Software Design Specification

Clothing Store POS Software

Prepared by Shreya Sridharan, Krish Jhaveri, Erick Hernandez

Version 1.0

October 13, 2023

1. Introduction

1.3 Purpose

This document will highlight the technical and software details that will go into the development and design of the clothing store POS system. It contains specific information about the inputs, outputs, classes, operations, as well as the specifications of the software architecture.

1.2 System Description:

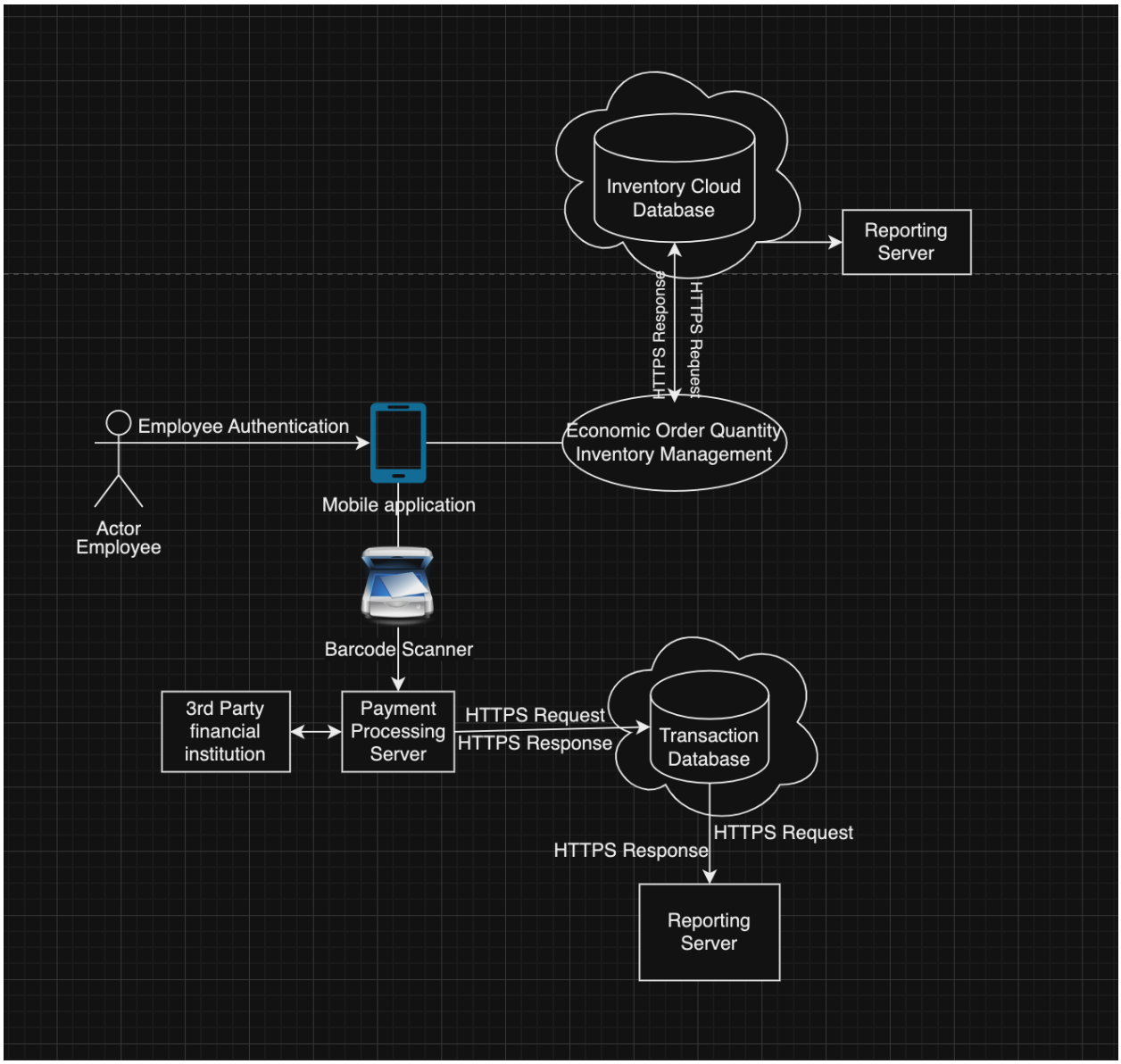
The system will allow employees to login using an employee ID and password or scanning of their employee ID card. Once they have logged in, they will enter the main page to select the action that they would like to perform. The user interface will be very simple and easy to navigate. The employee can select whether they would like to perform a transaction, and then have the option to refund or purchase. A barcode scanner will allow items to be added to a receipt, as well as employee IDs to be scanned. A card/chip reader will be employed to complete transactions. Employees can also choose to create reports based on the criteria that they input. For example, daily closing reports for cash/checks received, inventory reports for certain products, etc. Another module that employees can access will be the inventory module where they can access inventory and manage the inventory counts and updates using a categorically separated system. The software will be coded using JavaScript to ensure mobility between different devices and systems.

1.3 Brief overview of software

The system will contain hardware and software components. It will utilize at least two cloud databases that will allow the system to store information that can be accessed from any of the three locations and will allow for further expansion. The system should also be able to access third party payment processors for payment of credit or debit. The hardware components include a barcode scanner and cash register that can accommodate any cash payments.

2. System Architecture Overview

2.1 Software Architecture Diagram



2.2 Description (SWA Diagram)

The System Architecture Overview (SWA) diagram provided depicts a comprehensive system that integrates various components to facilitate employee-driven processes within an organization, primarily focusing on inventory management and financial transactions. Breaking down the key components and interactions in this diagram:

The employee is the primary user of the system. They access the system through a mobile application to perform various tasks related to inventory management and financial transactions. Before gaining access to the system, employees must authenticate themselves, ensuring secure access and data protection. The hardware barcode scanner within the mobile application enables employees to scan barcodes on products, making it easier to manage inventory and track items

The first component that the app is capable of is the transaction and POS software. The barcode scanner is used by the payment processing server. By interacting with the barcode scanner, the software will give the user the option to perform a purchase or refund. In either case, it will go through the payment processing server which is a specialized server which is responsible for handling financial transactions. This server interacts with various payment gateways and financial institutions to perform transactions that are secure and accurate.

Each transaction will have to be stored in a cloud database, which will include sales, returns, and any other transactions. Storing the transaction data in the cloud ensures real time access and analysis of the records. This system contains at least two cloud databases. One for transaction history and another for inventory management. Both databases are connected to the payment processing system and inventory management system respectively. They simply store the resulting information from each system.

The other component of the system is the inventory management system. We will utilize an economic order quantity (EOQ) inventory management system that is a formula used in inventory management to determine the optimal order quantity for a product. An EOQ inventory management system is a software that uses a formula to help organizations manage their inventory efficiently. It will categorize each of the products by the product type- i.e shirts, pants, accessories.

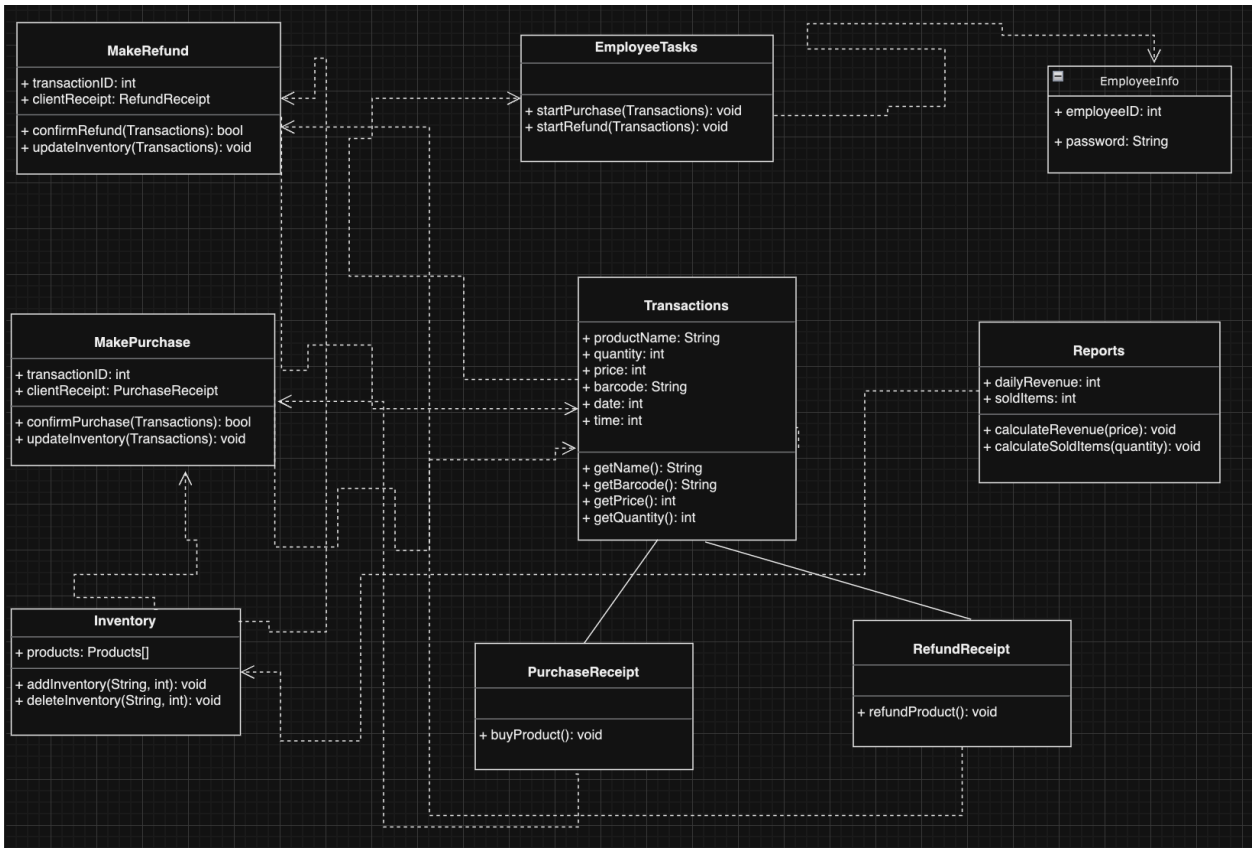
The inventory information must be stored in a cloud database as well separated by location. This data includes product details, stock levels, etc. Cloud based databases offer advantages such as scalability, accessibility, and data security.

The reporting server is dedicated to generating and managing reports related to inventory management and financial transactions. It processes data, formats it into a PDF or spreadsheet format and delivers it to the users. It utilizes the data from both the transaction and inventory cloud databases.

Overall, the system also sends https requests and responses. A HTTPS request is a type of communication between a client (web browser) and a server over a secure and encrypted connection. An HTTPS response is the data provided by a web server in response to an HTTPS request. It includes requested content like web pages. It ensures data security during transmission.

In summary, the provided SWA diagram represents a comprehensive system designed to streamline inventory management and financial transactions for employees within an organization. It emphasizes security and data protection through HTTPS communication, uses external financial institutions for payment processing, and employs dedicated servers for transaction data storage and reporting. The incorporation of Economic Order Quantity and inventory cloud database reflects a commitment to efficient and data-driven inventory management practices

2.3 UML Class Diagram



2.4 Description (UML Diagram)

As the UML diagram shows, under EmployeeInfo first the employee needs to login using the employee ID provided as an integer and password provided as a string. Once the employee logins in, the system will have the option to perform mainly two tasks or operations. This is why after the EmployeeInfo is given successfully the diagram goes to the class name of EmployeeTasks. The employee can start a purchase and the other option would be starting a refund. This explains why the diagram shows the operations of startPurchase() and startRefund(). Both operations take the parameter of "Transactions." Because both options are technically a transaction, the diagram then moves to the class name of Transactions. This provides the productName with the attribute type of a string, then the quantity as an integer, the price as an integer, the barcode as a string, and finally the date and time as an integer. In addition, this class has operations like getName() and getBarcode(), both will return a string. This class also includes the getPrice() and getQuantity() operations, both returning an integer value.

Depending on the transaction being made the system will move to the class names of MakePurchase or MakeRefund. Both classes have a transactionID attribute as an integer, but if a purchase is being made then the attribute clientReceipt will be PurchaseReceipt from the class PurchaseReceipt. Otherwise, if a refund is being performed the attribute clientReceipt will be taken from the class RefundReceipt. The class MakePurchase has operations of confirmPurchase() and updateInventory(), both using "Transactions" as a parameter but confirmPurchase() will return a boolean value and updateInventory() will return a function. Similarly, the class MakeRefund has operations of confirmRefund() and updateInventory(), both using "Transactions" as a parameter but confirmRefund() will return a boolean value and updateInventory() will return a function.

Now, going back to the classes MakePurchase and MakeRefund. If the confirmPurchase() operation from MakePurchase returns true then the next proceeding step will be a receipt. This receipt will come from the class PurchaseReceipt that has the operation buyProduct() as a function. The PurchaseReceipt class will have the attributes from the Transaction class. However, the confirmPurchase() operation can return false if the product is not in stock anymore. Similarly, if the confirmRefund() operation from MakeRefund returns true then the next proceeding step will also be a receipt. This receipt will come from the class RefundReceipt that has the operation refundProduct() as a function. The RefundReceipt class will also have the attributes from the Transaction class.

Another class that depends on the classes MakePurchase and MakeRefund is the Inventory. This class includes the products attribute with Products[] as attribute type. This means that it will contain all the products in an array. Additionally, the Inventory class has the addInventory() and deleteInventory() operations. Both operations take a string and integer as parameters and will return a function. For instance, if a product is being purchased then the quantity purchased will be deleted from the inventory. However, if a product is being returned then the quantity will be added to the inventory.

Finally, this system will be able to create a report. This is why the class Reports is included in the diagram and it will depend on the Inventory class. The Reports class includes the dailyRevenue and soldItems attributes, both provided as integers. Also, this class has the calculateRevenue() operation taking “price” as the parameter, and calculateSoldItems() operation taking “quantity” as the parameter. Both operations return a function. These operations will be performed according to the updates made in the Inventory class.

3. Development Plan

3.1 Team Members

For the development and implementation of this software system, we will need system designers, developers, testers, managers, security officials, data analysts, and client and business analysts. This will require a team of around 8-12 people. For the system designers, we will require a software architect and UI/UX Designer. The development team will consist of both front-end and back-end developers, as well as a hardware integration developer. The testing team can have 1-2 people who test quality assurance. The management team will consist of a release manager, client relations manager, and data manager. We will also require a security and compliance officer.

3.2 Partitioning of Tasks

A list of the tasks and responsibilities that each role is responsible for has also been included in list format below:

- Software Architect: Create the high-level system architecture and define the components and data flow
- UI/UX Designer: Design the user interface and experience for the mobile app and the web site
- Front-end Developers: Build the front end portion of the web application and commit the technical implementation of that UI/UX designer
- Back-end Developers: Implement the application layer, integration with the databases, and any other server side software

- Hardware Integration Developer: Develop the functionality to integrate with the hardware such as the barcode scanner and the cash register
- Tester: Perform testing- functional testing, user acceptance testing, and security testing- as well as in-store testing.
- Release Manager: Plan and oversee the release of the system as well the deployments and updates of future versions
- Client Relations Manager: Communicate and interact directly with the client and act as a mediator between the team and the client to accommodate any other needs or arising issues
- Data Manager: Manage the data storage and ensure capacity, security, and accessibility of the database in the cloud format.
- Security and Compliance Officer: Ensure that the system complies with data protection regulations

3.3 Timeline

The project will take an estimated time of around 7 months. The timeline has been outlined below in week to week format.

Month 1: Project Initiation

- Week 1-2: Software Architect creates system architecture
- Week 3-4: UI/UX Designer starts design work, and front-end and back-end developers communicate

Month 2-4: Development Phase

- Week 5-7: Front-end developers build UI based on design
- Week 8-10: Back-end developers work on application layer and the database integration
- Week 11-13: Hardware Integration developer begins working on incorporating the hardware components
- Week 14-16: Software Architect oversees the component integration

Month 5-6: Testing and QA

- Week 17-18: Tester performs testing for the current stage of development, as well as in store testing

- Week 19-21: Back-end developers fix any bugs and optimize the system based on tester's information and input

Month 6-7: Deployment and Release

- Week 22-24: Release Manager begins coordinating the system's release
- Week 25-27: Deployment of the system, in both web and mobile app formats
- Week 28: Any necessary updates and maintenance

There are also some components of the system that must be completed throughout the entirety of the project. The client relations manager must engage with the client at the start of the project when interviewing them about their needs, as well as throughout the project to collect feedback and ensure that the project aligns with the client's needs. The data manager must also work throughout the project to ensure that the data storage meets the capacity, security, and accessibility standards. Finally, the security and compliance officer must ensure that the project and software system meets security standards and complies with data protection regulations.