Version: 1.4

Online Banking Application

Architecture/Design Document

ID: OnBankArch14

Tarik Salay, David Tran, Jonathan Wolfe

University of Missouri - Kansas City
Computer Science 451R

Table of Contents

Change History

Version: 1.0
Modifier: David Tran
Date: 10/24/2019
Description of Change: Initial

Version: 1.1
Modifier: Tarik Salay, David Tran, Jonathan Wolfe
Date: 10/28/2019
Description of Change: Update Introduction, Design Goals, High-Level Design, Use Case View

---

Version: 1.2
Modifier: David Tran
Date: 10/30/2019
Description of Change: Update Versioning and Changelog

Version: 1.3
Modifier: David Tran
Date: 11/05/2019
Description of Change: Updates to Title page, Table of Contents, Low-Level Design added class diagram

---

Version: 1.4
Modifier: Tarik Salay, Jonathan Wolfe
Date: 11/08/2019
Description of Change: Update Versioning, Table of Contents, Introduction, Design Goals, System Behavior, Middle-Level Design, Database Schema diagram, Process View, Development View, Physical View and Use Case View

---

1 Introduction

This document describes the architecture and design for the Online Banking application being developed for Commerce Bank. This application pulls in transactions details and allows the user to set rules for alerts and receive notifications around them. The application saves data to a database and generates recurring reports.

The purpose of this document is to describe the architecture and design of the application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
- Developers – they want an architecture that will minimize complexity and development effort.
- Project Manager – the project manager is responsible for assigning tasks and coordinating development work.
- Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no one diagram or model that can easily express a system's architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the Online Banking application is described from 4 + 1 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Physical/Deployment View - describes the mapping of software onto hardware
4. Development View – how system modules map to development organization.
5. Use Case View – a walk through of a use for the application showing the interactions between high-level components to validate the design is following the functional objectives.

2 Design Goals

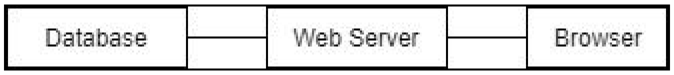
The Online Banking application will provide users with the ability to access information about their banking account from a web application using an internet browser. Users will be able to check their bank account balance, transactions and alerts by logging in to their account through the web app. Users will be able to specify rules to trigger alerts on transactions and receive the alerts through the application. The app will have a Login, homepage, transaction page, alerts page, reports page and rules page. In addition to these goals, users be able to easily navigate the app without guidance. The app will also secure information through encryption in the database and in connecting the user to the app. The app abstracts the connections between modules to provide better maintainability and security from SQL injection. For a more detailed account of software scope, goals and objectives, see the Software Requirements Specification For CS451 Group 6: Commerce Online Banking Application document ID:BankReq1.

3 Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction. In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

3.1 High-Level Design (Architecture)

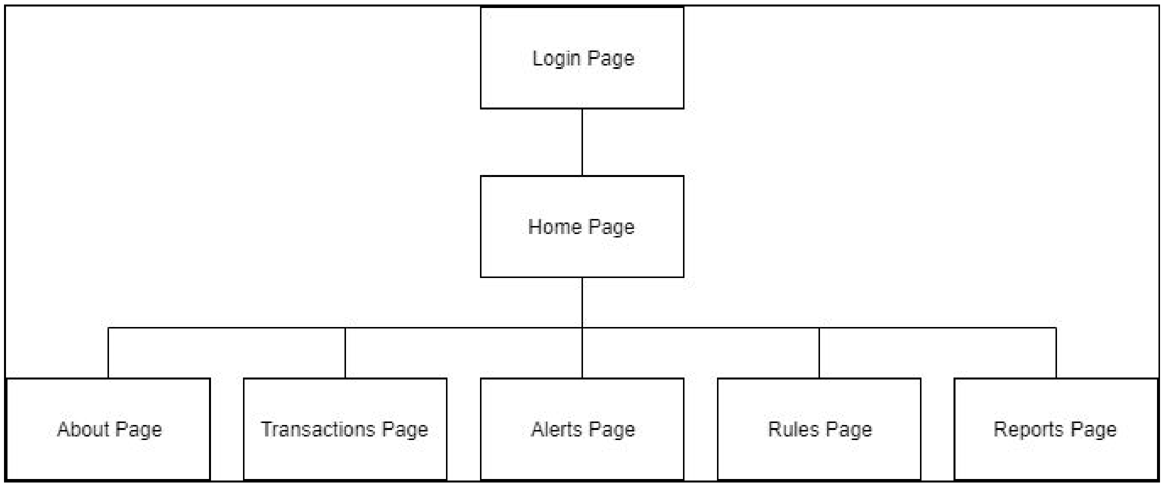
The high-level view or architecture consists of 3 major components:



- The **Database** is a central repository for data on transactions, alerts, customers, accounts, rules and reports as well as handling for generating alerts and scheduled generation of monthly and yearly reports.
- The **Web server** is the routing component of the application. It collects data from the database to present to the user and routes information from the user to the database.
- The **Browser** is the presentation layer of the application. It presents information to the user and reacts to user inputs.

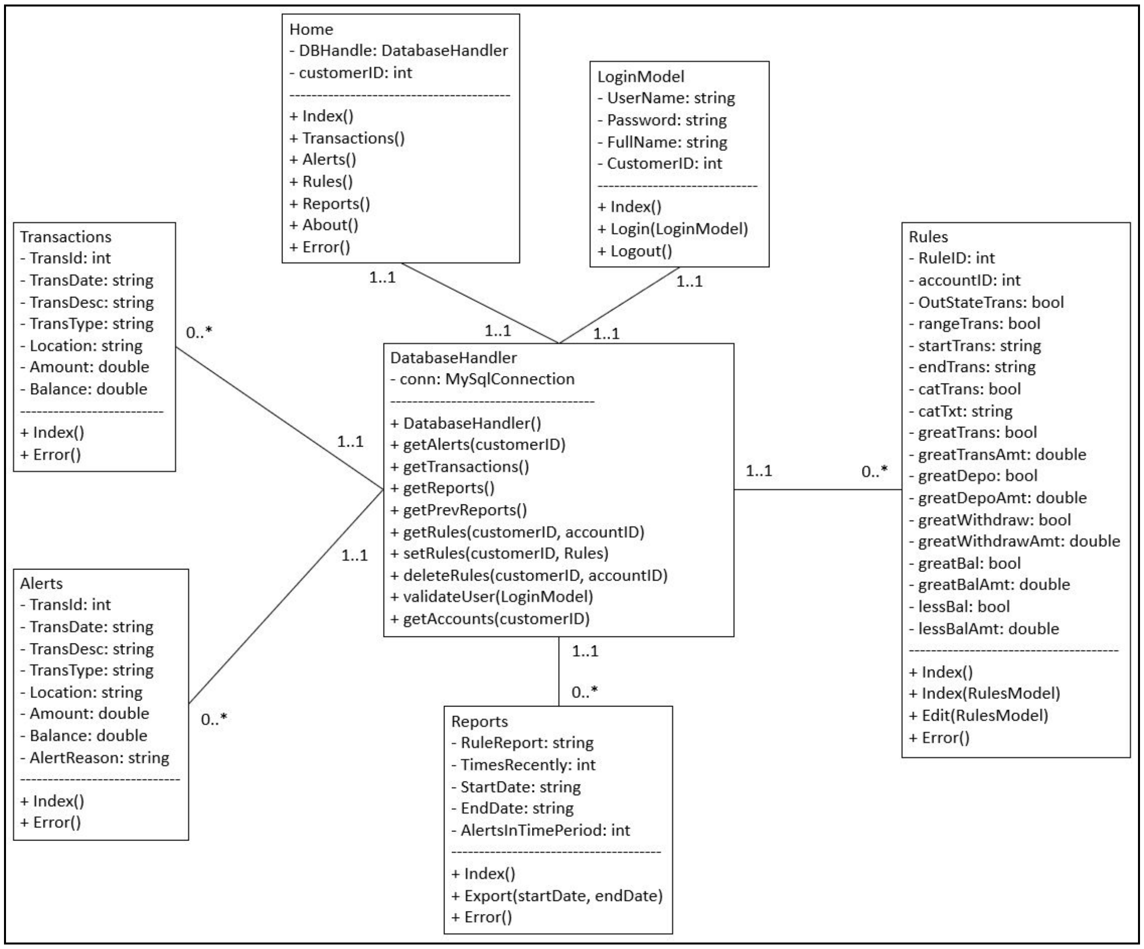
3.2 Middle-Level Design

Site Map

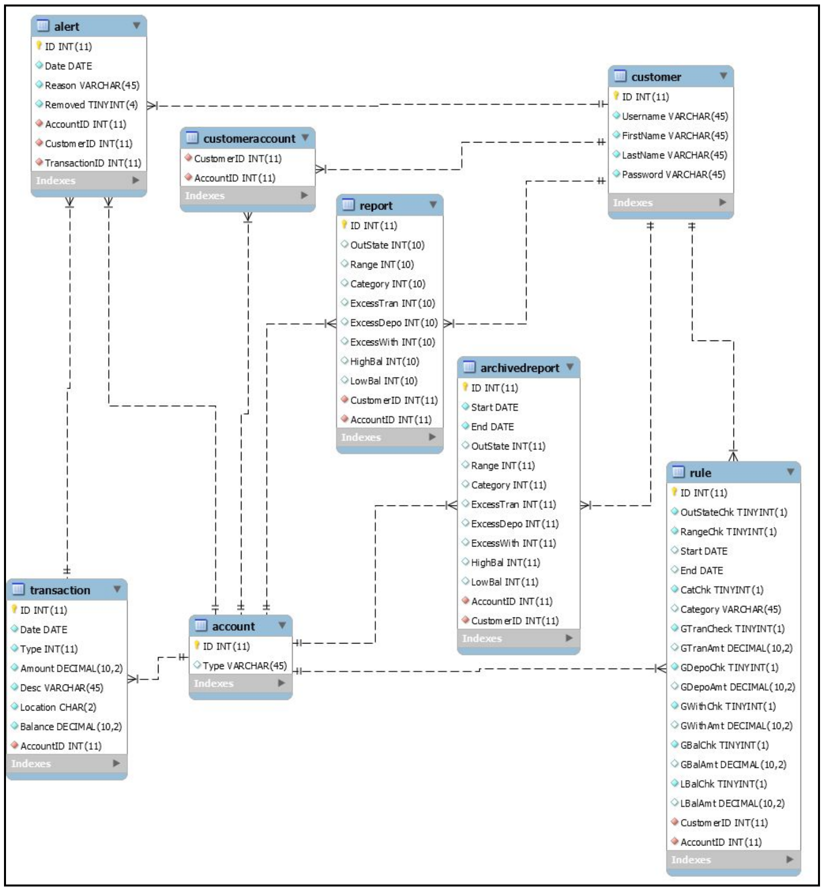


3.3 Detailed-Level Design

Class Diagram



Database Schema Diagram



4 Process View

4.1 Process View Description

The Process View is essential in understanding how the separate components and subcomponents communicate with each other in a concurrent application. By better understanding the necessary paths of communication between the components, it may be possible to optimize the data flow and storage of the application, as well as ensuring thread-safety.

4.2 Application Thread

This is the main thread created upon starting the web server. This thread is not created by the user. This thread handles program flow from one page to another, retrieving and sending information to the database and retrieving user input from the web pages.

4.3 Presentation Thread

This thread is created when the user accesses the web site and is created by the user. It is responsible for displaying content sent from the web server and providing an interface for entering information to send to the server.

4.4 Database Thread

This thread is created when the database server begins and is not created or directly connected to the user. This thread is responsible for retrieving data requested from the web server and storing data sent from the web server in addition to compiling periodic reports as well as generating alerts in response to new transactions.

5 Development View

The application is divided into models, controllers, views, data access layer and the database. The routing of the application between UI screens is handled by the controllers. The data exchanged between the database and the UI is held by models. The views provide an arrangement of controls of the UI from the models and routed from the controllers. The data access layer provides an interface to communicate from the application to the database. The database provides a data store. Both the database and the data access layer is designed by the database developers. The data access layer is a .NET connector API and is hosted with the web server application.The database is a separate database server from the web server and uses MySQL. The views are in HTML, CSS, Javascript and ASP.NET and are developed as part of the web server application by the web developers. The models and controllers are C# and .NET and are developed as part of the web server application by the application developers. Every class has a corresponding controller, model and view and ties together the application.

6 Physical View

The database is hosted from a UMKC school server. The application is currently hosted in the development environment and can be hosted from a separate server from the school server or within the school server. The UI is generated from the clients' browser.

7 Use Case View

The use cases described here refer to the Software Requirements Specification For CS451 Group 6: Commerce Online Banking Application document ID:BankReq1.

Use Case: 1

Description: Login to Homepage

Actors: Bank customers

Basic Path:

1. User opens Online Banking application via a browser and an internet connection.
2. System prompts user to login with a username and password from the login page.
3. The login page then collects the username and password and sends them to the database to verify the existence of the combination.
4. Once the user has been verified, the username is stored and used to generate a homepage and other following pages.
5. The homepage displays Account Balance, Recent Transactions, Alerts and a Navigation Bar to navigate to other pages of the site use case ends in success

Use Case: 2

Description: Set Up Alerts Based on Business Rules

Actors: Bank customers

Basic Path:

1. User login to homepage(Use case 1)
2. User selects on Rules from the Navigation Bar.
3. The rules page is then generated from the stored username and the first account associated with the username by querying the database for the associated rules and populating the controls with the rules.
4. The user selects from a dropdown the account to alter and the application will repopulated the page with the existing rules, if they exist.
5. The user checks rules to use a rule and enters rule specifications for each rule then submits the rules.
6. The application then checks if there are any rules being used and if not it removes any rules from the database but if there are any rules it alters or creates a new set of rules in the database from the user entries.
7. The page then sends a confirmation of the committed rules after the changes are finished.
8. The user then clicks a Logoff button to return to the Login page, remove stored credentials and complete the use case in success.

Use Case: 3

Description: Receive Notifications and View Alerts

Actors: Bank customers

Basic Path:

1. User login to homepage(Use case 1)
2. User selects on Alerts on the Navigation Bar.
3. The Alerts page is then generated from the stored username and the first account associated with the username by querying the database for the associated alerts and populating a list of alerts associated with the user.
4. The alert page displays a list of transactions that triggered a rule to alert the user and the rule that was broken.
5. The user can dismiss alerts by clicking a remove option which updates the alert in the database to as a removed alert, which prevents the alert from showing in future alert page calls.
6. The user then clicks a Logoff button to return to the Login page, remove stored credentials and complete the use case in success.

Use Case: 4

Description: Export Transactions & Reports

Actors: Bank customers

Basic Path:

1. User login to homepage(Use case 1)
2. User selects on Reports on the Navigation Bar.
3. The Reports page is then generated from the stored username and the first account associated with the username by querying the database for the associated reports and archived reports and populating the appropriate tables.
4. The reports page displays the current count of alerts based on the rule that was broken and previous reports based on the date of the counts of broken rules.
5. The user then selects to export the current report or an archived report.
6. The data associated with the report is then collected from the database then compiled into a CSV format and sent to the user.
7. The user then clicks a Logoff button to return to the Login page, remove stored credentials and complete the use case in success.