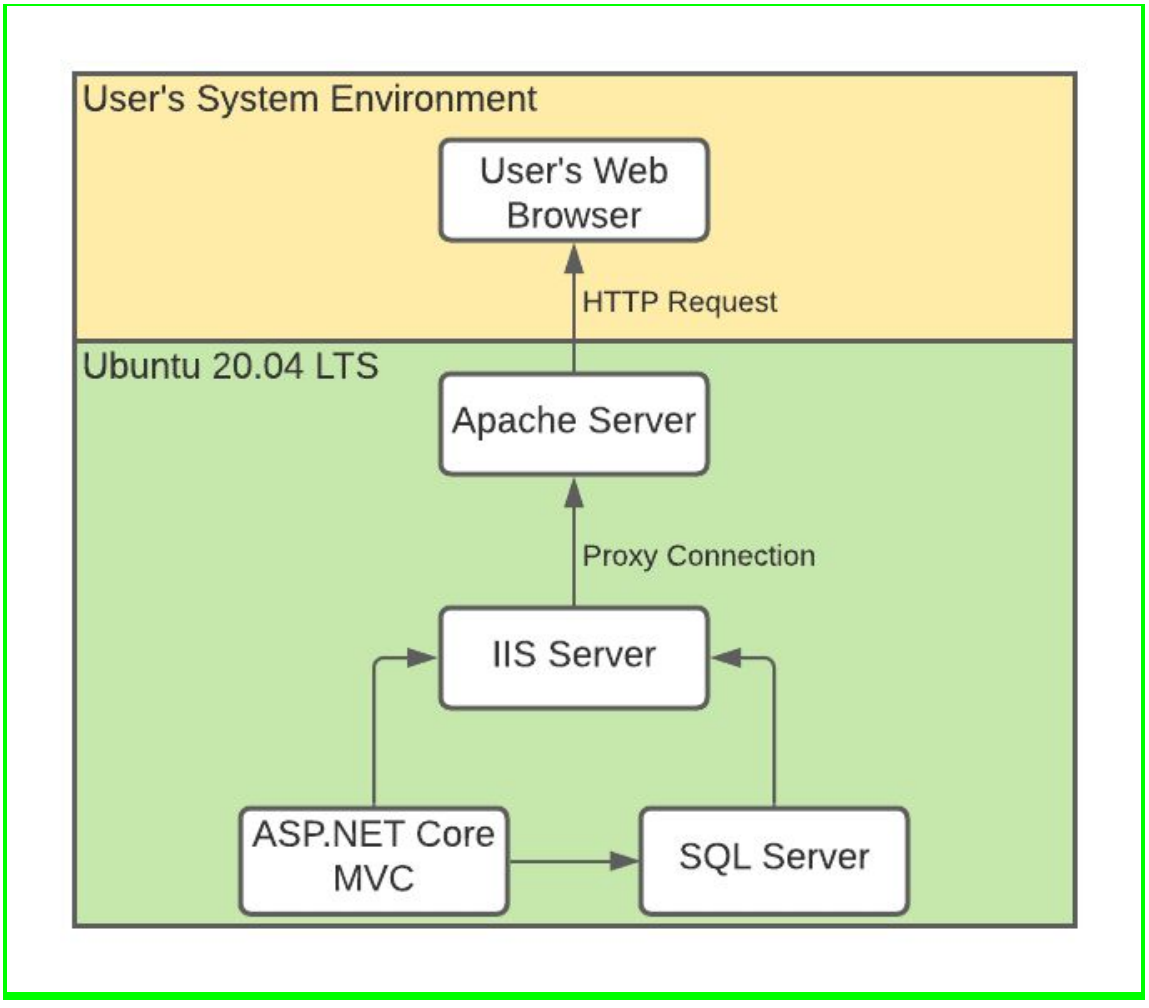
Users of this application are assumed to be a Commerce Bank Customer with access to a desktop, laptop, or mobile device with a web browser. Scheduled deadlines assume that team members have access to all their technology needs: ability to work individually on their machines, access to a testing database, and access to the Git repository to collaborate. Team members are also assuming that the Commerce Bank representatives will be able to help them with any feedback they may need in order to reach deadlines.

2 General Design Constraints

2.1 Product Environment

For the scope of this project, the web application will be isolated from external systems. However, Commerce Bank has existing software infrastructure and large databases of existing customer information that shall be considered when designing the application. Things should be scalable to accommodate the large amount of data and resources that the application would need if it was deployed in a Commerce Bank environment.

Deployment of the web application will be on a system running Ubuntu 20.04 LTS. The project will be limited to using ASP.NET Core MVC as the backend to create dynamic web pages, and the .NET framework will also access MS SQL Server for its persistent database needs. This technology stack will run on Microsoft's IIS Server, which will then be proxied to outside connections by Apache for security reasons as IIS is not intended for remote web authoring. Lastly, after the IIS Server links the dynamic data tier and presentation tiers together, it will be served to a user via their web browser on their device.



2.2 User Characteristics

Commerce Bank: Experts in financial matters. High priority user. Responsible for maintaining transactions of Customers. Well-versed in customer service and satisfaction. Should be assumed to have competence in an online banking environment.

Commerce Bank Customers: Patrons of Commerce Bank. High priority user. Places trust in Commerce Bank, its employees, and otherwise contracted staff. Assumes perfect execution of withdrawals and deposits of funds. Cannot assume technical expertise. Priority should be given to Customers that are inexperienced desktop users - the user experience of inexperienced mobile users is beyond the scope of this project.

Application Administrators: Responsible for maintaining high standards of application use and security. Responds to both Commerce Bank and its Customers in terms of troubleshooting the application. Moderate priority user. Highly tech-savvy.

2.3 Mandated Constraints

The web application shall...

1. ...use the .NET 5.0 framework.
2. ...be a web application (as opposed to a desktop application).
3. ...have at least 10% code coverage for unit tests.
4. ...use at least one CSS framework.
5. ...use SQL Server 2012 or newer.
6. ...use Commerce Bank branding style (colors and fonts) as detailed in the project description.

2.4 Potential System Evolution

Architecture should allow developers to easily construct additional notification rules if need be. We should also anticipate future development to include more avenues of receiving notifications (email, text, app...). The architecture should also support the ability to add additional functionality far beyond the scope of this project (in the distant future) such as customer support and new product registration.

3 Nonfunctional Requirements

3.1 Usability Requirements

The app shall be intuitive. Roughly ninety-five percent of users should be able to use all of the features of the application using their own intuition without needing to reference the documentation or extra sources. After deployment, surveys could be conducted to evaluate the user satisfaction of the web application.

3.2 Operational Requirements

The user will need either a computer or mobile device, an input device, a connection to the internet, and a web browser in order to use the web application. No other devices, such as audio output or webcam, shall be required.

3.3 Performance Requirements

The application shall be responsive. Loading any of the web pages in the application should take no longer than 3 seconds on the web application's end; however, loading times based on slow user internet speed should be taken into account when considering how much data is sent to a user. As a lot of user frustration with applications comes from slow response times, it's vital that the application is designed in such a way that it can scale horizontally and vertically.

3.4 Security Requirements

Cybersecurity is incredibly important in banking systems. When accessing the database, we should take precaution when using the database password. The password itself should not be present anywhere within the project outside of its designated configuration file (which is explicitly excluded from version control).

The SQL Server shall be configured to keep the best DBMS security practices in mind - physical security, encryption, user policy, value minimization, firewalls, regular audits, stored procedures, SQL injections, etc.

The passwords of customers' accounts shall never be stored in plaintext. They will be stored within the database only using an encryption algorithm. The encryption algorithm shall include a salt so that two users with the same password will have different password hashes in order to prevent the exploitation of rainbow tables.

3.5 Legal Requirements

The application needs to be ADA compliant to allow use by differently abled people.

For the ability to remember user sessions, the application may need cookie consent depending on the jurisdiction of the application's domain.

3.6 External Interface

3.6.1 User Interface

Since the average user of a banking site is a professional adult, the user interface should convey a professional feel to it. This can include but isn't limited to things like fonts, background designs, menu styles, and button styles.

The interface should be intuitive to use, 95% of users should not require additional help or training in order to effectively make use of all of the application's features. Increased technical ability should not influence a user's ability to use the application.

The web application should be accessible to both desktop and mobile users and the interface should remain cohesive and understandable for both platforms.

The interface should not be difficult to use for visually impaired users. This mostly includes using larger sizes for fonts and buttons so that they are easy to read and see.

3.6.2 Software Interface

The user will interact with the frontend webpage, which will go through an Apache proxy server that connects to the IIS server. These servers will work as an interface between the frontend webpage and the .NET Core API.

.NET Core will also interface with SQL Server to connect to the database holding the customer and transaction information.

The SQL Server will interface with the IIS Server to ultimately send data to the frontend webpage.

4 System Features

4.1 Feature: Customer Login / View Transaction Summary

Actors: Any Registered Commerce Bank Customer
Value: High
Cost: High

Use Case

Basic Path:

1.) Application prompts Customer to enter required Login and Password
2.) Customer enters correct Login and Password and clicks Login
3.) Application displays the Transaction Summary screen
4.) Customer selects “Log off”
5.) Application exits

Alternate Path:

1.) Application prompts Customer to enter required Login and Password
2.) Customer enters incorrect Login and/or Password credentials
3.) Application displays error message stating invalid credentials
4.) Customer may choose to Login again returning to step 1 or exit
5.) Application exits

4.2 Feature: Customer Notifications

Actors: Any Registered Commerce Bank Customer
Value: High
Cost: High

Use Case

Basic Path:

1.) Customer has logged in and is at the Transaction Summary screen (4.1)
2.) Customer selects to either Add, Edit, or Delete Transaction Notification
3.) When prompted Customer inputs the parameters/trigger for their Transaction Notification
4.) Application sets parameters/trigger for the Customer and returns to the Transaction Summary screen
5.) Application informs Customer with Notification if triggered

Alternate Path:

1.) Customer has logged in and is at the Transaction Summary screen
2.) Customer selects to either Add, Edit, or Delete Transaction Notification
3.) When prompted Customer inputs the parameters/trigger for their Transaction Notification
4.) Application prompts Customer that an invalid parameter/trigger has been set
5.) Customer may choose to input parameter/trigger again returning to step 3 or exit back to Transaction Summary screen

4.3 Optional Feature: Direct Deposit

Actors: Commerce Bank Customers, Customer's Employer
Value: High
Cost: Medium

Use Case

Basic Path:

1.) Customer has created an account and has been provided an account number.
2.) Customer provides routing number and account number to Employer.
3.) Employer uses this information to deposit funds into Customer account.
4.) Deposited funds are reflected on Customer's transaction summary.

* Important note here is that the employer need not have access to the account itself, but rather just the given credentials.

4.4 Optional Feature: Transfer Funds

Actors: Commerce Bank Customer
Value: Medium
Cost: Low

Use Case

Basic Path:

1.) Customer has created multiple accounts under the same login information.
2.) Customer navigates to transaction page of given account.
3.) Customer chooses "Manage Money" option.
4.) Customer selects alternate account and chooses to deposit or withdraw from this alternate account.
5.) Customer is prompted to input an amount of money.
6.) Money is transferred from one account to the other.

4.5 Optional Feature: Security Scan

Actors: Developers
Value: Medium
Cost: Low

Use Case

Basic Path:

1.) Developers are prepared to full release or update the application.
2.) A developer will run a security scan on a test of the new release.
3.) Using the results of the scan, the developers will fix any critical issues that were found.
4.) Repeat steps 2 and 3 until there are no more critical issues.
5.) Push new release/update to the master branch.

4.6 Optional Feature: Deployment

Actors: Operations Engineer, Developers
Value: Medium
Cost: Low

Use Case

Basic Path:

1.) Developers complete a pull request to the master (deployment) branch.
2.) The operations engineer runs a script (a Makefile) that pulls the branch, builds the web application from source into an object file, and uploads it to the appropriate location on the deployment server.
3.) The database is synced from the development version if necessary.
4.) The operations engineer restarts the IIS Server.
5.) Developers and the operations engineer manually test the live build to make sure everything is working as appropriate.
6.) Everything works as intended; consider deployment complete.

Alternative Path:

1.) Developers complete a pull request to the master (deployment) branch.
2.) The operations engineer runs a script (a Makefile) that pulls the branch, builds the web application from source into an object file, and uploads it to the appropriate location on the deployment server.
3.) The database is synced from the development version if necessary.
4.) The operations engineer restarts the IIS Server.
5.) Developers and the operations engineer manually test the live build to make sure everything is working as appropriate.
6.) The live build has a critical error, glitch, or security issue.
7.) The operations engineer reverts the build to the previously working live build.
8.) The operations engineer restarts the IIS Server.