**System Guide**

**Commerce Notifications Application**

**Group 2**

April 26, 2020

**Team Members**

Kory Overbay

Matthew Hopkins

Jimin Choi

Connor Marchand

# Contents

Contents 2

1 Introduction 4

2 Setup MySQL Server 4

3 Setup Spring Boot Application 4

4 Setup Angular Application 5

5 System Maintenance 5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1 | 4/26/2020 | Kory Overbay,  Matthew Hopkins,  Jimin Choi,  Connor Marchand | Initial Document |

# Introduction

The Commerce Notifications Application allows users to set up custom notification rules for new transactions coming in through their Commerce Bank accounts from a web browser. This document details instructions for accessing the code base and running the application locally.

Link to Application Repository: <https://github.com/umkc-cs-451-2020-spring/Commerce-Notifications-Web-App>

Software Required:

Node.js

Angular 9

Java JDK1.8

MySQL 8.0

# Setup MySQL Server

During development, a MySQL server was created on a Ubuntu Linux PC, but any device hosting the MySQL server should work for local development

1. Install MySQL 8.0 on your computer
2. Create a database ‘CommerceDB’
3. Set up a user for the Spring Boot application
4. Run the DDL0 (And if you want test data, run DDL1) files from the project

# Setup Spring Boot Application

For development, the team used Spring Tool Suite 4 for running the application, but anything capable of running a Spring Boot server should work.

1. Install Java JDK1.8 and Spring Tool Suite 4
2. Import the Commerce-Server project as a Maven project in STS.
3. In the application.properties file, set the spring.datasource.url to the MySQL database along with the application’s database username and password
   1. spring.datasource.url=jdbc:mysql://<Server IP here>/CommerceDB?useSSL=false&useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
4. Build and run the application

# Setup Angular Application

For development, the team used Visual Studio Code to make typescript, html, and css changes to the application. To run the front end:

1. Install Node.js and run ‘npm install -g @angular/cli’ to install Angular on your system
2. Navigate to the Commerce-Client folder in a terminal and run ‘npm install’
3. To run the application, run ‘npm start’ in a terminal in the Commerce-Client folder
4. Navigate to localhost:4200 to access the web application

# System Maintenance

## Angular Front End Code

The user interface consists of 3 main pages with a navigation bar at the top of each. Each page is implemented within a folder of it’s name in Commerce-Client/src/app. Each folder follows the standard angular component format of a CSS file, an HTML file, and a TypeScript file. The HTML and CSS files implement the visual look of the pages while the TypeScript files implement the logic behind them.

The Notifications and Transactions pages each have popup called from them that are implemented in the rule and new-transaciton folders respectively. These popups follow similar logic as the main pages.

All three main pages reference a specialized service.ts file that create and handle the HTTP calls to the Spring Boot server.

On the first run, the application displays the login screen. Once the user has logged in, the view is switched to the Notifications Dashboard. The navigation bar at the top of the pages enable the user to navigate between the pages mentioned above. The user is able to switch to other pages before logging in, but no data is available to be shown until they are logged in.

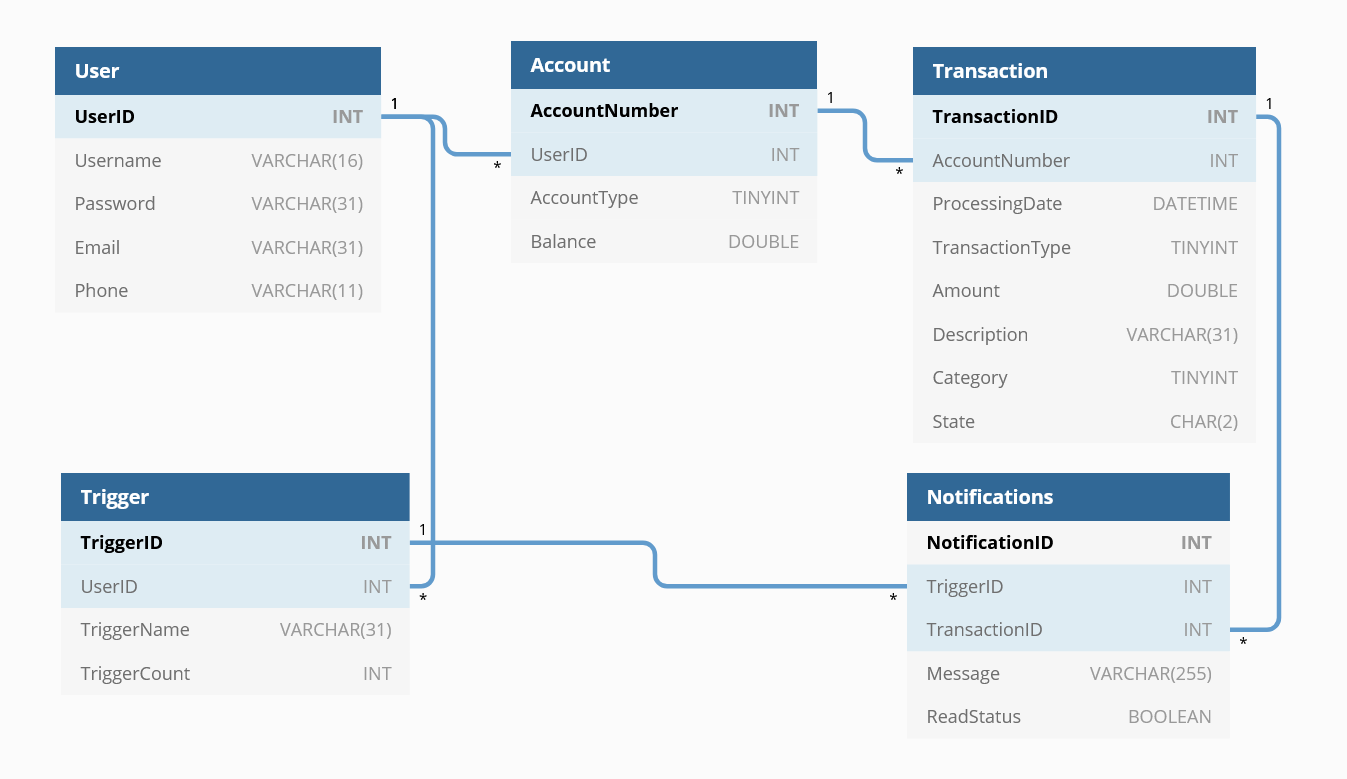
## Spring Boot Back End Java Code

The Spring Boot application is implemented in Java. The application acts as an API to the front end for retrieving data from the database.

The flow of the application begins in the api package within the controller files of packages labeled for their respective pages. These controllers use Spring HTTP request annotations to listen for requests coming from the front end. Each endpoint has a function path through a respective DAO -> DAOImpl file. DAOImpl files contain the Jdbc functions for querying and updating data on the database.

The most complex function in the back end belong to the addTrigger function in the NotificationsDAOImpl file. This function takes in user given data and string builds a unique MySQL Script for adding a trigger to the transactions table. These triggers are responsible for the notifications feature of the application. Spring JdbcTemplate and NamedParameterJdbcTemplate provide SQL Injection prevention for parameters, however the Trigger names coming from the front end need to be sanitized before being executed in the add rule function, as they cannot be added to scripts as a parameters. Other functions that call MySQL scripts use scripts defined in the sql package of the application.

## MySQL Database

The MySQL Database is best represented by this ER Diagram:

Underlying MySQL Triggers are created by users on the Transaction table, which are kept track of through the Trigger table.