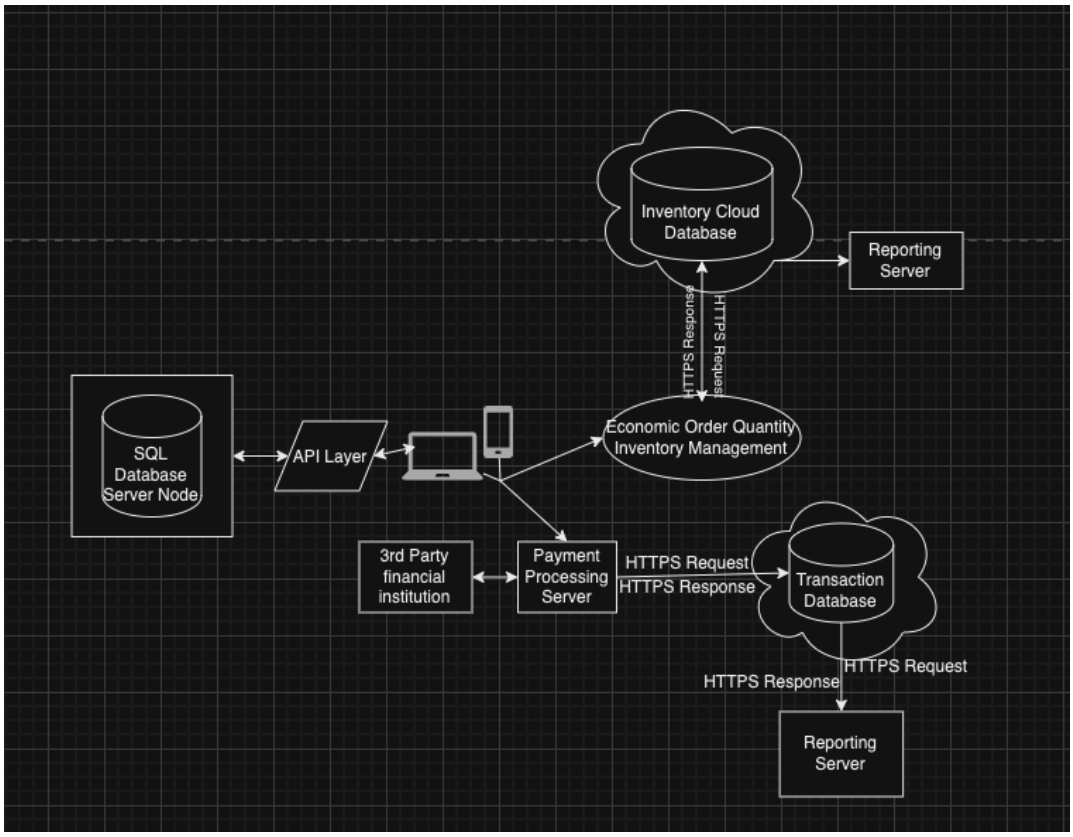
Software Design 2.0

November 10, 2023

Shreya Sridharan, Krish Jhaveri, Erick Hernandez

1. Software Architecture Diagram



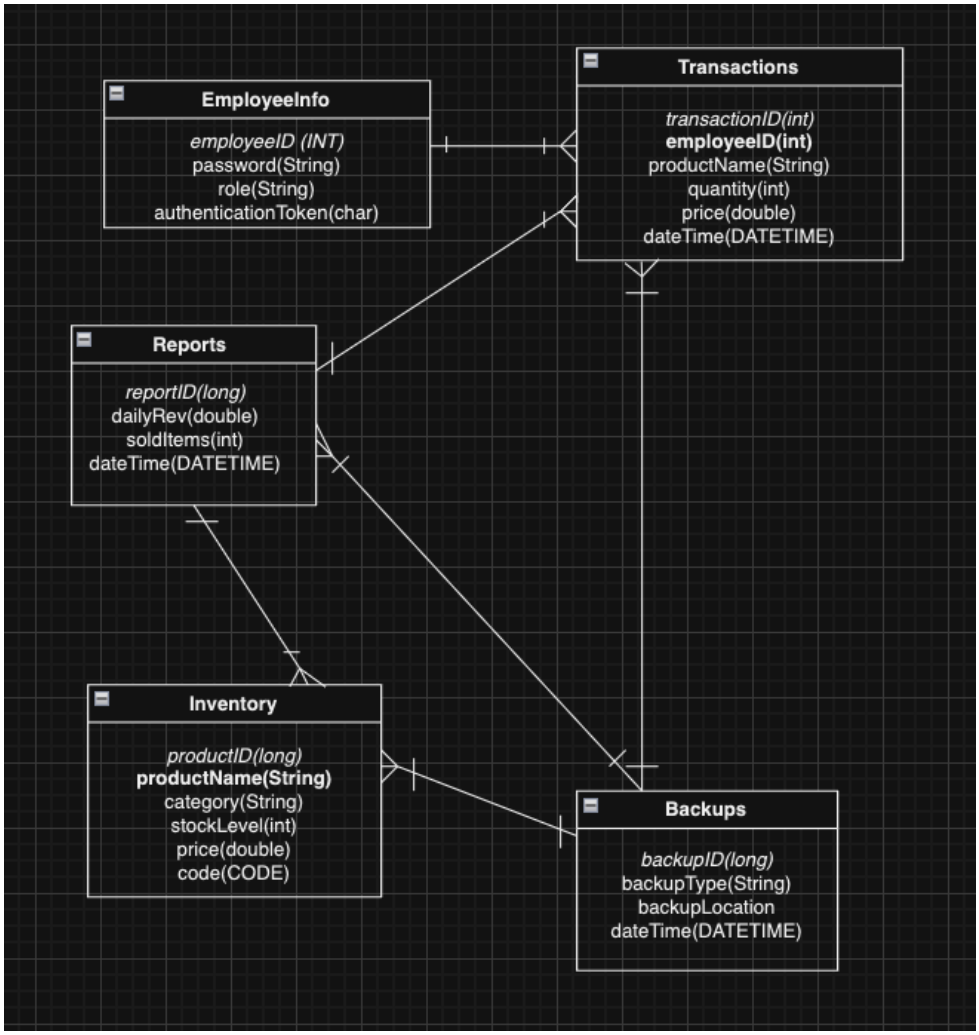
1.1 Description of SWA Diagram

The software architecture diagram has been modified to include the database management representation of SQL. The SQL Database server node functions as the central repository for storing and managing the data related to the clothing store's operation. This component will store critical data such as employee information, transaction details, security information, etc. It also plays a crucial role in managing transactions within a system. This involves handling the execution of SQL queries related to purchases and refunds. It will also facilitate regular backups of the stored data to prevent data loss. If the system ever fails, the backup and recovery mechanisms of the SQL database server ensure that the data will be restored.

The API Layer is another component that acts as an intermediary between the SQL Database Server and the other components. It provides a standardized way for different parts of the software system to interact. This API layer allows components such as the user interface, business logic, and external services to interact with the SQL Database Server. It also manages authentication, which verifies the identity of users and components before allowing access to the

SQL Database Server. Additionally, it handles errors, providing the necessary responses to components when any issues occur.

1.3 SQL Database Diagram

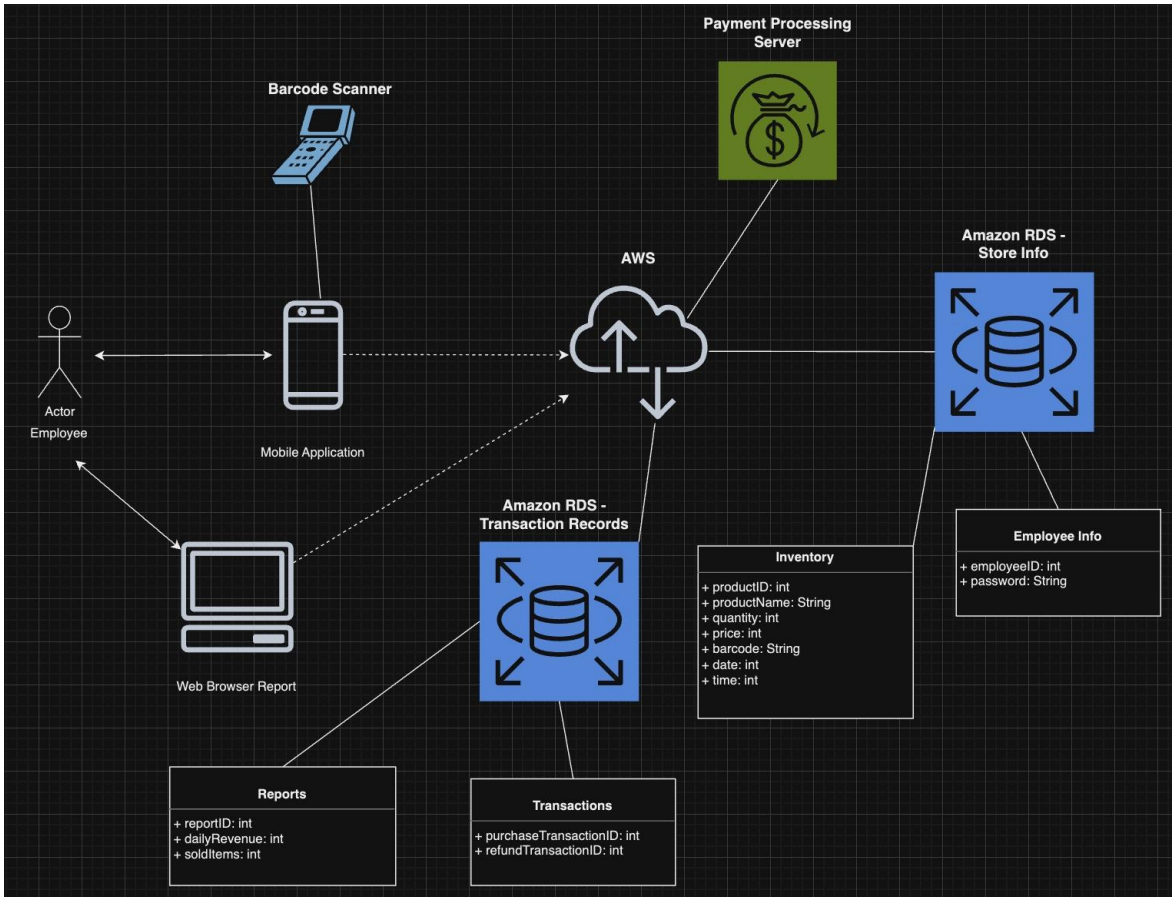


1.4 Database Diagram Description

There are five major database components that will be utilized in this project that are represented in the diagram above. EmployeeInfo will contain the parameters employeeID which is also a primary key, password, role, and authenticationToken to ensure security. Each employee will be able to perform many transactions. The Transaction database contains transactionID which is a primary key, employeeID, which is the secondary key, productName, quantity, price,

and dateTime which is an instance of the class DateTime. The report generating feature is able to produce compilations of both inventory and transactions. It will include a primary key of reportID, and other parameters such as dailyRevenue, and soldItems. The inventory database has parameters productID which is a primary key, productName which is a secondary key, category, stockLevel, price, and code. Finally, the backups class will be able to store all backups of the reports, inventory, and transaction history, and will contain the parameter backupID, and backupLocation which will categorize and store the backup information in the corresponding location.

2.1 Data Management Diagram



2.2 Description of Data Management

The data management strategy diagram for this software system shows that it will consist of two relational databases(SQL) and will be accessed by the actor, in this case the employee, after interacting with the mobile application or web browser. However, in order to access the data the employees need to login with their respective employee ID and password. As shown in the

diagram, the employee ID data type is an integer and will be stored in the cloud using a relational database. The password data type is a string and will be encrypted. Once the employees are successfully logged in, then they will be able to access the inventory and start the purchase function or refund function. The inventory will be stored in the same database that contains the employee login information, but they will be in different tables. For the table containing the inventory it will have different columns to store the product ID as an integer, the product name as a string, the quantity as an integer, the price as an integer, the barcode as a string, the date created as an integer and the time created as an integer.

The second relational database will be used to store the transaction records of the business. This database will have two separate tables, one containing the transactions and the other storing the reports. The transaction table will have two columns where a transaction would either create a purchase transaction ID or a refund transaction ID, both will be stored as integers. For the reports table three columns will be needed since it will store the reports ID, the daily revenue and the items sold. All of these items will be stored as integers. The information that is stored in the second database can be accessed if the employee decides to login using the web browser since this is specifically for the report function.

2.3 Tradeoffs

Trade Offs:

The choice of multiple databases introduces complexity in terms of data consistency and transaction management across services.

SQL databases may have structural changes that require careful conventional strategies.

Alternatives Considered:

Considered using a single, polyglot database-This approach allows different parts of an application to use the database system that best suits their needs. For example, it might use a relational database for structured data like user profiles and a NoSQL database for handling more flexible and dynamic information like product catalogs. The idea is to choose the right tool for the job, creating a more flexible and efficient overall system.

Alternative databases: were considered based on specific use cases (e.g., Redis for caching), but simplicity was prioritized.

SQL vs. NoSQL:
SQL databases provide strong consistency and data integrity but may introduce complexities in scaling and flexibility.
NoSQL databases offer flexibility and scalability but may sacrifice some consistency and transactional guarantees.

Multiple Databases:
Provides modularity but introduces challenges in maintaining data consistency across services.
Allows for independent scaling but requires careful coordination in data updates.
Organization of Data:
Logical separation of data allows for better management of microservices but requires more careful coordination to maintain consistency.

In conclusion, the chosen architecture is SQL database; this is because they are likely organized filing cabinets for information, ensuring data consistency, integrity, and reliability. They're great for scenarios where the relationships between different pieces of information are crucial, such as managing user profiles or tracking order details. While SQL databases might require careful planning for changes (like adding new types of information), their strength lies in providing a structured and organized environment for data, making them a solid choice for applications that demand a high level of data integrity.