Trigger Tracker Commerce Project
Architecture/Design Document

Table of Contents

1 INTRODUCTION 3

2 DESIGN GOALS 4

3 SYSTEM BEHAVIOR 4

4 LOGICAL VIEW 4

4.1 High-Level Design (Architecture) 5

4.2 Mid-Level Design 5

4.3 Detailed Class Design 6

5 PHYSICAL VIEW 7

6 USE CASE VIEW 7

Change History

Version: 1.0
Modifier: Alexa Summers
Date: 03/30/2020
Description of Change: Initial document creation

---

1 Introduction

This document describes the architecture and design for the Trigger Tracker application being developed for Commerce Bank. This web application will allow Commerce Bank customers to look at their accounts and view past transactions, as well as set notification triggers. A database will keep track of user information as well as transaction information, and the user will be able to pull information from the database when requested. Triggers will also be stored in a database, and will include options pertaining to timing, location, and transaction cost.

The purpose of this document is to describe the architecture and design of the Trigger Tracker 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 application is user friendly and will complete all the functional as well as non-functional requirements.
- Developers – they want an architecture that will minimize complexity and development effort in order to prevent wasting costs or resources.
- 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 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. UI logic is grouped into one module, database management is grouped into another, and connecting frontend and backend is grouped into another module.

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 Trigger Tracker 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 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 Trigger Tracker application are:

- The design should minimize complexity and development effort.
- The design should target an audience with average-to-little knowledge of computers.
- The design should allow for tasks to be split up into different groups and assigned to people with expertise in the area, without hindering other people's ability to work on their assignments.

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 expect 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.

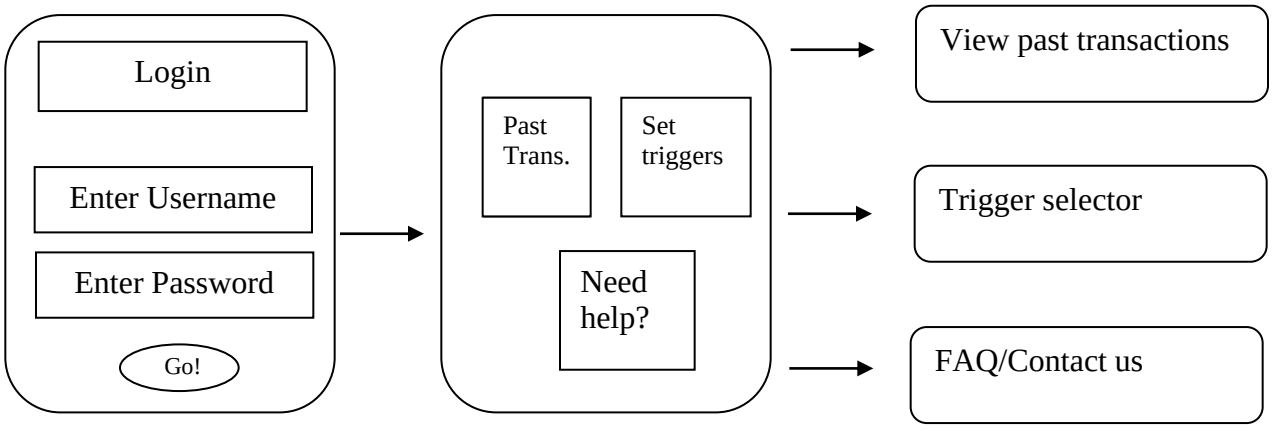

Fig. 1 System Behavior

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.

4.1 High-Level Design (Architecture)

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

Fig. 2 System Architecture

- The **Database** will be running in a Maria DB instance using Microsoft Azure.
- The **Spring Boot App** will access the database, host the REST endpoints, generate Excel reports, and manage the notifications. This will also act as the front end host.
- The **Angular App** will be the UI for the login page and the dashboard, as well as setting triggers and viewing past transaction reports.

4.2 Mid-Level Design

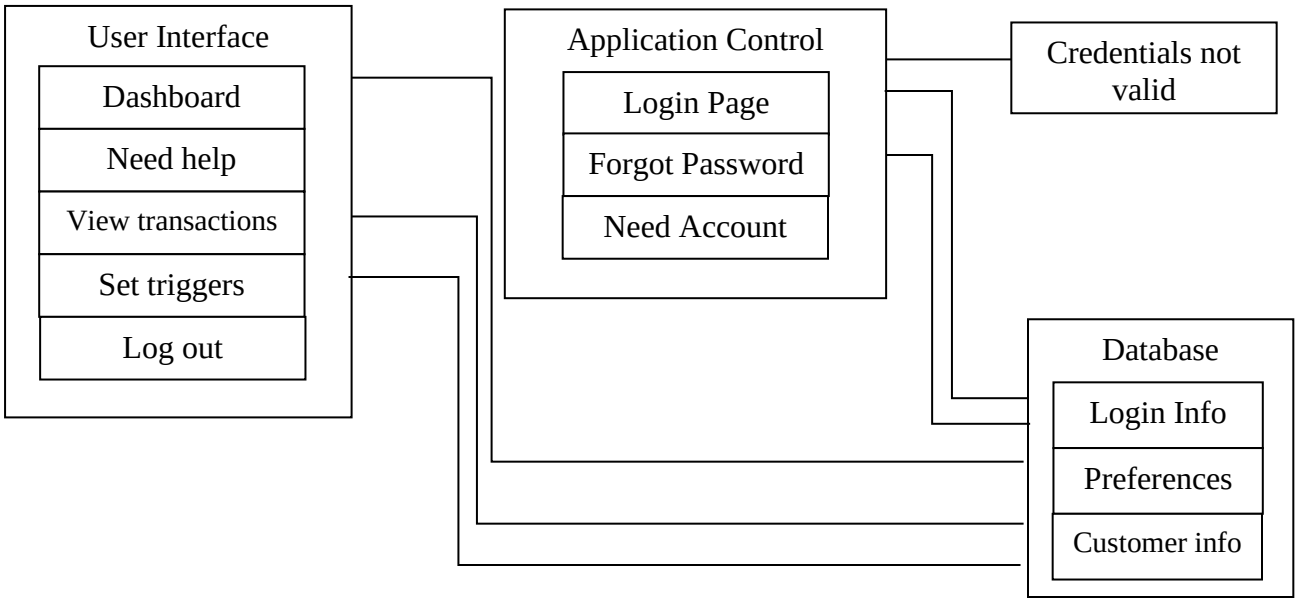

Fig. 3 Mid-Level Design

4.3 Detailed Class Design

Because the bulk of our project is database driven, we included our database ER diagram for our Detailed Class Design.

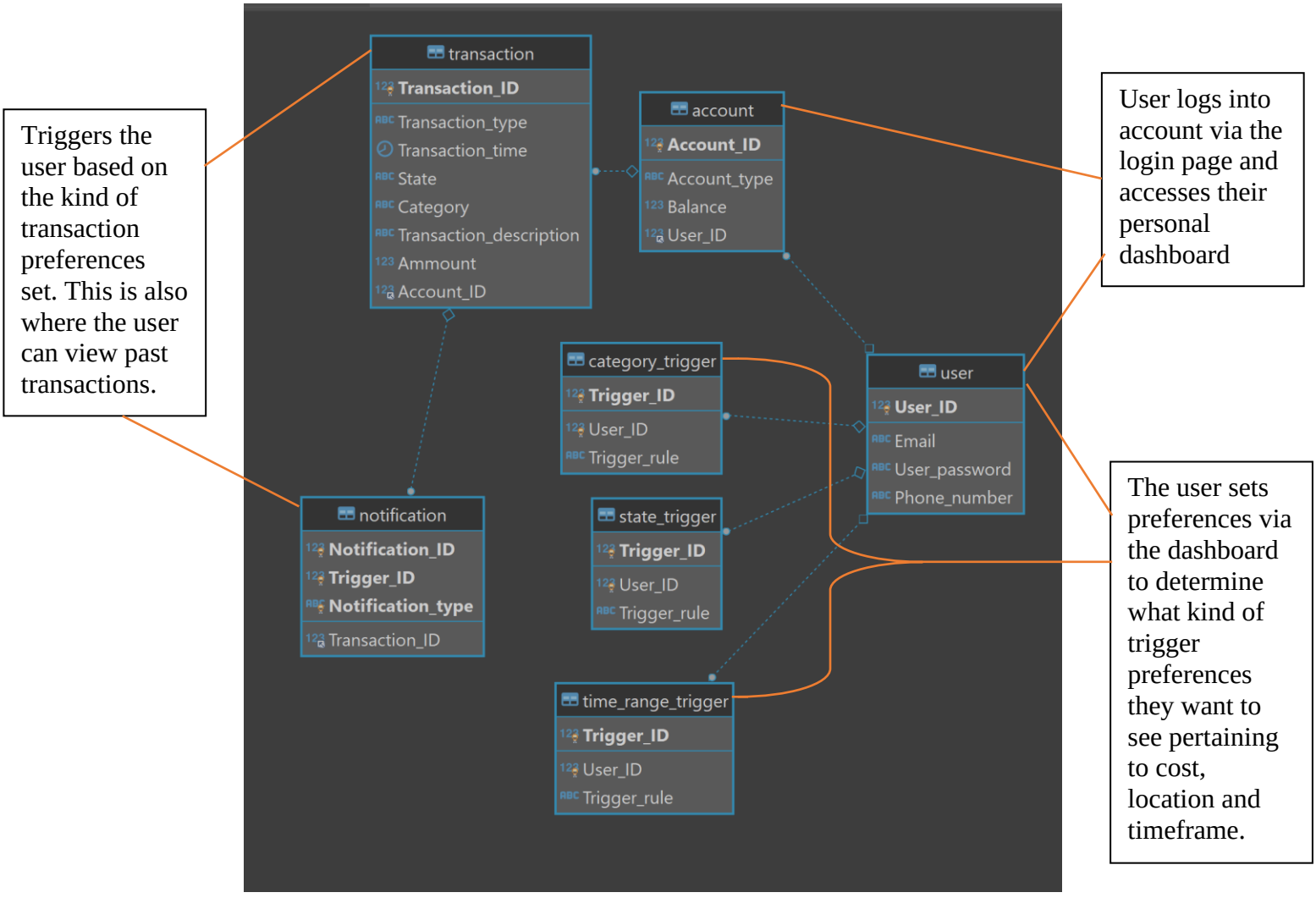

Fig. 4 Detailed Class Design

5 Physical View

Because our design is relatively simple, our physical view looks really similar to our high-level design. This is how we know we are staying on track with our cost, resource, and usability goals.

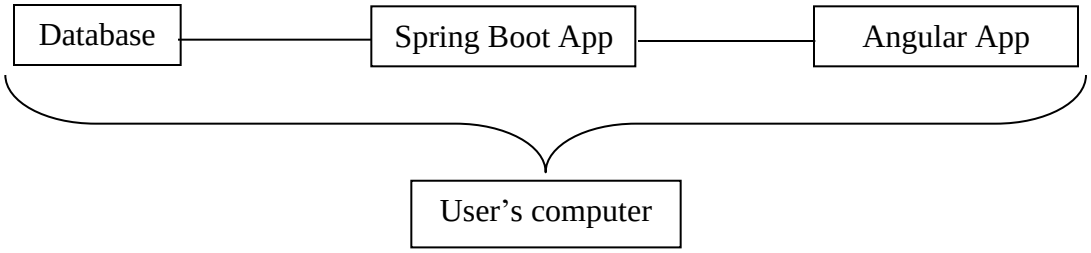

Fig. 5 Physical Design

6 Use Case View

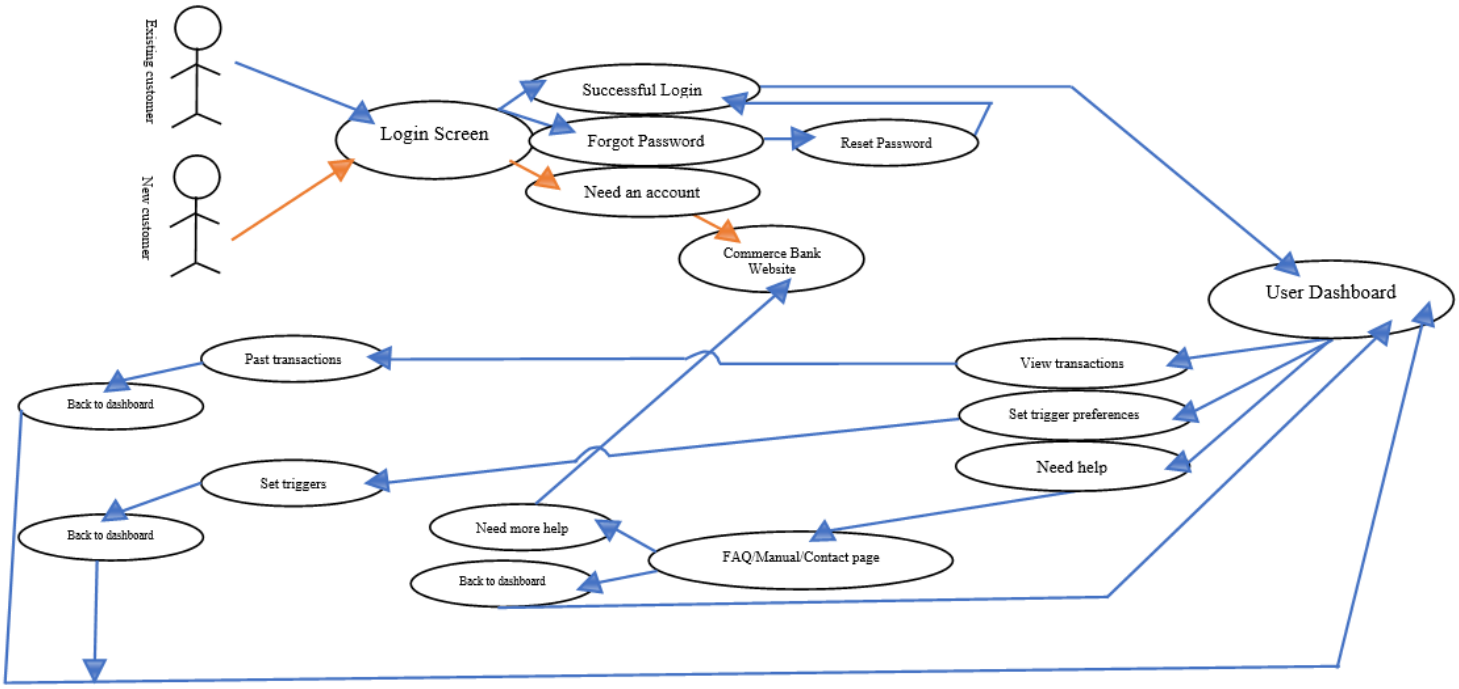

Fig. 6 Use Case Design