**Architecture Document**

**Commerce Bank System**

04/05/2020

**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 | 04/2/2020 | Linden Stirk | Initial release |
| V1.1 | 04/3/2020 | Linden Stirk | Updated diagrams |
| V1.2 | 04/4/2020 | Linden Stirk | Fixed Process View; added descriptions |
| V1.3 | 04/5/2020 | Linden Stirk | Final release |
|  |  |  |  |

**Table of Contents**

**1 INTRODUCTION 4**

**2 DESIGN GOALS 4**

**3 SYSTEM BEHAVIOR 4  
  
4 LOGICAL VIEW 5**

4.1 High-Level Design 5

4.2 State Diagram 5

**5 PROCESS VIEW 6**

**6 DEVELOPMENT VIEW 7**

**7 PHYSICAL VIEW 8**

**8 USE CASE VIEW 8**

# Introduction

The purpose of this document is to outline the architecture for 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. This document includes a high-level view of the system as well as a process, development, physical, and use case view.

# Design Goals

The main goal of this application was to design a loosely coupled system. With five team members, this would reduce the workload on specific team members and allow multiple members to work on separate components at the same time. For example, the database access service is written to an interface, which allows the database to be switched easily requiring only a single line of code to be updated in a start-up file. This also allows multiple developers to work on the program with each using a different or mock database implementation.

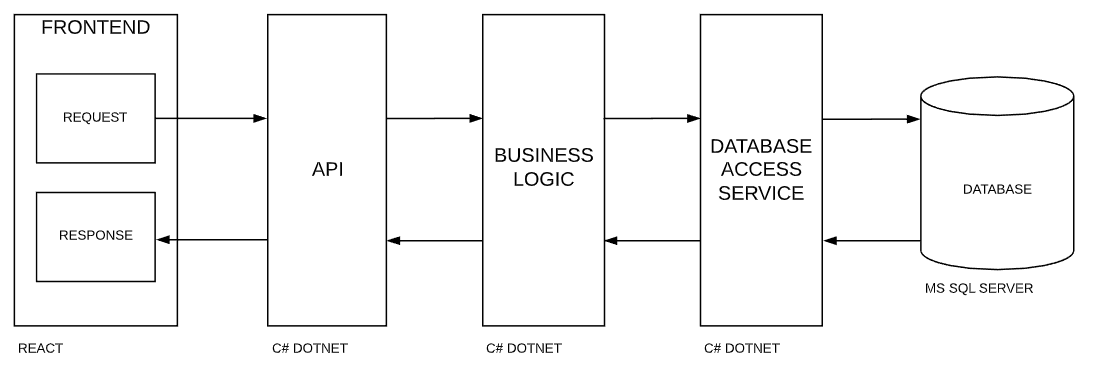
# System Behavior

The Commerce Bank System uses an n-tier architecture where the presentation, business logic, and data access layers, as well as the database, are separated. This aligns with our design goal of achieving loosely coupled components. The following sections provide various views of the system architecture, starting with a high-level logical view. For a more detailed discussion of the software requirements, see the Requirements Document.

# Logical View

## 4.1 High-Level Design

This high-level view maps the main layers of the Commerce Bank System. The frontend sends GET, POST, PUT, and DELETE requests to the API layer. This layer communicates with the business logic or service layer, which in turn, communicates with a database access service. The database access service communicates directly with the Microsoft SQL Server database. The frontend is written in React, and the backend is written C# with .NET.



## 4.2 State Diagram

# This state diagram describes the behavior of the system as a user logs in, navigates the site, and performs actions.

# 

# Process View

# This process view describes the processes and threads created during program execution. It also shows which process and object dependencies.

# 

# Development View

# This view better describes the development environment, showing specific module and layer organization of the system.

# 

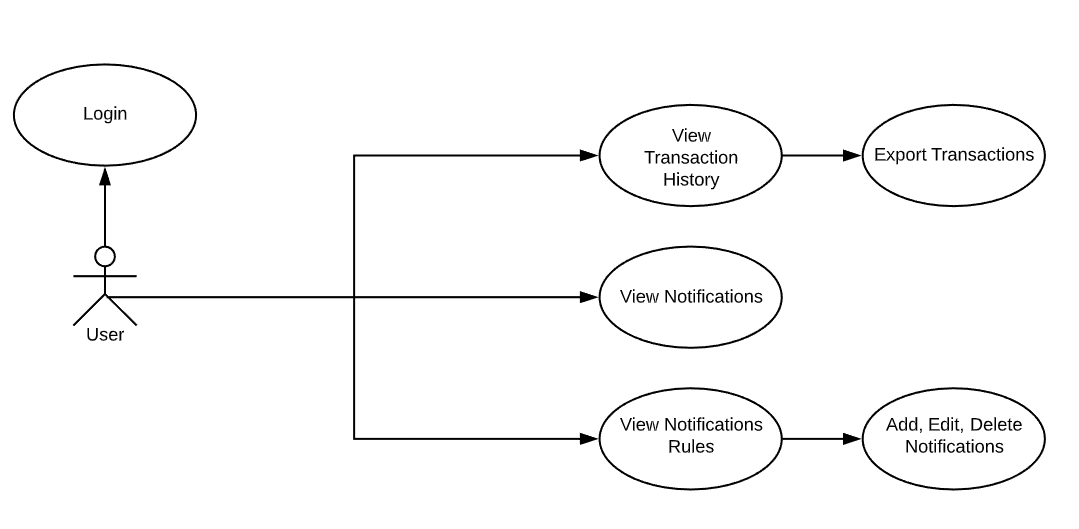
# Physical View

The following deployment view shows how the components are physically distributed among separate devices or logical devices.

# 

# Use Case View

The use case view maps a single user, which is an online Commerce Bank customer.



Login

The login use case allows the user to access their account. Upon entering the website, the system prompts the user to login with a username and a password. The user enters a username and password, which is validated by the system. Then the user has access to the account.

View Transaction History  
The view transaction history view allows the user to see a list of transactions and details of each transaction. Upon logging in, the user views the dashboard on the homepage, which displays transaction history. The system displays the transaction date, the balance at the time of the transactions, the type of transaction (deposit or withdrawal), the amount, and a description of the transaction. The user scrolls down to see all transactions. The system dynamically loads older transactions as the user scrolls

Export Transactions  
The export transaction use case allows the user to export a CSV file of the transactions. On the homepage/dashboard, the user clicks a download icon. A prompt asks the user to confirm the action. The entire transaction history is downloaded to a csv file. A dialogue box tells the user that the CSV was successfully exported.

View Notifications  
The view notifications use case allows the user to see which transactions were flagged by the system. On the homepage, the user clicks a notification icon to open a list of notifications. The system displays the number of times each notification rule has been triggered over the past month and year.

View Notification Rules  
The view notification rules use case allows the user to see which trigger conditions are set to flag transactions. On the notification view, the user clicks on an icon to open the list of rules.

Add, Edit, Delete Notifications  
This use case allows the user to be able to add, edit, and delete notification rules. The user clicks on the notification settings icon to enter the notification settings screen. The system displays the current notification rules associated with the account. The user clicks on a notification rule to edit the rule. The system displays options to edit or delete the rule. The user edits or deletes the rule. The system prompts the user to confirm the edit or delete. The user confirms and returns to the notification.

Export Transactions

The export transactions use case allows the user to download a CSV file of their transactions. On the homepage, the user clicks a download icon. A prompt asks the user to confirm the action. The entire transaction history is downloaded to a CSV file. A dialogue box tells the user that the CSV was successfully exported.