Commerce Bank Web Application Project – Group 4

Architecture/Design Document

**Table of Contents**

[1 Introduction 3](#_Toc68396974)

[2 Design Goals 3](#_Toc68396975)

[3 Logical View 4](#_Toc68396976)

[3.1 High-Level Design (Architecture) 4](#_Toc68396977)

[3.2 Mid-Level Design 5](#_Toc68396978)

[3.3 Detail Level Design 6](#_Toc68396979)

[4 Process View 7](#_Toc68396980)

[4.1 Process View Description 7](#_Toc68396981)

[4.2 Application Thread 8](#_Toc68396982)

[4.3 Database Thread 8](#_Toc68396983)

[4.4 Presentation Thread 8](#_Toc68396984)

[5 Development View 8](#_Toc68396985)

[6 Physical View 8](#_Toc68396986)

[7 Use Case View 9](#_Toc68396987)

[7.1 Use Case 1 9](#_Toc68396988)

[7.2 Use Case 2 9](#_Toc68396989)

[7.3 Use Case 3 10](#_Toc68396990)

[7.4 Use Case 4 11](#_Toc68396991)

[7.5 Use Case 5 12](#_Toc68396992)

[7.6 Use Case 6 13](#_Toc68396993)

Change History

**Version:** 0.1

**Modifier:** Ruby Rios

**Date:** 4/4/2021

**Description of Change:** Initial rough draft.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

This document describes the architecture and design for the Commerce Bank Project web application being developed for Commerce Bank. This application will allow Commerce Bank customers to log into their bank accounts and view their transaction history. In addition, users can set and receive notifications for transactions based on their preferences, and they can add in notifications within the application.

The purpose of this document is to describe the architecture and design of the Commerce Bank Project web 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 Commerce Bank Project web 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. Development View – how system modules map to development organization.
4. Use Case View – a walk-through of a use for the application that shows the interaction between high-level components to ensure that the design meets functional objectives.

# Design Goals

With there being no decisive measure between good and bad design, stakeholder priorities and feedback will be used in the design process to elicit the best design for all stakeholders. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vice versa. Therefore, the design for our application will be judged on how well it meets the following priorities:

* The design should focus on usability and minimalism. The main priority is to ensure that Commerce Bank users possessing a range of technological experiences can use the application effectively.
* The design should minimize complexity and development effort.
* The design shouldn’t inhibit reusability. The two previous design goals are more important, but the ability to reuse components is also desirable.

# 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.

## High-Level Design (Architecture)

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

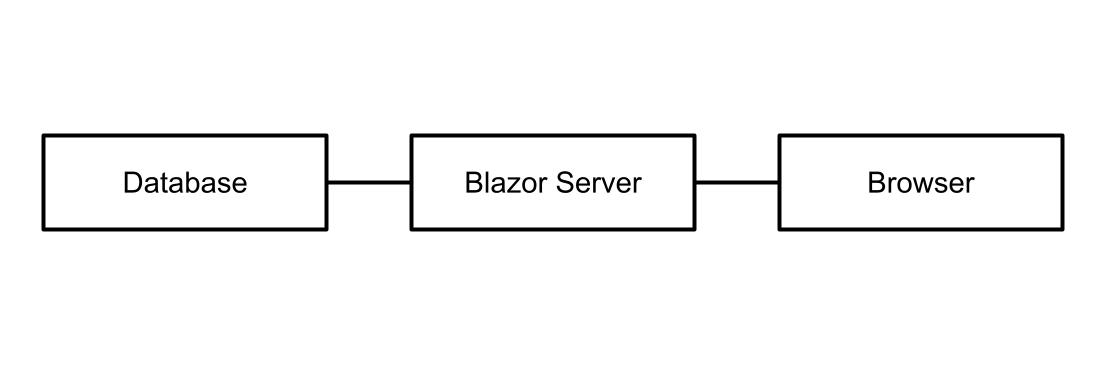


Figure System Architecture

* The **Database** is a central repository for data on Commerce Bank Accounts, transactions, and notifications. It also handles generating notifications based on rules that it stores for notifications.
* The **Blazor Server** collects data from the database using Dapper and displays it to the user. It also collects information from the user to store in the database, such as transaction details and rules for notifications.
* Given a position on Earth, the **Mapping Logic** will calculate nearby buildings.
* The **Browser** presents information to the user and responds to user input.

## Mid-Level Design

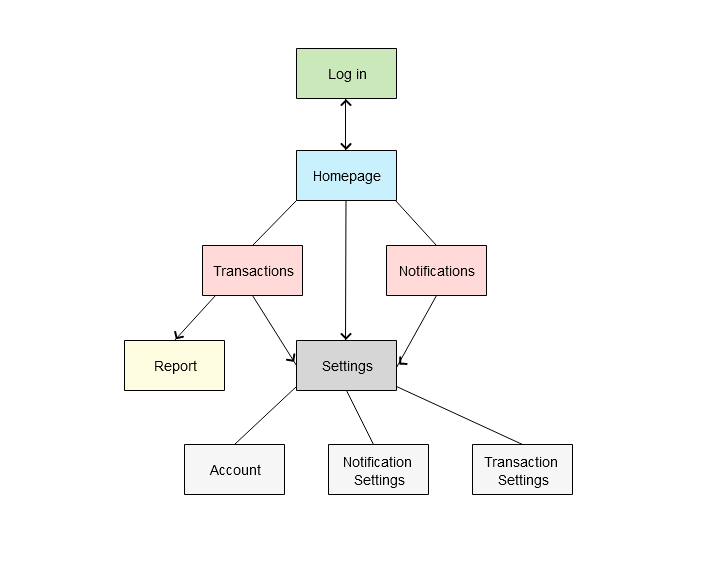
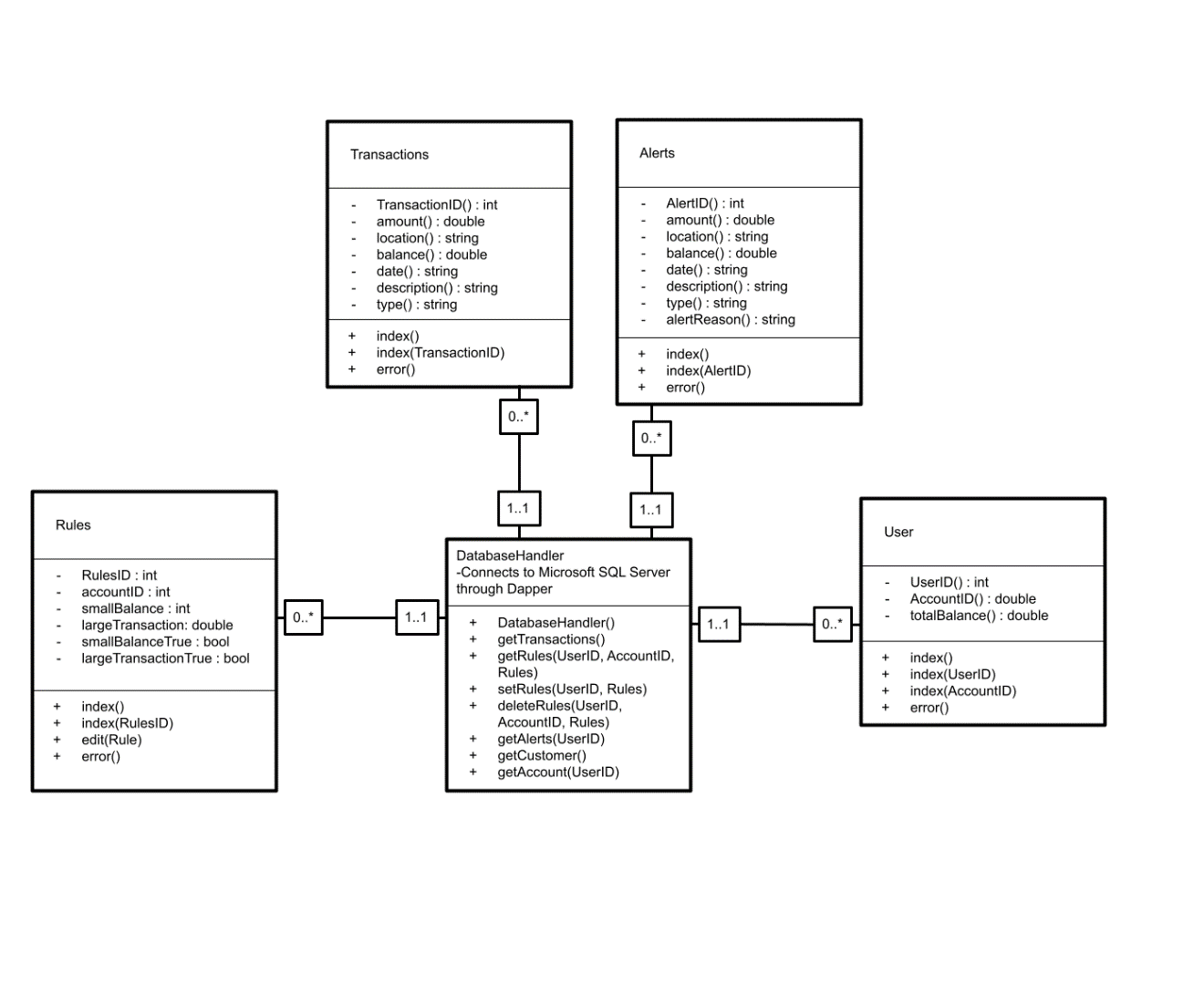
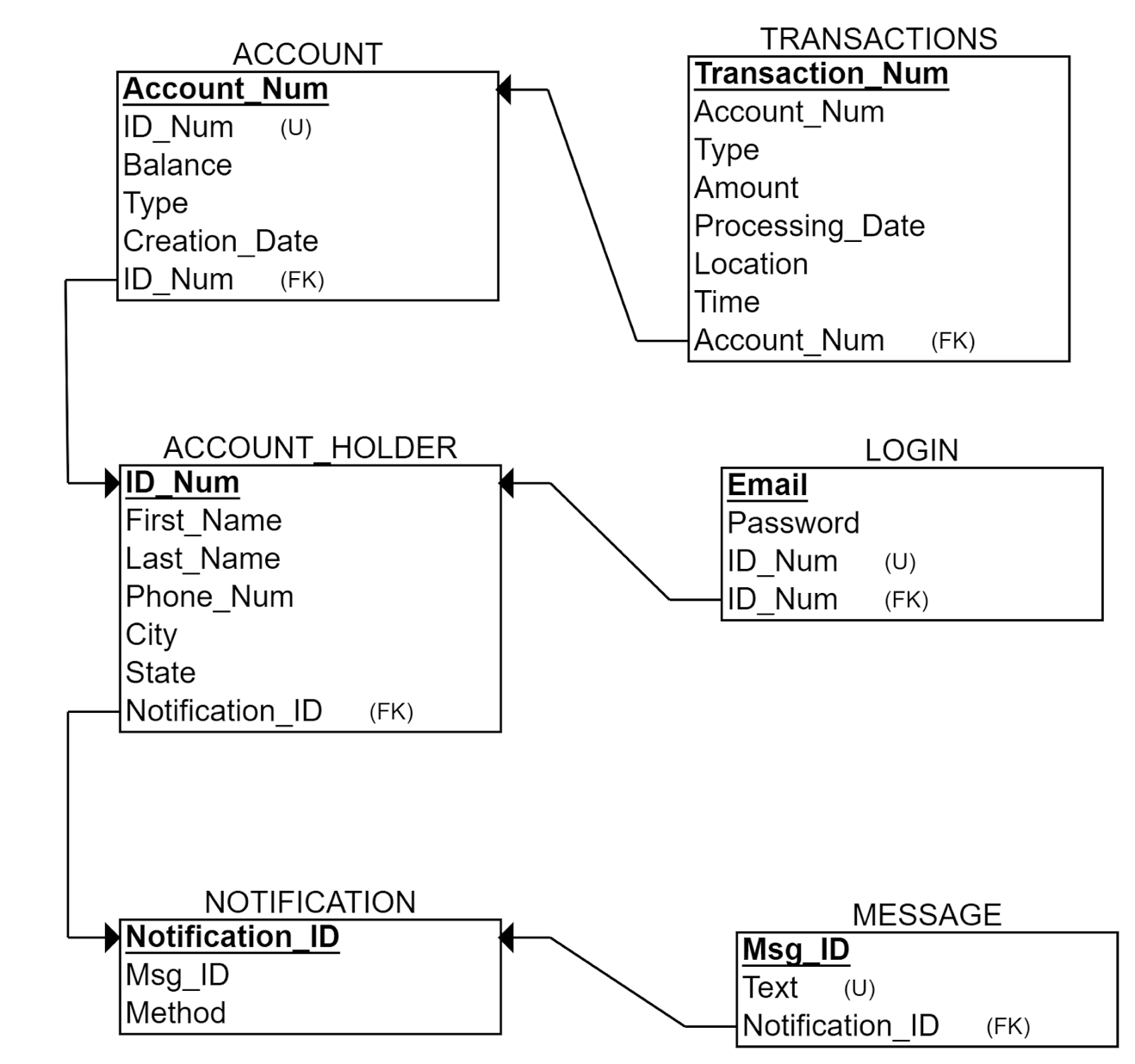


Figure 3 – Wireframe Diagram of the Commerce Bank Project

## Detail Level Design



**Figure 4 – Class Diagram**



**Figure 5 – Schema Diagram**

# Process View

## Process View Description

The process view helps in the understanding of how the components and subcomponents communicate with one another in the application. By having a better understanding of how this communication happens, it can be possible to optimize storage and dataflow within the application.

## Application Thread

This is the main thread of the application. It is created upon starting the web server, and it handles the flow of the program through the various web pages, the retrieval and sending information to the SQL server, and the retrieval of user input from the web pages.

## Database Thread

This thread is created when the SQL server is booted up, and it handles retrieving data requested from the web server and storing data passed in from the web server. It also handles creating notifications when transactions are entered in meeting rules stored in the database.

## Presentation Thread

This thread is created by the user when they access the web application. It handles displaying information from the web server to the user and collecting user input and sending it to the web server.

# Development View

The application can be divided into models, controllers, views, the data access layer, and the database. Controllers handle the routing of the application between web pages. Models handle exchanging data between the database and the front-end. Views handle a variety of controls of the front end from the models and routed from the controllers. The data access layer handles the communication of the application with the database. The database handles storing data. The database is designed by the database designer/lead, and the data access layer is designed through the cooperation of the database designer/lead and the web API lead. The database is a Microsoft SQL Server. The data access layer is made up of Dapper to connect the Blazor Server to the SQL Server. The views are designed by the UI/UX designer/lead. The views are coded in C# through Blazor Pages, with Bootstrap providing the framework. The models and controllers are designed by the cooperation of the web API lead and the UI/UX designer/lead, with assistance from the database designer/lead and the integration lead. The models and controllers are coded in C# through the Blazor server. Every class in Figure 4 has a controller, model, and view.

# Physical View

The application is currently hosted on a development environment. The database is hosted through Hamachi VPN. The UI is programmed in Blazor Pages with the Bootstrap framework, and is generated in the user’s browser.

# Use Case View

The use cases here are also described in the Software Project Requirements document.

## Use Case 1

***Description: User Login/Checks Transaction Notifications***

Cost: High

Risk: Medium

Value: High

Basic Path:

1. User opens up the web application.
2. The system prompts user to enter in a username and password.
3. User enters a correct username and password.
4. System displays the home page, which has information on transaction details.
5. User clicks Log off.
6. System Exits.

Alternate Path:

1. User opens up the web application
2. The system prompts users to enter in a username and password
3. User enters an incorrect username or password
4. System displays an error message “Invalid Email and/or Password.”
5. User may choose to login again, returning to step 2, or exit.
6. System Exits

## Use Case 2

***Description: Enter in a Transaction Manually***

Cost: Low

Risk: Low

Value: Low

Basic Path:

1. User successfully logs in.
2. User clicks icon for “Enter in a transaction”.
3. System displays a new page entry boxes for entering in transaction details, such as (but not limited to) date of transaction, where the transaction was made, amount of transaction, etc.
4. User enters in these details and then clicks “Enter”.
5. The system saves transaction details into the transaction database and returns user to home page.
6. User clicks log off.
7. System exits.

Alternative Path:

1. User successfully logs in.
2. User clicks icon for “Enter in a transaction”.
3. System displays a new page entry boxes for entering in transaction details, such as (but not limited to) date of transaction, where the transaction was made, amount of transaction, etc.
4. User does not fill in all of the details necessary and clicks “Enter”.
5. The system displays an error message “Please fill in all transaction fields.”
6. User may choose to fix the missing fields, returning to step 2, or exit.
7. System exits.

## Use Case 3

***Description: Set Notifications***

Cost: Medium

Risk: Medium

Value: Medium

Basic Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to set notifications.
5. System displays a new page with different buttons for various notification options (notification based on time, notifications based on amount, notifications to display at a certain time, etc.).
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User fill in all of the information required and then clicks enter.
8. The system saves this information in the notification database, and returns the user to the homepage.
9. User clicks log off.
10. System exits.

Alternative Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to set notifications.
5. System displays a new page with different buttons for various notification options (notification based on time, notifications based on amount, notifications to display at a certain time, etc.).
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User does not fill in all of the information required and then clicks enter.
8. The system displays an error message “Please fill in all notification fields.”
9. User may choose to enter the missing fields, returning to step 4, or exit.
10. System exits.

## Use Case 4

***Description: Set Notifications***

Cost: Medium

Risk: Medium

Value: Medium

Basic Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to set a new notification
5. System displays a new page with different buttons for various notification options (notification based on time, notifications based on amount, notifications to display at a certain time, etc.).
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User fills in all the information required and then clicks enter.
8. The system saves this information in the notification database and returns the user to the homepage.
9. User clicks log off.
10. System exits.

Alternative Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to set a new notification.
5. System displays a new page with different buttons for various notification options (notification based on time, notifications based on amount, notifications to display at a certain time, etc.).
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User does not fill in all the information required and then clicks enter.
8. The system displays an error message “Please fill in all notification fields.”
9. User may choose to enter the missing fields, returning to step 4, or exit.
10. System exits.

## Use Case 5

***Description: Edit Notifications***

Cost: Medium

Risk: Medium

Value: Medium

Basic Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to edit a notification.
5. System displays a new page with different buttons for various notification options.
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User fills in all the information required and then clicks enter.
8. The system saves this information in the notification database and returns the user to the homepage.
9. User clicks log off.
10. System exits.

Alternative Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”.
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to edit a notification.
5. System displays a new page with different buttons for various notification options.
6. Depending on which option the user selects, new options will be displayed to help the user with setting that type of notification and an enter button.
7. User does not fill in all the information required and then clicks enter.
8. The system displays an error message “Please fill in all notification fields.”
9. User may choose to enter the missing fields, returning to step 4, or exit.
10. System exits.

## Use Case 6

***Description: Delete Notifications***

Cost: Low

Risk: Low

Value: Low

Basic Path:

* 1. User successfully logs in.
  2. User clicks icon for “Edit Notifications”.
  3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
  4. User clicks to delete a notification.
  5. System displays a pop up, asking whether the user would really like to delete the notification, with buttons for yes and no.
  6. User clicks yes.
  7. The system deletes the notification from the notification database and returns the user to the homepage.
  8. User clicks log off.
  9. System exits.

Alternative Path:

1. User successfully logs in.
2. User clicks icon for “Edit Notifications”
3. System displays a new page with current notifications and buttons to delete and edit current notification, and a button to set a new notification.
4. User clicks to delete a notification.
5. System displays a pop up, asking whether the user would really like to delete the notification, with buttons for yes and no.
6. User clicks no.
7. The system does not edit the notification database and returns the user to the homepage.
8. User clicks log off.
9. System exits.