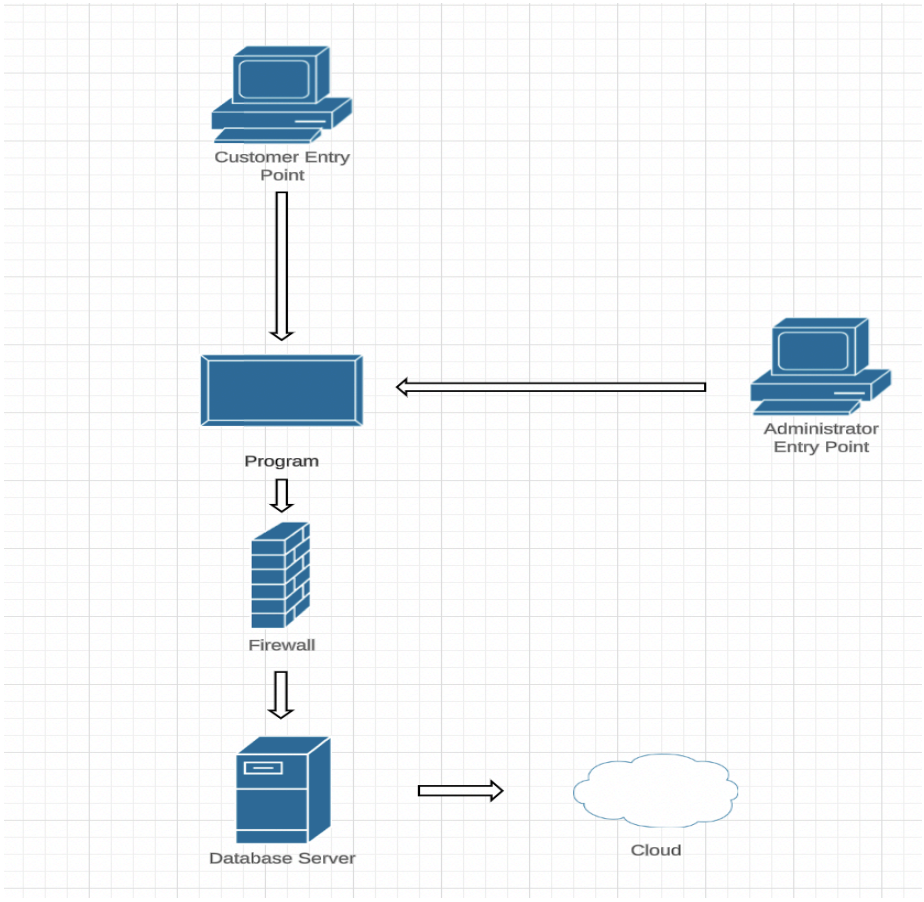
Software Design 2.0 for Clothing Store POS System

By: Shreyas Basu and Salvador De La Torre

1 Introduction

This document outlines the software architecture diagram as well as a strategy to efficiently and properly manage the data needed to use this diagram for a storewide kiosk system that will initially be launched in one store and eventually spread store-wide. The purpose of this system is to make it easier for customers, employees, and the owners to shop/work in the store. It's needed because the current system is inefficient, not helpful/intuitive, and an upgrade is long overdue. This system-wide change will affect employees in that it will make their job easier. It will also affect customers by making the shopping experience next-level, more fun, and unique. Overall, it will be in the best interest for the company to get this system. This document is outlined into four sections, the introduction, software architecture diagram, data management strategy, and tradeoff discussion.

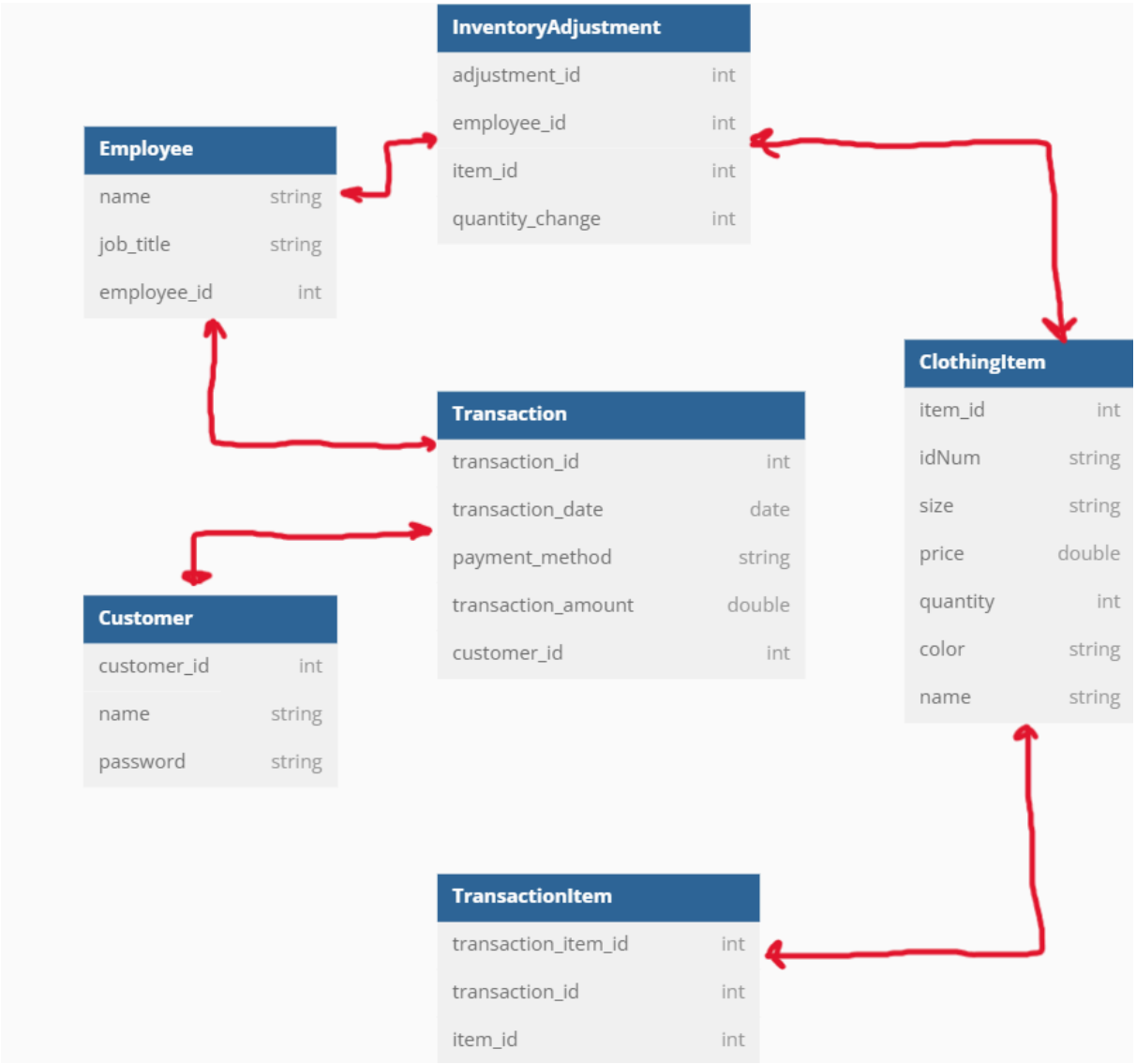
2 Architectural Diagram:



Description

This Software Architectural Diagram is precise and concise. There are two entry points, the user and administrator entries. Once correct credentials are entered, it goes to the program where they do what they need to do. For customers, it would be handing the items to the employee for them to be scanned and checked out. For administrators, it would be actions such as price matching, returns, account setup, seeing purchase history, etc. These programs store data in a database server which is then backed up by the cloud

3 Data Management Strategy (SQL Diagram):



Description:

The SQL diagram is used, to show how the data needed for our system will be organized and how it all links together.

The **Employee** part of the diagram, is meant to store all the needed information and data regarding employees in the system. This will hold the employee's name, job title, as well as their employee ID which will be unique to each employee and allow them to login and access the system properly.

The **Customer** part of the diagram, holds any and all information and data regarding the customers that are in the system. The information needed to be stored by the customer include their name, password to access their account in the system, and their unique customer ID to identify each individual customer that is a part of the system.

The **ClothingItem** part of the diagram is meant to store the data regarding the clothing items that are input into the system. In this table, there will be certain pieces of information and attributes to the clothing items that will make it easier to identify and organize the product that is available in the system.

The **Transaction** part of the diagram is meant to store information regarding each transaction that is made using this system. The information needed includes the transaction ID, date, payment method, amount, and customer information. This will allow for quick and easy access when needing to lookup or find past transactions for any particular reason that may be needed.

The **TransactionItem** part is similar to the Transaction part of the diagram, however it holds what items were being sold/returned during that transaction to make sure proper information and records are kept in the system.

Lastly the **InventoryAdjustment** part is meant to hold the data regarding any adjustments that are made to the inventory. This includes, the ID number for a particular adjustment, the ID of the employee who made that adjustment, the ID for the item that was adjusted in the inventory, and lastly the quantity change that was made for that particular item.

4 Tradeoff Discussion:

Since SQL is better for organized data types and our data types can be very easily organized, SQL is the best choice. A non-SQL data structure would be more suited for a

system with unstructured data types as there is a less strict organized structure. In SQL, there is a strong focus on data relationships which makes it fit with this system well. It's also really easy to search using SQL's query searching. The structure makes it easy to vertically scale meaning it's easier to add resources to the server. A non-SQL data structure would not allow for any of the above benefits. Overall, these reasons made it clear that SQL was the structure to use.