[Group 1 Commerce Bank Project]

Architecture/Design Document

Table of Contents

Change History

Version: 1.1
Modifier: <Sushant Acharya, Kole Keeney, Ozzie Loewen, Thomas Star-Timberlake>
Date: 04/04/2021
Description of Change: <First Agreed Upon Documentation>

Version: <1.2>
Modifier: <Ozzie Loewen>
Date: 04/04/2021
Description of Change: <Use Cases Added>

Version: <2.0>
Modifier: <Ozzie Loewen>
Date: 05/03/2021
Description of Change: <Amendments>

1 Introduction

This document describes the architecture and design for the Commerce Bank Project application being developed for Commerce Banking Members. Commerce Bank would like to provide a web application for their customers to view and edit their transactions and the details relating to them. Commerce Bank has the information their users are wanting but want it in user-friendly application that allows them to view their transaction history as well as add their information. The purpose of this document is to describe the architecture and design of the Commerce Bank Project 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. The want the program to be expandable and adaptable based on the needs of the user and business.
- Project Manager – the project manager is responsible for assigning tasks and coordinating development work. She wants an architecture that divides the system into components of comprehensible size and with complexity fitting of the features. These tasks should be designed with minimal dependency, but with reference to related tasks as needed. For this to happen, the modules need well-defined requirements and interfaces. Since the individuals working on this project are not specialized in a particular skill or technology, modules should be designed to the best ability of the individual, with the idea of straightforward delivery.
- Maintenance Programmers – they want assurance that the system will be maintainable, and it will have the ability to evolve.

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 application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
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 – the use case view is used to both motivate and validate design activity. At the start of design, the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

2 Design Goals

Good and bad design are subjective and hard to distinguish because of that fact. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The design priorities for the Commerce Bank Project application are:

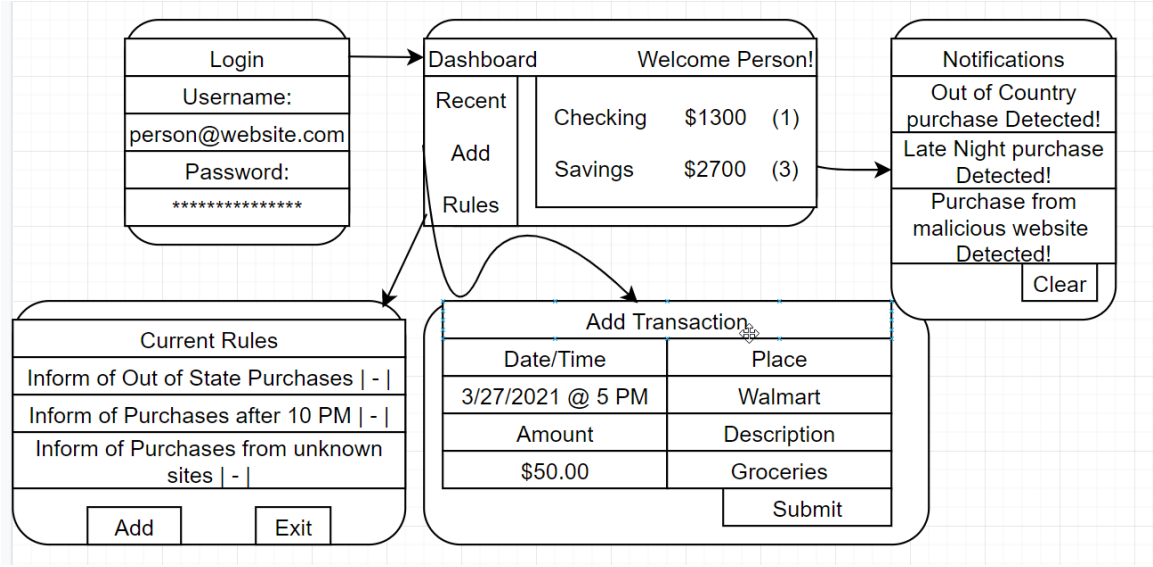
- The design should minimize complexity and development effort.
- The design should allow customers to view their transactions. The user should be able to see their transactions in a way that is appealing and has sense and meaning behind it.
- The design should allow customers to add their transactions. Along with being able to view their transactions, the user should be able to add transactions they have made. They have a specific screen that allows them to 'create a transaction' for it to be added to their transaction history.
- The design should allow customers to edit their transactions. A user should see their information and make changes that apply to their account.
    a. Should they accidentally fill out the wrong information, they should be able to edit it to accurately match the transaction.
- The design should be easy to use for all users. The design should be intuitive and have a logical flow. Things should be correctly labeled so users can easily navigate through the site.
- The system should be useful. It should provide the user with the ability to accomplish their transaction goals in easy and functional way.
- The user should be able to have notifications that fit their needs. The user will provide access to their email and phone number so they can best select notification types that fit their needs.
- The notifications are editable so the user can select what works best for them.

3 System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The system behavior diagram presented here starts with a walkthrough of the expected system behavior in order to set the stage for the

architecture description that follows. For a more detailed account of software requirements, see the requirements document. The architecture should also demonstrate the capability ease of implementing additional aspects such as unit testing and extendibility from additional user preferences that could be created in the future. Said design should make these prospective features easy to implement by Commerce Bank administrators among other operating staff.

Figure 3.1



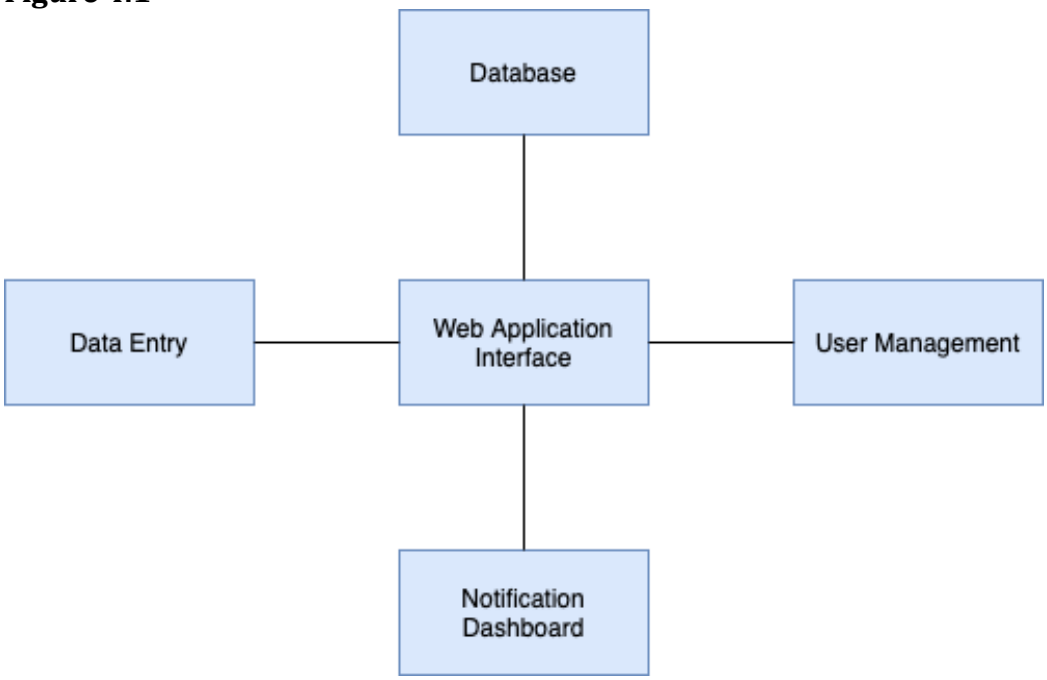
4 Logical View

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. The logical view begins with a brief high-level overview, and increasingly gets more specific as we break down individual components.

4.1 High-Level Design (Architecture)

The high-level view or architecture consists of 5 major components:

Figure 4.1



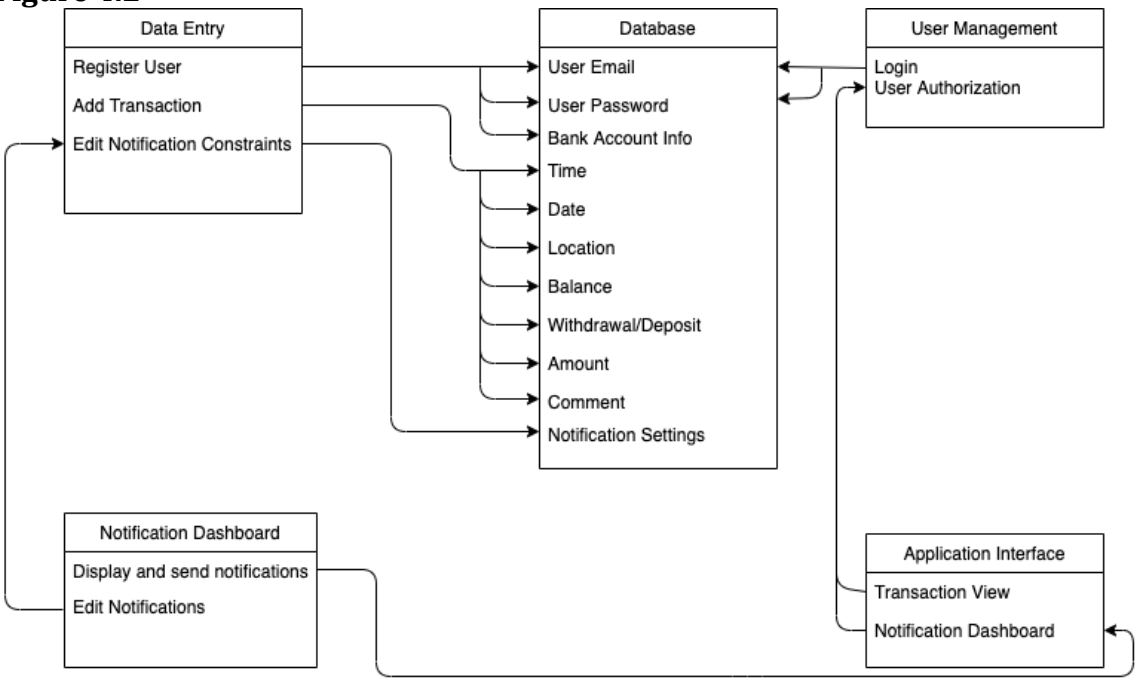
System Architecture

- The **Notification Dashboard** displays any notifications based on custom constraints entered by the user or an admin. These notifications are triggered once transaction details meet the constraint criteria.
- The **Database** is a central repository for data on users and their corresponding notifications and transactions.
- The **Data Entry** level interacts with the database and allows the user to alter transaction and notification information in the database through the UI. This is necessary to edit and interact with the Notifier application.
- The **User Management** layer makes sure that typical users only have access to certain parts of the application (their own notifications and transactions) and allows admins to enter their own constraints, view all user data, and amend transactions if needed.
- The **Web Application Interface** is the main driver of the application. It displays information to the user and responds to user inputs.

4.2 Mid-Level Design

Provided is a mid-level design diagram which shows how our major components interact within our project.

Figure 4.2



4.3 Detailed Class Design

Provided are some key associations, attributes and methods for main classes in this project. Due to the nature of ASP.Net and EntityFramework, some features of our project come handled, specifically a lot of the user validation. These complex classes and models are constantly working in the background to provide support. An overview of our current database has also been provided, and again, due to the nature of our framework, a lot of these tasks are handled independently, despite all being interwoven within our project.

Figure 4.3.1

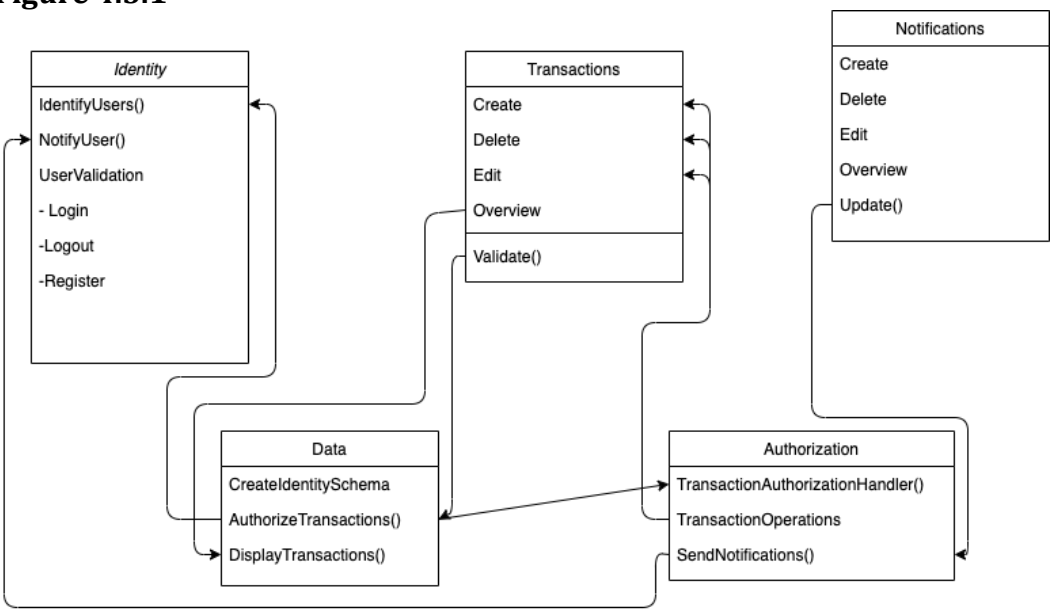
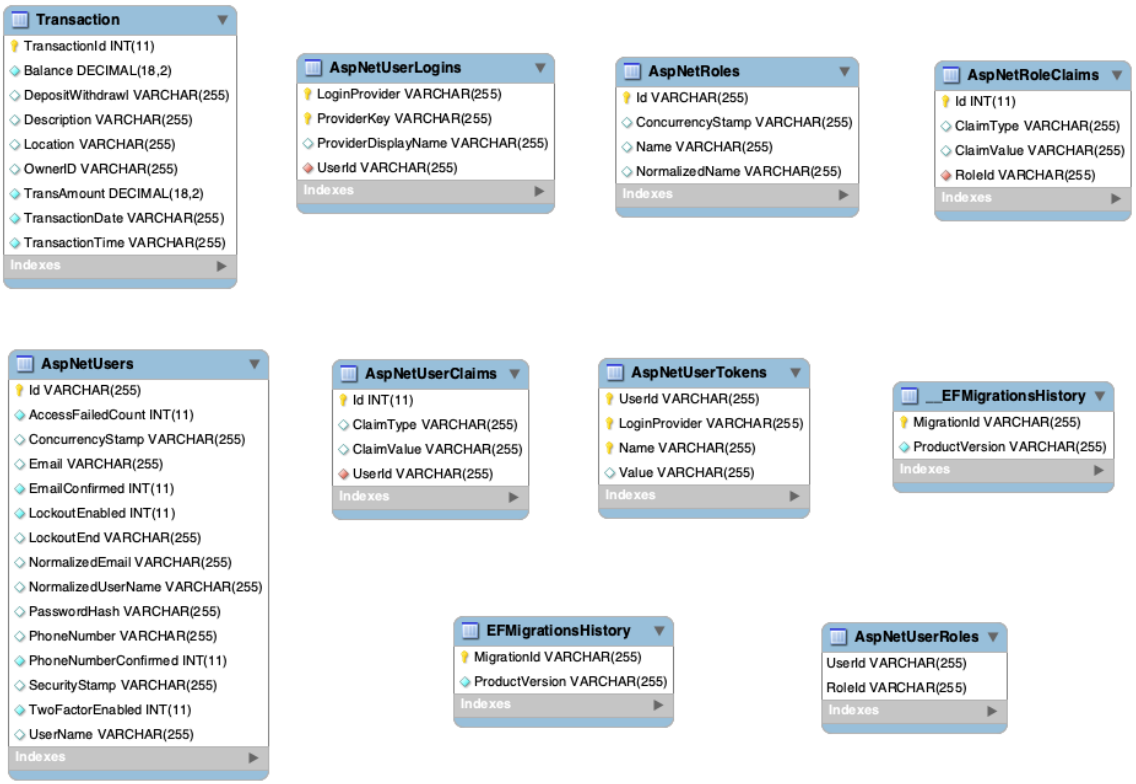


Figure 4.3.2



5 Process View

There are three threads of control in the application.
Front end thread

The front-end thread is hugely user dependent. It requires the user to provide input to proceed. Whenever a user provides input, depending on the input, the front end either loads the appropriate next page or sends that information for the back end to process.

Back-end thread
The back end is the bridge joining the front end with the database. It is hugely dependent on the information sent from the front end and the data stored in the database. Depending on the information provided by the user and relayed by the front end, the backend executes various functions that do one of four things: creates, updates, reads or deletes data from the database. The back end sends data back to the front end as well. Any information required by the user is pulled from the database by the backend and relayed back to the front end for the user to view.

Database thread
The main purpose of the database is to store the information. It stores any new information provided by the user through the front end interface or any data processed by the backend. Along with that, any data the backend needs to be processed on or relayed back to the front end is pulled from the database.

6 Physical View

Provided is an approximation of the deployment of the various components required for the Notification system. The majority of the business end of the system will be hosted on a singular Web Server containing the IIS Server and Kestrel server to handle the HTTP requests for the .NET Application Code. The SQL database will be hosted on a different physical server on the same network, passing data to the Web Server on request using the SQlite database software.

Figure 6.1

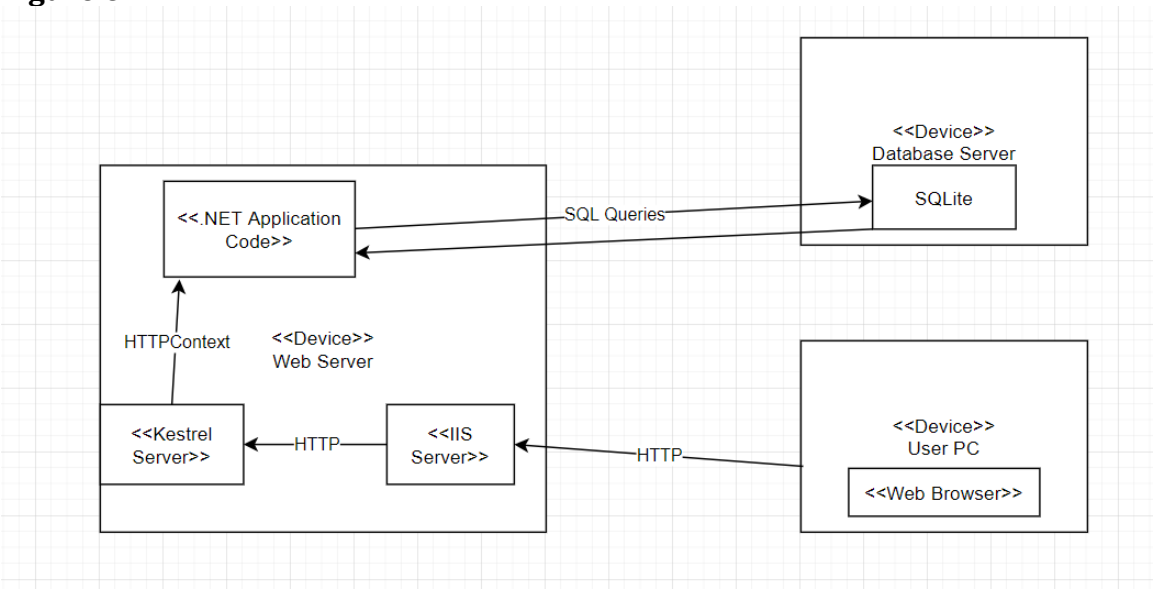
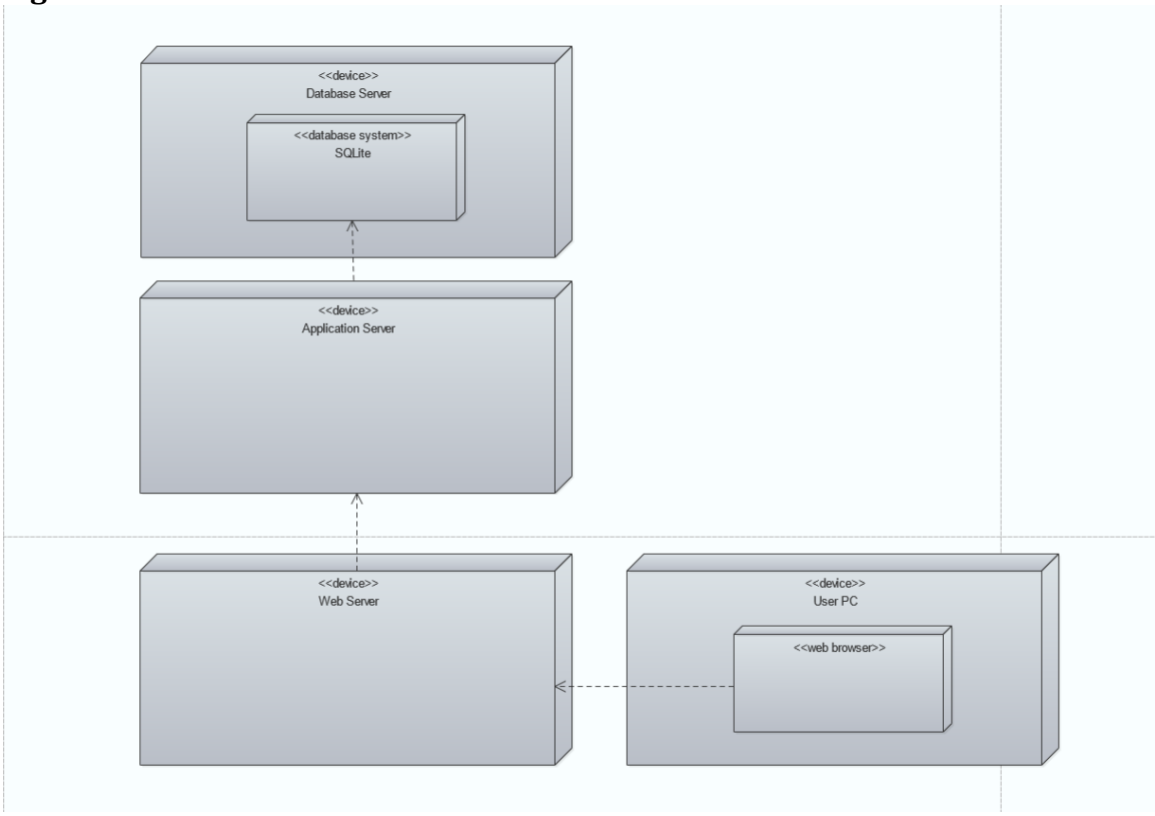


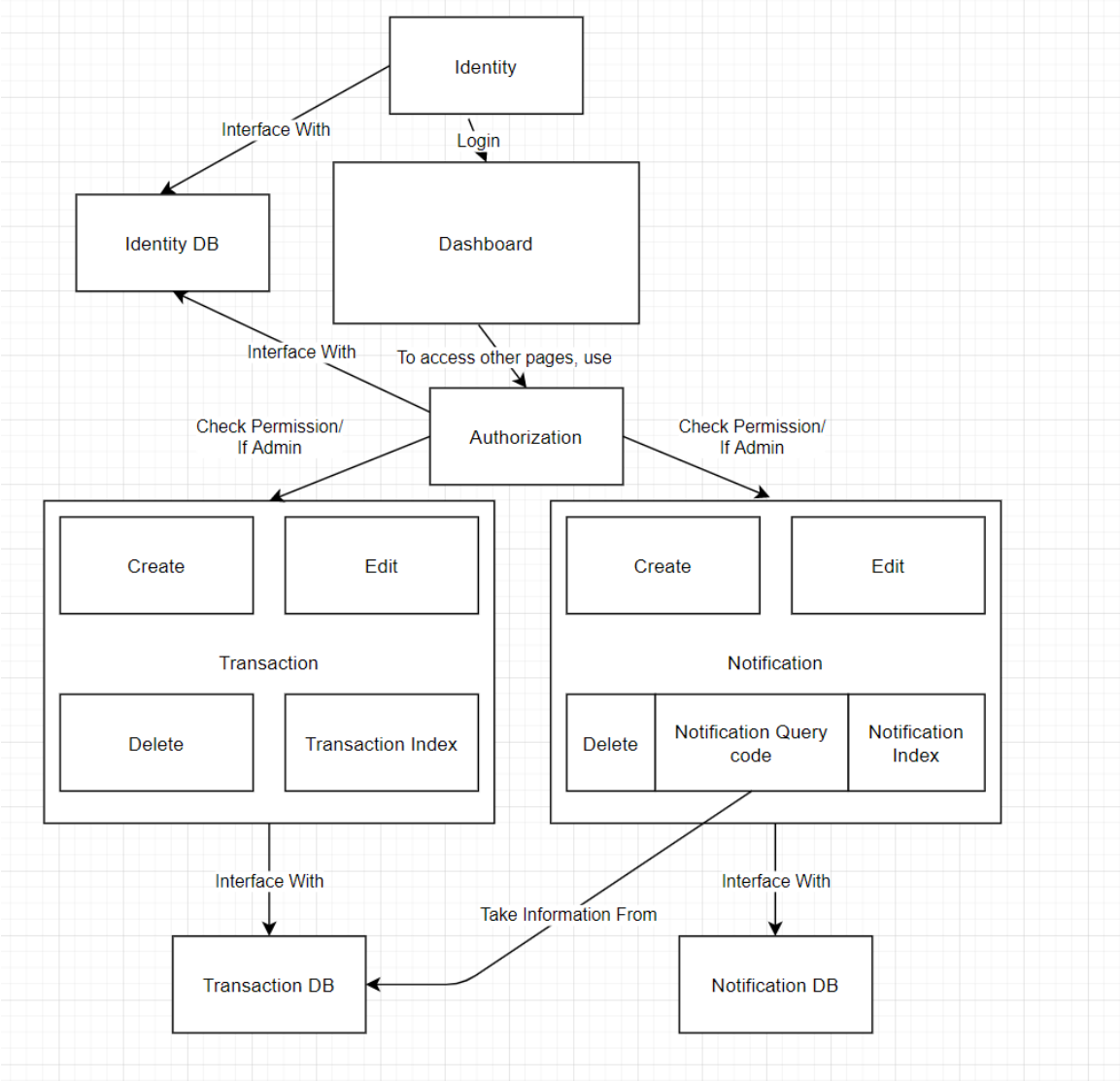
Figure 6.2



7 Development View

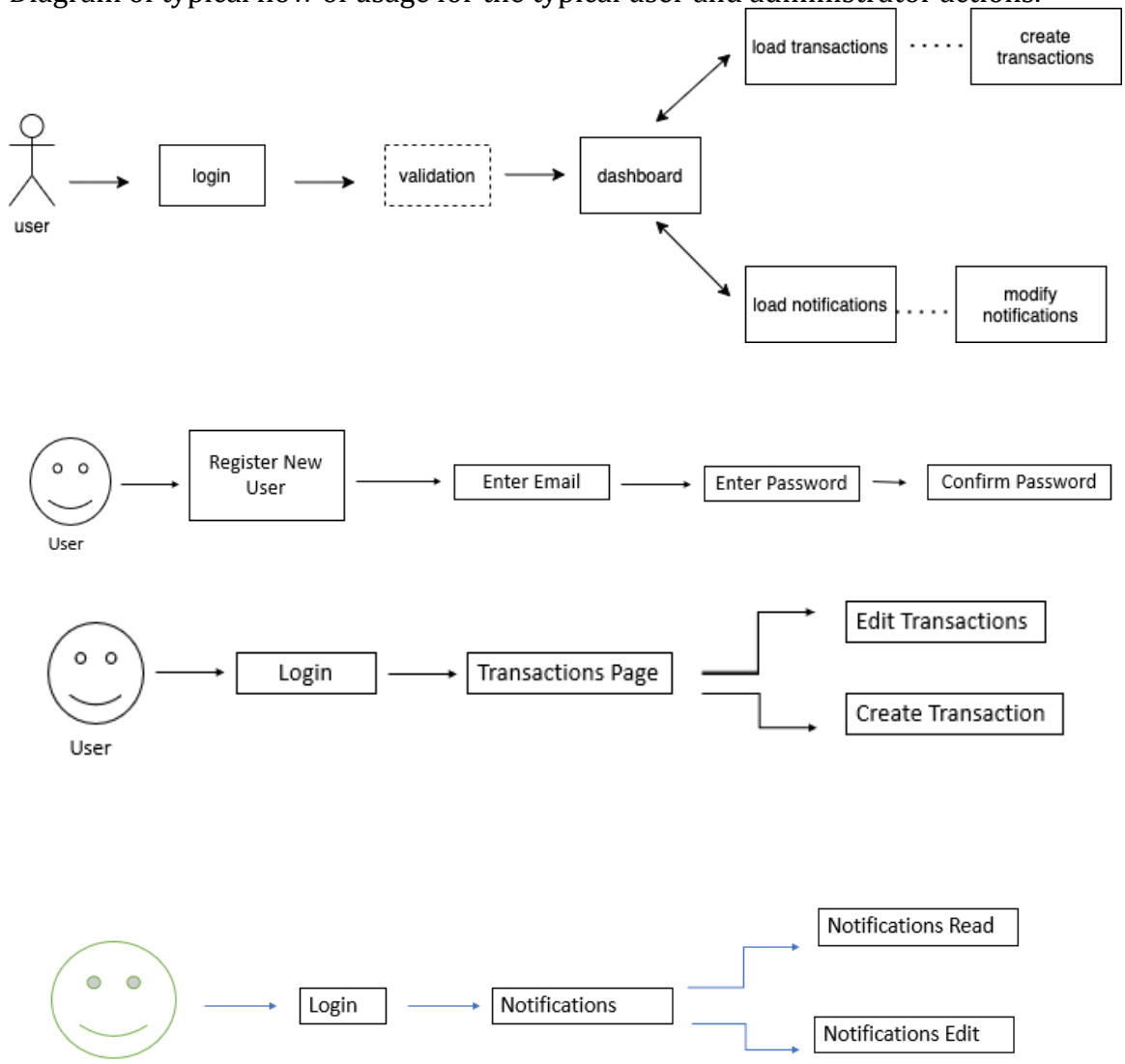
The software will adhere to the notification rules set by the user. Whenever the notification rules are triggered, the backend will access the database for the user data and relay it to the front end for the user to view. The UI component will be made simple such that any user with any level of expertise will be able to use the software easily. The application will know when to trigger a notification based on a certain given event by sending a specified query to the database. This query will tell the database to return any transactions in said database that match the criteria of this query. The application will count the number of returned queries and if that amount is one or more, it will add an entry into a separate table that includes all notifications triggered for that user, inputting information based on the information provided by the returned entries from the first query, including date/time or location of the transaction. The algorithm that does this should be based on either a SELECT SQL query or the equivalent function from the Entity Framework.

Figure 7.1


Identity
Interface With
Login
Identity DB
Dashboard
Interface With
To access other pages, use
Check Permission/
If Admin
Authorization
Check Permission/
If Admin
Create
Edit
Transaction
Delete
Transaction Index
Create
Edit
Notification
Delete
Notification Query code
Notification Index
Interface With
Take Information From
Interface With
Transaction DB
Notification DB

8 Use Case View

Diagram of typical flow of usage for the typical user and administrator actions.



Note: The base of this document was provided to the group in template form from Professor Bingham and modified to fit the needs of our team.