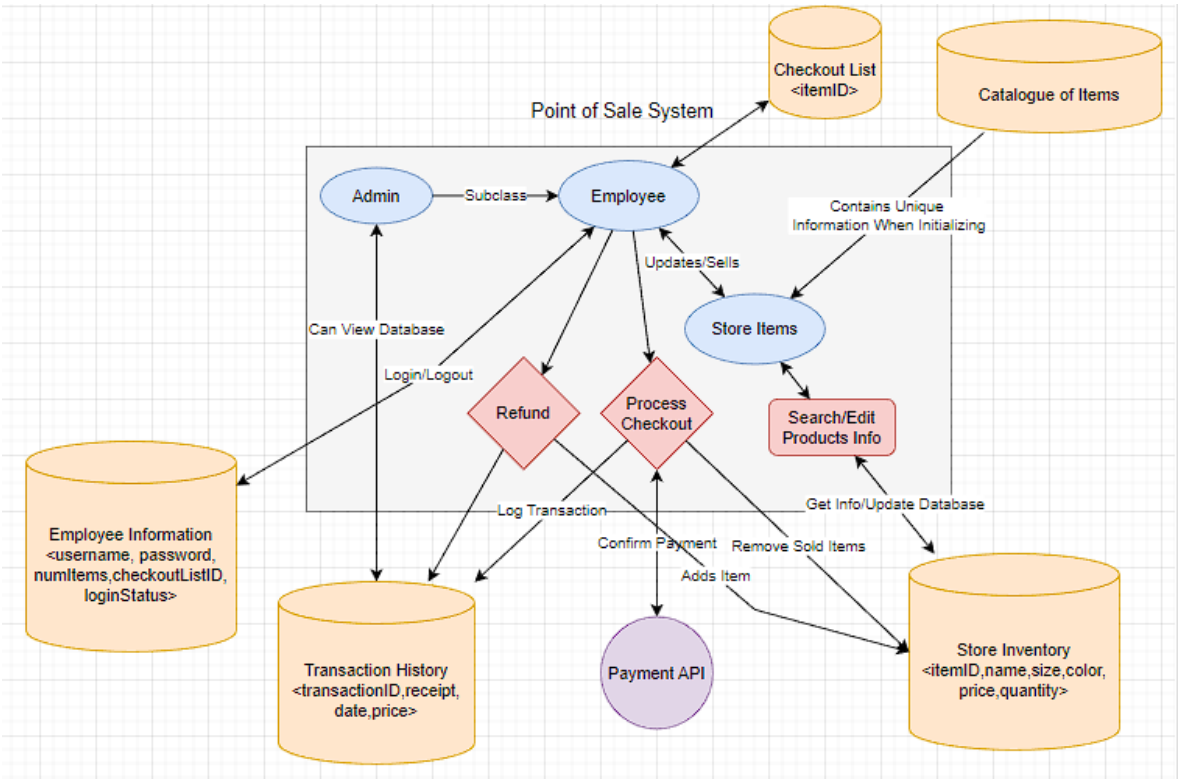
Point of Sale System
Software Design 2.0

By: Matthew Smith, Nicholas Stark, Alicia Loya

4/21/2023

Software Architecture Diagram 2.0



Modifications to SWA

While the core functions of our original system remain unchanged as of now, some modifications have been implemented to reflect changes to our existing database, along with a new way to represent the list of items used in the checkout process. Below are listed changes to our Software Architecture Diagram.

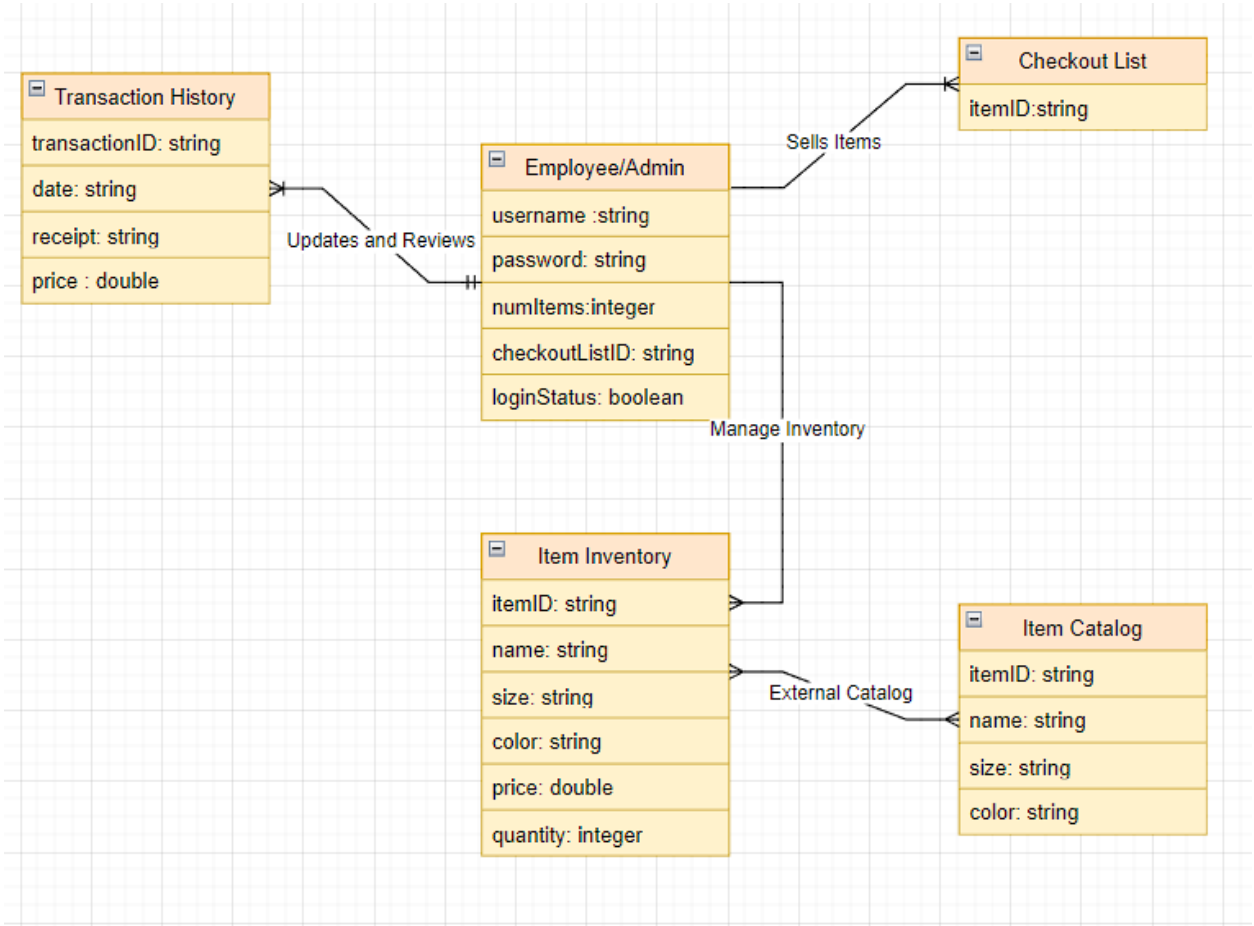
Employee Information

In our initial design, the employee information database was intended to only store information regarding the username and password of the employee/admin users in order to authenticate login, however this database has been modified in order to store other information regarding the employee, that is retrieved when an employee logs into the system with the matching username and password. In addition, the database will also keep records of the current login status of each employee.

Checkout List

This new database will be utilized to keep track of items that are being processed for checkout, this differs from our original design in which each employee would store its own list of items used for checkout. The reason for this change is that in case of any failure or errors during the checkout process, the information of our database will not be updated in order to prevent faulty information within our system and inventory databases.

Data Management Strategy



Tables

In our software system, the employee/Admin is responsible for managing most of the information present in other related databases. Because of this we have chosen to use SQL for its relational database structure, connecting the multiple tables present in the order system. First off, the employee database stores the user's login information, consisting of a username and password in order to authenticate employees accessing the system. In addition, it contains information relevant to employee functions, such as the numItems used for checkout, the login status of employees, and finally, an ID that corresponds to a unique Checkout List for each employee. This ID corresponds to the Checkout List, which stores the list of item IDs, which will then be processed upon checking out and selling the items.

In addition to the item Checkout List, the Employee/Admin is responsible for maintaining the Transaction History Database, which is another database, in which each unique transaction ID is stored with a corresponding date, price, and a string copy of the receipt in order to document the customer transaction in the case of disputes.

In addition to the Transaction History Database, there are two other important databases, the Item Inventory, which contains information about items that are present in the store's inventory, and the Item Catalog, which contains information related to all items, whether or not they have been added into the inventory. The Item catalog database is an external database that is used by our system in order to initialize information for new items based on ID, so that when new items are added into the inventory that did not previously exist, this information can be copied into the store's inventory, and initialized with a price, name, and quantity that are unique to the our store and can be modified when sales happen or when items sell. Both databases contain corresponding fields that are identical, so that items initialized with a unique ID can be copied over with the corresponding information.

Strategy

Because this system relies on the communication between multiple devices, such as Ipad tablets interacting with servers storing data, it is important to implement this database utilizing SQL, as this system will require many transactions between different devices. SQL provides a structured and consistent way for multiple devices to communicate with each other around a centralized database. In addition, the ACID benefits of SQL are important to utilize in our system in order to maintain accuracy. This database is required to be accurate in order to function properly and will need to retain information in case of emergencies, as losing information regarding purchases or item inventory could result in catastrophic consequences regarding mismanagement of inventory, such as theft or fraud. In addition, if purchases are unable to go through, or a device fails to complete any transaction, it is important that errors pertaining to a single transaction do not damage other processes or transactions. As a result, it is important to utilize SQL in order to maintain the integrity of our system in case of unexpected events. In conclusion, due to the relationships present between the entities present in our

system, along with the necessity for accuracy and stability being our most important features for our database, utilizing SQL in order to maintain our system's databases will allow our system to meet the demands expected for our software.

TradeOff Discussions

For this design, we chose to use five different databases for our software system: inventory, transaction history, employee/admin, checkout list, and item catalog. Each database and the design for each is described below.

Employee/admin

The employee/admin database stores the processing information for all employees which work in the store. An alternative noSQL format could be used to store the data more flexibly. Unfortunately for noSQL, there is no hierarchy of data storage here, and the employee/admin database is related to four other databases, which is not suitable for noSQL. A possible alternative for the structure of the data would be to remove checkoutList and numItems as fields, since they would have no relevance to the additional data. In spite of this, these fields are required as part of a relation to the inventory and transaction databases.

Transaction history

The transaction history database stores all of the transactions which have been processed by the system. An alternative noSQL format would be useful to sort the data in a hierarchy, like the date processed or the cost of the transaction. However, SQL is best suited for high-transaction based applications, which this database uses extensively. An alternative way to store the information would be to add a field including the name of the employee who processed the transaction. This would give the user a useful way to identify who was responsible for each transaction processed. This would also constitute a breach of privacy if the system becomes compromised or an administrator feels a mistake was made during the transaction process when no mistake was made. The receipt field could also be removed to save space, but it is necessary for the relation to the employee/admin database because it contains the items sold in the transaction.

Item catalog

The item catalog database stores all the items in the warehouse which are not yet for sale. A noSQL column/wide column store format would enable the system to store the items in the warehouse and query the data much quicker due to the sheer volume. On the other hand, the database is the most important relation component in the system, something noSql does not support. An alternate method for storing the data would be to only include the field for the item by its item ID number. The problem with this method is that the item would only be able to be recognised by the system, not the user, which needs to be done to ensure the transfer of information is performed correctly.

Inventory

The inventory database stores all the items available in the store. If a noSQL document format were to be used, then the data would be able to be more easily kept along with its description. Despite this, the system is a transaction application and the inventory database is a necessary part of the checkout process, which SQL is much better at. No fields in this database can be removed to save space: all are necessary for the relation between the Item catalog and the employee/admin database. An additional field like department could be added for an easier search, but this would only take up space and has no relation to any of the other databases. Also, only employees are going to use the system, so they already know which department the item would belong to.

Checkout List

The checkout list database stores all the ID of the items about to be purchased. A noSQL database would enable the database to be more dynamic and it would be easier to both add and remove data. However, this database is an integral part of the transaction process which SQL supports better than noSQL. Additional fields would enable the user to better read the items in the list, but because this is a list database only the item ID needs to be entered.

Other Alternatives and Trade-offs

All of the five databases are separate, but there are relations between them. An additional database for the employee login could be created to store employee login information which would add additional security to the system. However, the system does not need this data and it would be considered a waste of space. That is why there is a field to see if the employee has logged in to the system. In addition, the checkout list database could be removed to save space, but it would not be able to undergo the necessary changes in data from transaction to transaction. The item catalog and item inventory databases could be combined to prevent redundancy with the items in the inventory given a quantity and price. Since both databases use SQL, a column-based noSQL system would be better suited for that kind of combined database. At the risk of confusion of the user and system backlog, it is better suited to keep those two databases separate.