NoTiFi

Architecture/Design Document

**Table of Contents**

[**1**](#_heading=h.gjdgxs) **INTRODUCTION 3**

[**2**](#_heading=h.30j0zll) **DESIGN GOALS 3**

[**3**](#_heading=h.1fob9te) **SYSTEM BEHAVIOR 4**

[**4**](#_heading=h.3znysh7) **LOGICAL VIEW 5**

[**4.1**](#_heading=h.2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_heading=h.3dy6vkm) **Mid-Level Design 6**

[**4.3**](#_heading=h.1t3h5sf) **Detailed Class Design 8**

[**5**](#_heading=h.4d34og8) **PROCESS VIEW 11**

**6****PHYSICAL VIEW 11**

**7****USE CASE VIEW 12**

Change History

**Version:** 1.0

**Modifiers:** Bryan, David Hoang, Davith Lon, Sandy Yang, Trevor Poirier

**Date:** 04/04/2021

**Description of Change:** Initial Draft

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

# **Introduction**

This document describes the design for NotiFi, a web application developed for Commerce Bank. NotiFi is an application that allows Commerce Bank customers to have the ability to access their accounts and to make transactions remotely. Additionally, the application notifies users about their account activities via text or email. The purpose of this document is to describe the architecture and design of the Notifi web application in a way that addresses the interests and concerns of all major stakeholders. For this web application the major stakeholders are:

* Commerce Bank Members - They want assurances that the architecture will be able to provide system functionality advertised.
* Developers - They want an architecture that reduces complexity of developing the application and removes unnecessary overheads.
* Project Manager - They want assurance that the architecture is divided into categories that are not tightly coupled. This allows separation of expertises to different segments in the project without exhausting the resources.
* Commerce Bank - They want assurances that the architecture is able to encapsulate system functionalities required for the application and fulfill all the desired non-functional requirements listed in the Requirements Document.

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 NoTiFi web 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.

# **Design Goals**

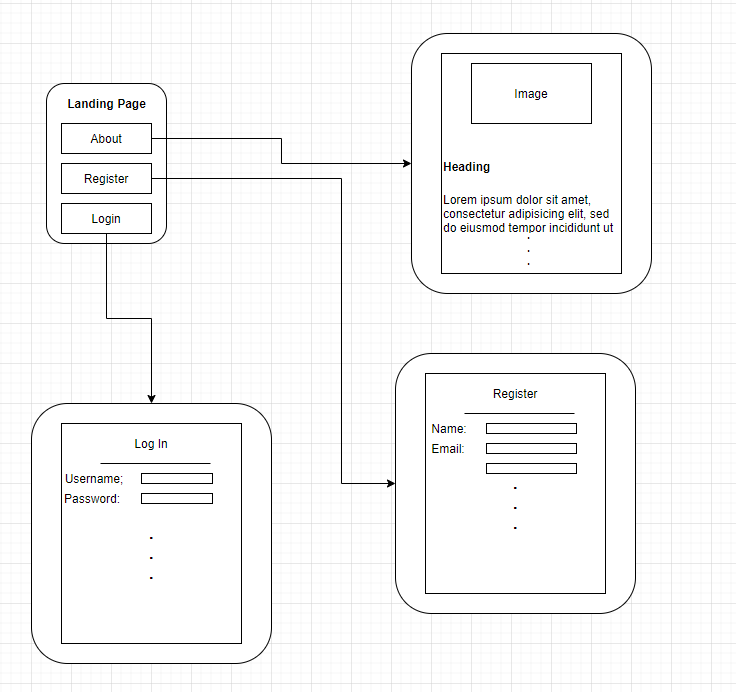
There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good 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 NoTiFi application are:

* The design should minimize complexity and development effort.
* The design should lead to minimal effort in terms of maintenance
* The design should be stable, having extremely minimal downtime between services
* The design should be scalable. We want the architecture to be able to handle future implementations of features and options.

# **System Behavior**

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review 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.



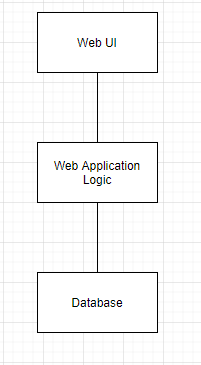
# **Logical View**

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.

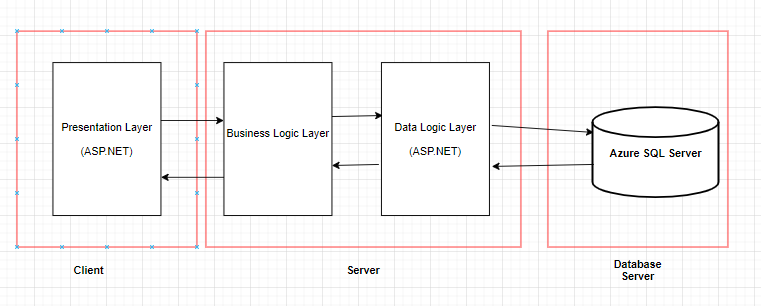
In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

## ***High-Level Design (Architecture)***

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



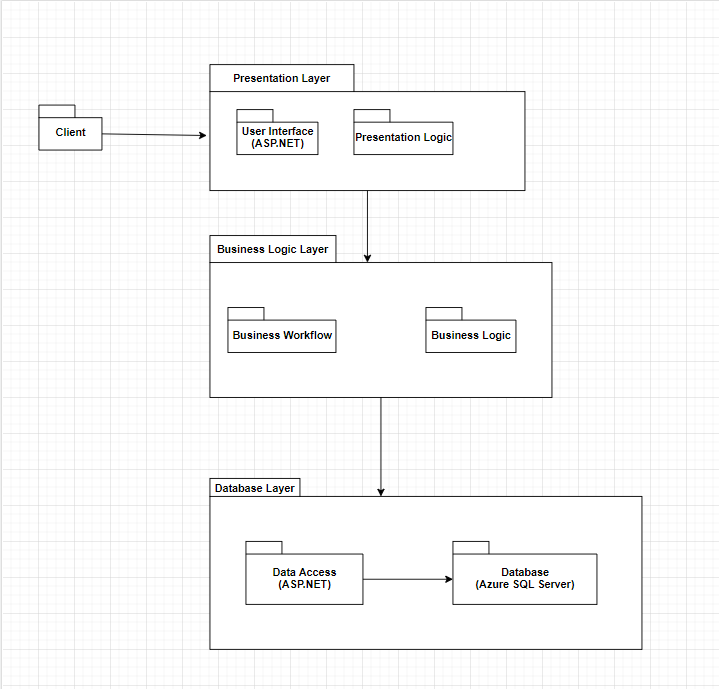
**Figure 1. High Level System Architecture**



**Figure 2. 3-Tier Model**

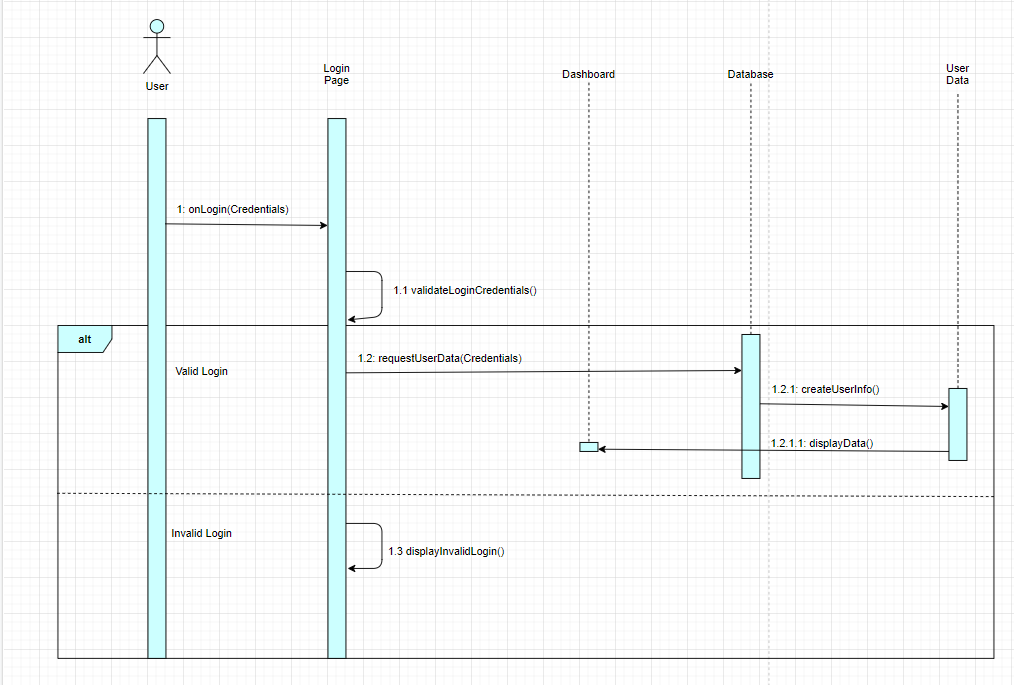
* The **Web UI** presents information to the user.
* The **Web Application Logic** handles all data and their logical applications, while also handling the requests and inputs from the uer.
* The **Database** is the central location for all of the information stored for the user.

## ***Mid-Level Design***

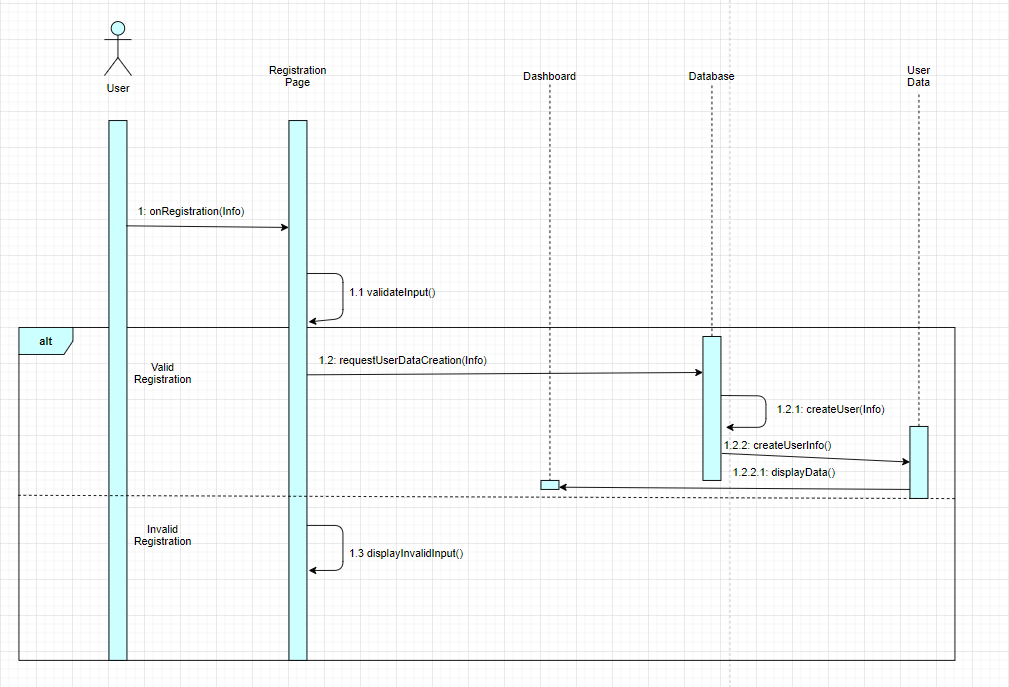


**Figure 3. Package Diagram**

* Client: Represents a client or user access our web application through the Presentation Layer
* Presentation Layer: The actual layer that clients or users interact with.
* User Interface: The GUI that is used to display information to the user or client. The User Interface is powered by ASP.NET
* Presentation Logic: The logic or instructions on how to build and style each web page that the client or user will see.
* Business Layer: The layer that handles implementing business rules and business logic.
* Business Workflow: The business workflows streamlines multi step processes. For example, logining in would need a business workflow as logging in is a multi step process (Receive Credentials, Verify credentials, Return data if valid and return error otherwise).
* Business Logic: The business logic implements rules and regulations such as CRUD (Create, Read, Update, Delete)
* Database Layer: The layer that handles accessing data from the database.
* Data Access: The way in which data is requested and queried from the database. The data accessing is powered by ASP.NET.
* Database: The location in which data is actually stored in. The database being used is an Azure SQL Server.

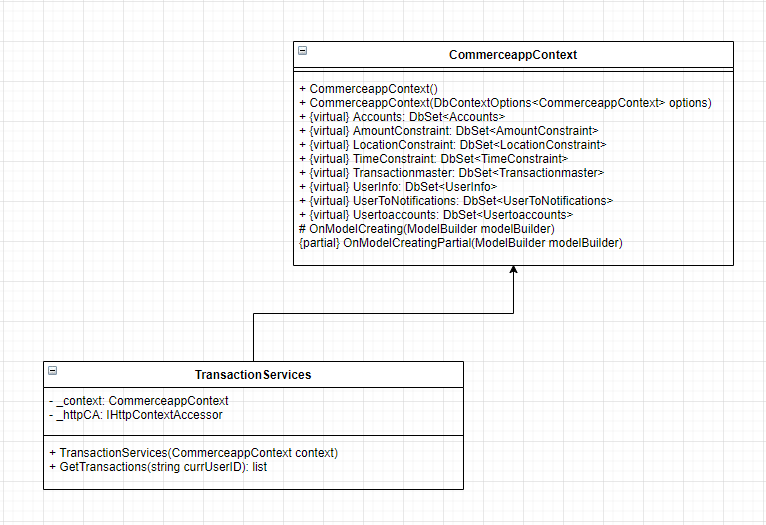


**Figure 4. Sequence Diagram for Login**

****

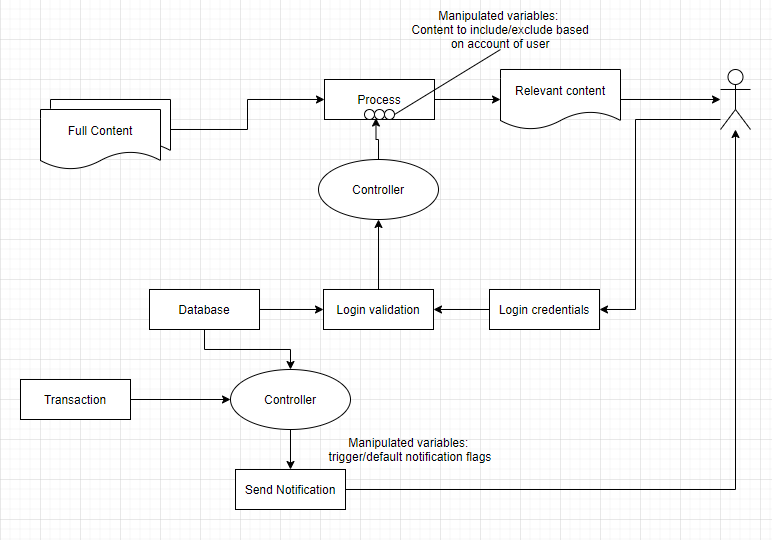
**Figure 5. Sequence Diagram for Registration**

## ***Detailed Class Design***



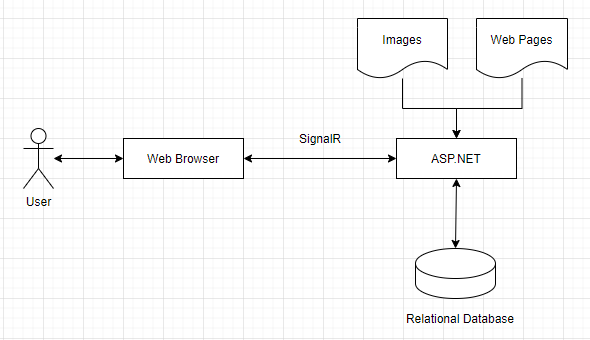
**Figure 6. Class Diagram**

# **Process View**



**Figure 7. Process Control Diagram**

# **Physical View**



**Figure 8. Physical View of Web Application**

# **Use Case View**

**Send Notification:**

Actors: Registered User

Basic path:

1. A transaction is made.
2. System checks whether a transaction applies with the notification rules of the account.
3. Transaction applies to all the notification rules set.
4. System sends an email or a text connected to the account about the notification.
5. System is idle until step 1 is initiated again.

Alternate path:

3b. Transaction does not apply to all of the notification rules set.

1. System does not send an email or a text.
2. System is idle until step 1 from the basic path is initiated again.

**Login:**

Actors: Registered User

Main Path:

* User already has an account with Commerce.
* User clicks on the login button on the splash screen.
* Login screen is shown. User enters username and password into respective fields and presses “login” or Enter key.
* Successful check against database, login success, user is brought to their dashboard.
* Their login timestamp is logged with appropriate keywords.
* Alternative Flow: Invalid ID or password.
  + Message shown on page that indicates invalid username or password. DO NOT SPECIFY WHICH IT IS. Red boxes around both fields.
  + At most three tries until the account is locked for an hour. In other words, they have the initial try and then two more.
  + Attempts are timestamp logged and so is account lockout with appropriate keywords.

**View Transaction:**

Actors: Registered User

Basic Path:

1. A charge hits the user’s account.
2. Pending and available balances are updated in response.
3. The charge is then added to the top of the Transaction Summary (what the charge was and for how much), along with the resulting balance when the charge hits.
4. User sees their balance is different from the last time they checked the application.
5. Users are able to clarify what monthly charge hit their account and for how much.

Alternate Path:

3b. The Transaction Summary is not updated with the aforementioned charge.

4. User sees their balance is different from the last time they checked the application.

5. User is unable to clarify what the charge was or for how much, and must call

or visit the bank to clarify.

**Display Dashboard:**

Actors: Registered User

Basic Path:

1. User successfully logs into the account.
2. The dashboard will load.
3. The dashboard will display the total number of notifications for each notification rule that was set up.

Alternate Path:

1. The dashboard will display the message “No new notifications”.