**System Guide**

**Commerce Bank System**

04/26/2020

**Team Members**

Michelle Frost

Debbie Kirchner

Brian Roden

Linden Stirk

Issac Zeilinger

Document Control

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Change Date** | **Change By** | **Description of changes** |
| V1.0 | 04/23/2020 | Linden Stirk | Initial release |
| V1.1 | 04/24/2020 | Linden Stirk | Updated multiple sections |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**1 INTRODUCTION 4**

1.1 Definitions 4

**2 INSTALLATION AND SETUP 5**

2.1 C# and .NET Core 5

2.2 Xunit Unit Testing 5

2.3 Microsoft SQL Server 5

2.4 React 6

# Introduction

This document will outline the steps required to run the application and to maintain the Commerce Bank System. The Commerce Bank System is a web application that will allow customers who bank with Commerce to access information about their accounts online. This system will provide a home page, a transaction history, the ability to create and apply notification triggers specific to the customer’s preference, and the ability for the customer to export their transactions in a spreadsheet for uses outside of the application. This application will allow customers to view their transactions and receive alerts when unusual or potentially dangerous transactions are made.

## **Definitions**

**Commerce Bank System**: the web-based application that is being described within this document.

**Transaction:** a record of some amount of money that has moved either in or out of a person’s bank account

**Notification:** a visual indication that a target event has happened

**Trigger:** a specific event that causes a notification to appear

**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 another software system that receives value from a use case.

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

**Client:** the person or organization for which the Commerce Bank System is being

developed.

**User:** the person or people who will interact with the Commerce Bank System.

**Developer:** the person or people who are developing the Commerce Bank System.

# Installation and Setup

In order to run the Commerce Bank System, several pieces of software are required. The following is a list of the software that is used and an overview of how to set up each component. This guide does not cover how to set up web hosting, such as with Google Cloud or Amazon Web Services. However, the application can be run locally without issues for testing purposes.

The frontend and backend code are available in separate repositories.

The backend code is available here: <https://github.com/umkc-cs-451-2020-spring/semester-project-group-6>

The frontend code is available here: <https://github.com/umkc-cs-451-2020-spring/semester-project-group-6-UI>

## **C# and .NET Core**

The backend application is written in C# with the .NET Core framework. We recommend using Visual Studio 2019 as this the preferred environment in which to run and maintain the application. Open *CommerceApi.sln* to view the backend source code

## **Xunit Unit Testing**

Unit testing is a necessary part of software maintenance, and over the course of the application’s life, further testing will likely be required. A separate project containing Xunit tests is included in the backend source code. This project, *CommerceApiTest* references *CommerceApi.* Opening the *CommerceApi.sln* also opens the testing project’s solution.

## **Microsoft SQL Server**

This application uses a Microsoft SQL Server database. Therefore, Microsoft SQL Server is required to be installed. We also recommend installing Microsoft SQL Server Management Studio 18. Should the use of a different database system be required, a new access service class and corresponding methods will need to be modified in the C# *dao* package, and a new SQL script will need to be created in order to mimic the same structure as our database. Furthermore, using a different database will likely require a different access service. In the *Startup.cs* file, make sure to update the singleton service, *services.AddSingleton<IValuesDao, AccessService>();*

Database migration tools could potentially be used for this; however, for the remainder of this document, we will assume that this system uses Microsoft SQL Server.

The database must be created manually in Management Studio. The database name is ‘commerceDB’. For specific connection information or where to change it, refer to the connection SQL commands in the AccessService.cs file in the *dao* package. After creating the database, use the script *Table Creation.sql* to create the schema. There are three additional SQL files containing stored procedures. Run these scripts after creating the schema.

## **2.4 React**

In order to view the frontend application properly and interact with it as intended, Node.js is required. Download the LTS version from nodejs.org. After installing Node.js. To test the React app locally, open http://localhost: 3000 in a web browser. For maintaining the front end React app we recommend using Visual Studio Code.