[Commerce Bank App]
Architecture/Design Document

Table of Contents

Change History

Version: <1.0>
Modifier: <Tayler Singleton>
Date: 04/05/2020
Description of Change: <Diagrams are added>

Version: <x.y>
Modifier: <Name>
Date: mm/dd/yyyy
Description of Change: <What was modified/added?>

1. Introduction

Architecture and Design

The purpose of the architecture/design document is to explain the organization of the code. A well-written architecture document will make it easier for new programmers to become familiar with the code.

The architecture/design document should identify major system components and describe their static attributes and dynamic patterns of interaction.

Software architecture and designs are typically expressed with a mix of UML models (class and sequence diagrams being the two most common) and prose. Dataflow diagrams are also helpful for understanding the interaction between components and overall flow of data through the system.

About this Template

This template suggests one way of documenting a software system's architecture/ design. You aren't required to include every section in this template nor all the content in the sections you do include. However, the document you do submit should pass the following checklist:

- Are design objectives clearly stated? For example, if performance is more important than reusability, this should be made clear at the start of the design specification.
- Does the architecture partition the implementation into clearly defined subsystems or modules with well-defined interfaces?
- Does the architecture express in a clear way the main patterns of communication between subsystems and modules?
- Does the architecture satisfy the requirements?
- Is the architecture traceable to requirements?
- Any models created should either be expressed with a well-known modeling language, or if a well-known modeling language isn't used, the syntax and semantics of the symbols that are used should be defined.

This document describes the architecture and design for the Commerce Bank App application being developed for Commerce Bank customers. Helps users display their transactions and get notifications of certain transactions based on the notification rules the user has set.

The purpose of this document is to describe the architecture and design of the Commerce Bank 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. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all business logic.
- 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 App 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

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 vice 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 App application are:

- The design should minimize complexity and development effort.
- The design should use the Commerce Bank color scheme in styling.
- The design should be loosely coupled and highly cohesive.

3. 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.

The user will login to the system and be redirected to the homepage.The homepage will give the user the ability to export transactions to a spreadsheet and to pull/compare notifications within different timeframes. Within the homepage, there is the dashboard, which will display the number of times a notification rule has been triggered and the user will also have the ability to hide notification rules. The transactions summary will give the user the ability to add transactions,which will automatically trigger any associated notification rule and the user can also sort their transactions. Triggers are used to add/delete/edit notification rules.

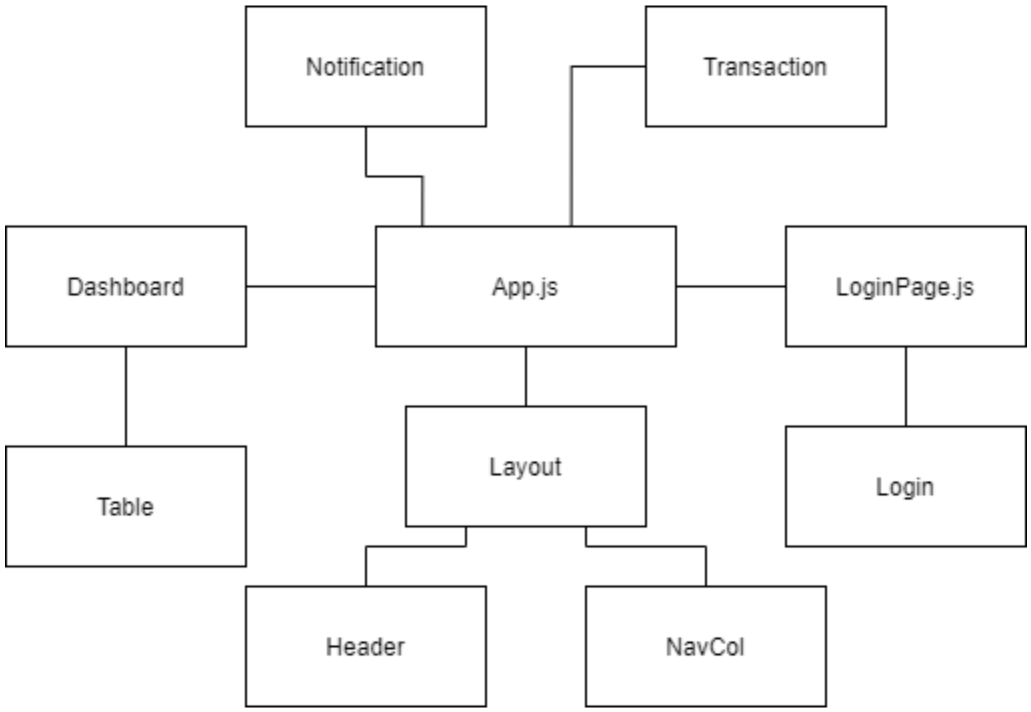
4. Logical View

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.

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)*

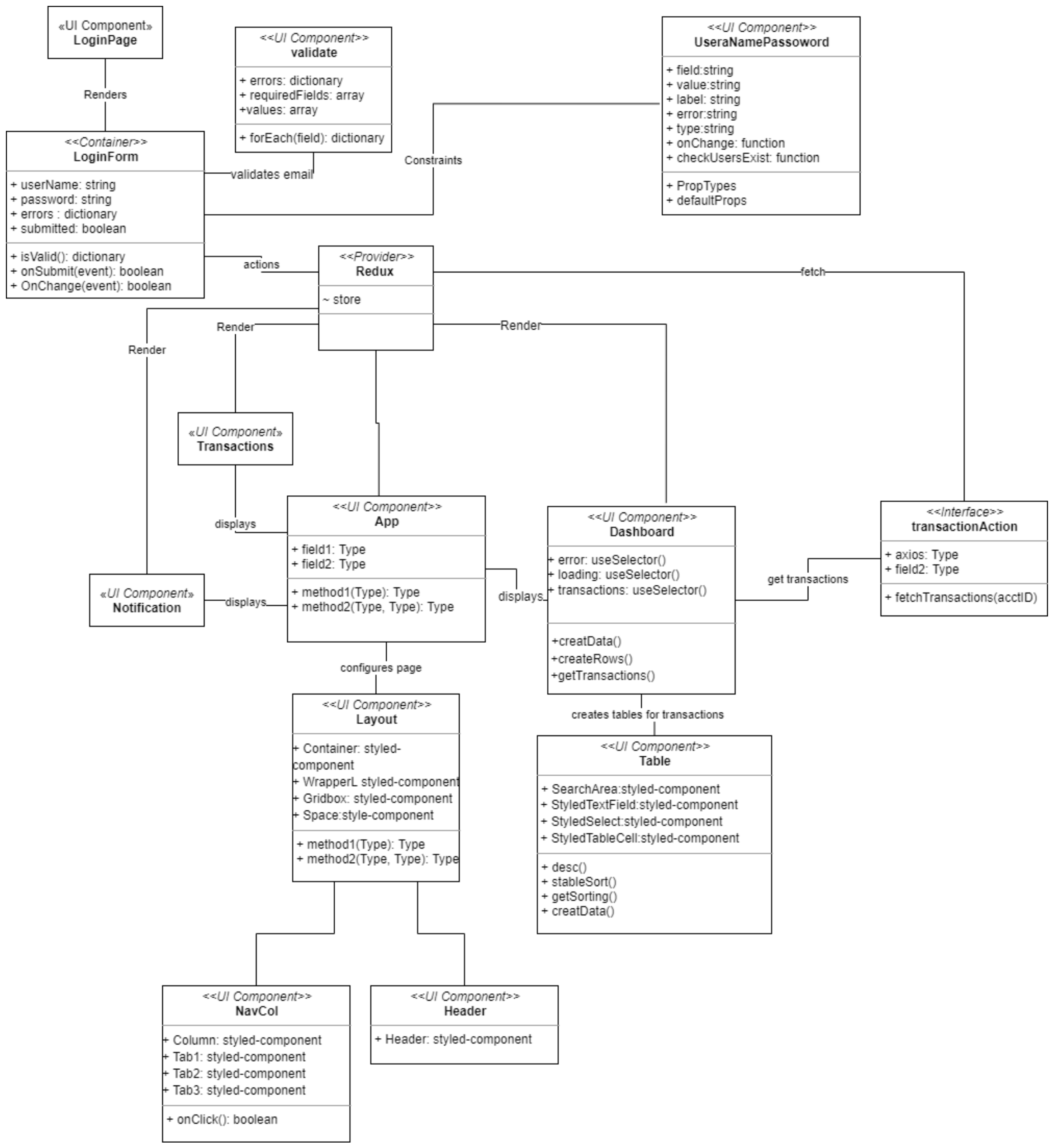
The high-level view or architecture consists of 9 major components:

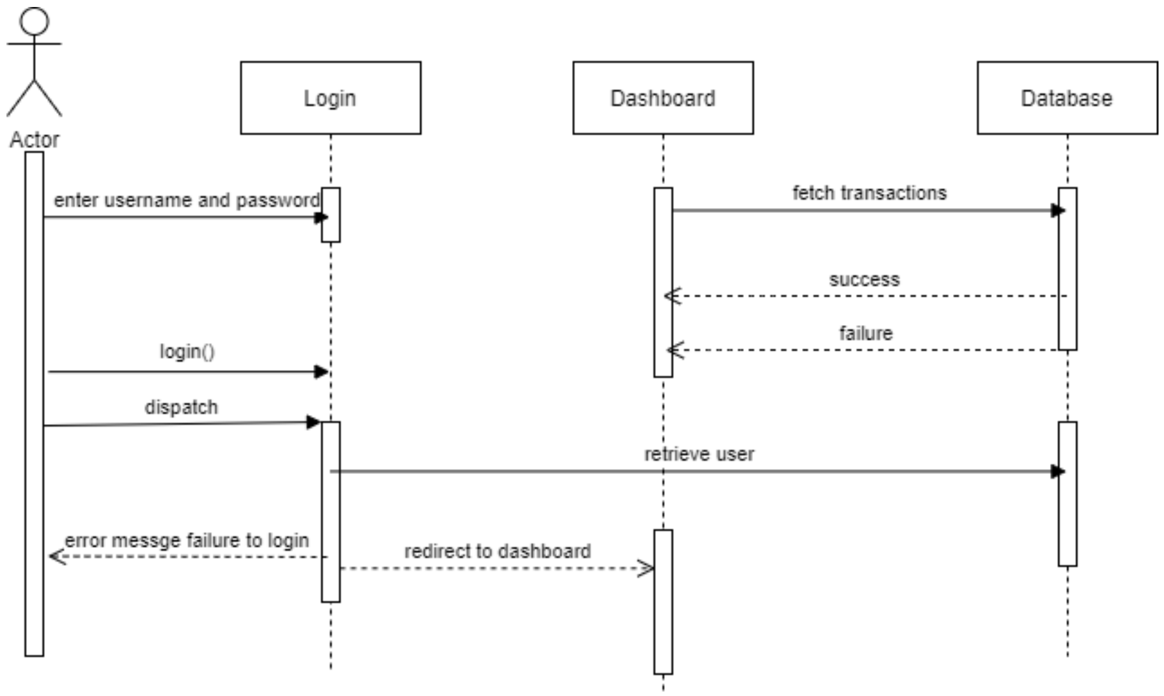


Architecture

- The **Login** component allows the user to login into the application and will redirect the user to the homepage/dashboard if the user inputs the correct username and password.
- The **Notification** component is used to keep track of the notification rules the user creates, deletes, or edits.
- The **Dashboard** component will have the functionality the homepage and the dashboard within the homepage will have such as displaying the number of times of notification rule has been triggered, the ability to hide notification rules, the ability to export all transactions in an spreadsheet and pull/compare notifications
- The **Transaction** component is to keep track of all transactions the user has and gives the user the ability to add transactions, which thereby will affect the triggers component and notification component.
- The **Triggers** component will be used to add the functionality of adding/deleting/ editing notifications
- The **App.js** is considered the root component for the app that renders the dashboard, transaction, notification.
- The **LoginPage.js** component is the root component that renders the Login component.
- The **Table** is used to create all tables used within the application
- The **Header** is used to navigate components that display information and actions relating to the current screen.
- The **NavCol** is used to navigate the columns in the application.
- The **Layout** is used to arrange the application using a grid system.

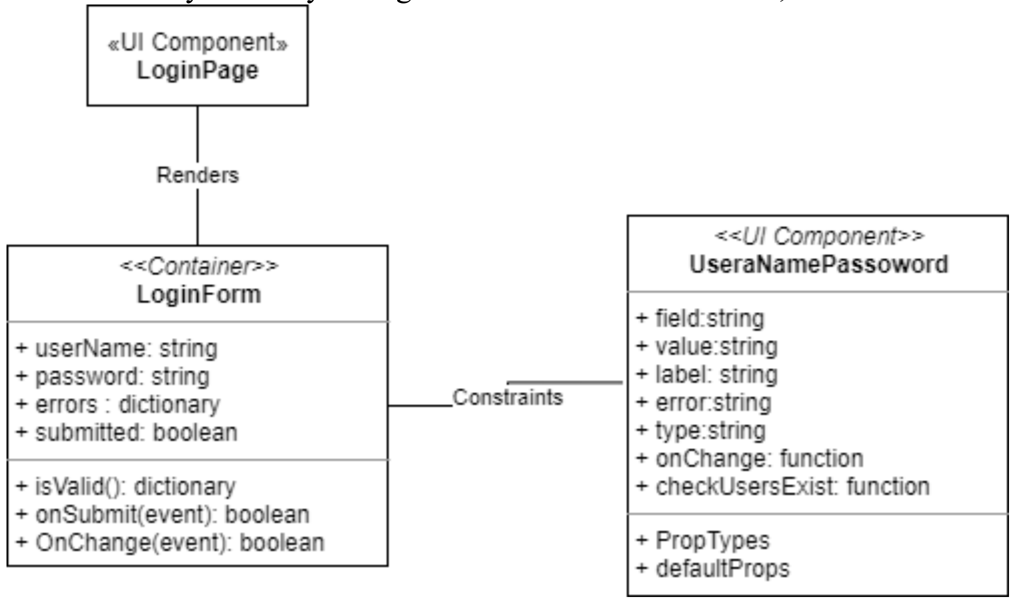
○ ***Mid-Level Design***



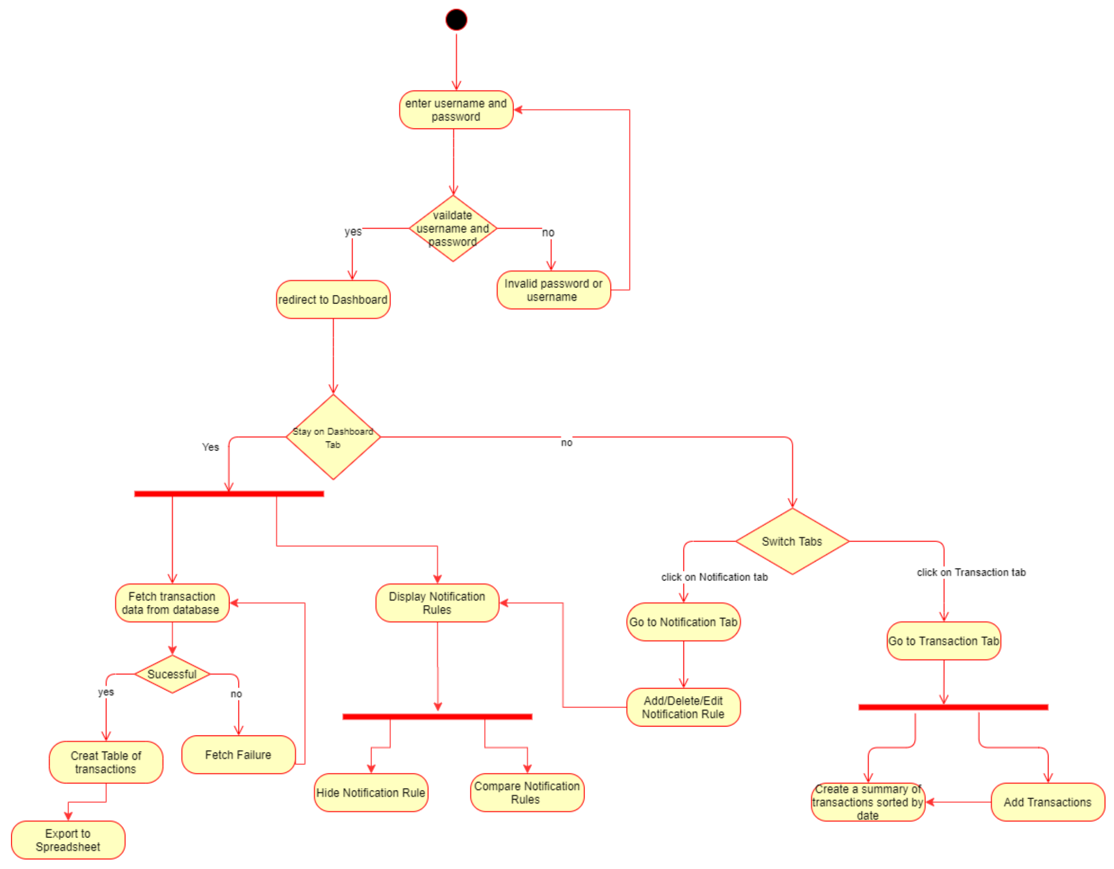


○ ***Detailed Class Design***

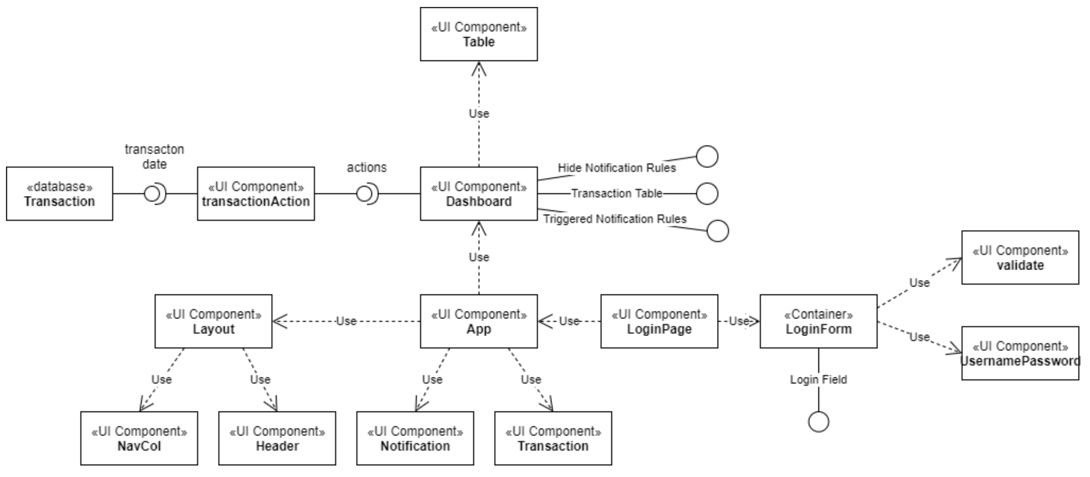
<For a few key classes you might want to show associations, attributes and methods.>



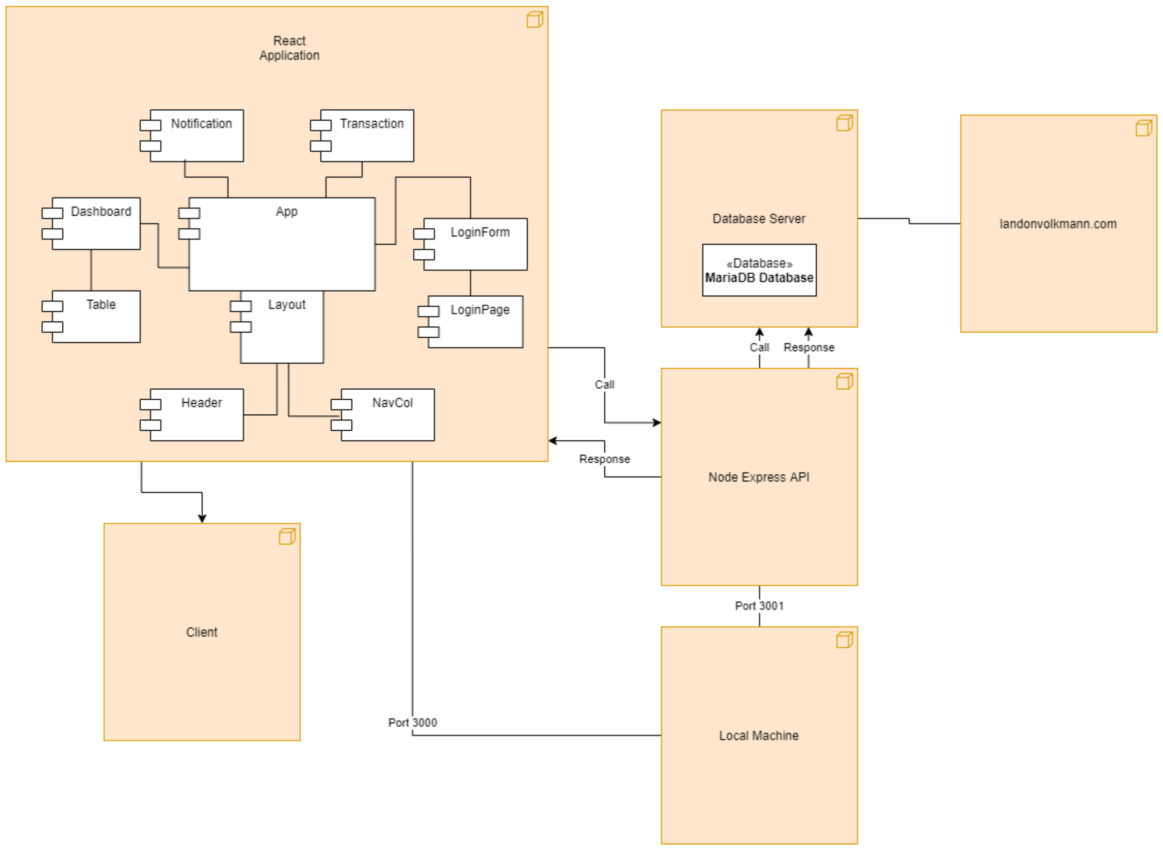
5. Process View


enter username and password
vaildate username and password
yes
no
redirect to Dashboard
Invalid password or username
Stay on Dashboard Tab
Yes
no
Switch Tabs
click on Notification tab
click on Transaction tab
Fetch transaction data from database
Display Notification Rules
Go to Notification Tab
Go to Transaction Tab
Sucessful
yes
no
Creat Table of transactions
Fetch Failure
Add/Delete/Edit Notification Rule
Hide Notification Rule
Compare Notification Rules
Create a summary of transactions sorted by date
Add Transactions
Export to Spreadsheet

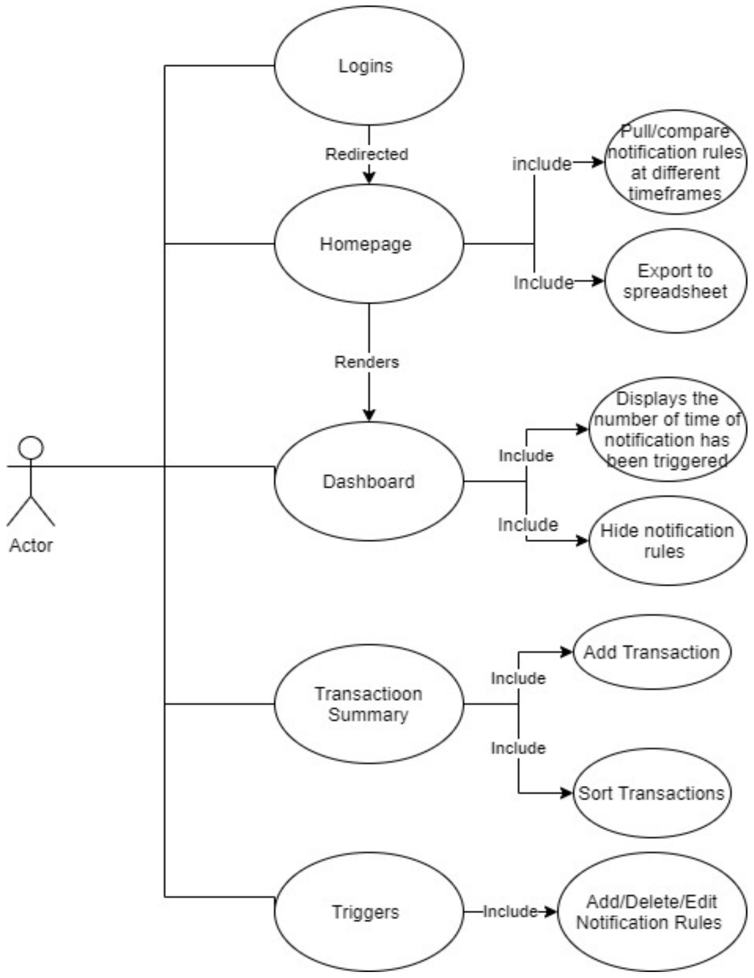
6. Development View



7. Physical View



8. Use Case View


Logins
Redirected
Homepage
include
Pull/compare notification rules at different timeframes
Include
Export to spreadsheet
Renders
Actor
Dashboard
Include
Displays the number of time of notification has been triggered
Include
Hide notification rules
Transactioon Summary
Include
Add Transaction
Include
Sort Transactions
Triggers
Include
Add/Delete/Edit Notification Rules