Software Design Specification: Test Plan

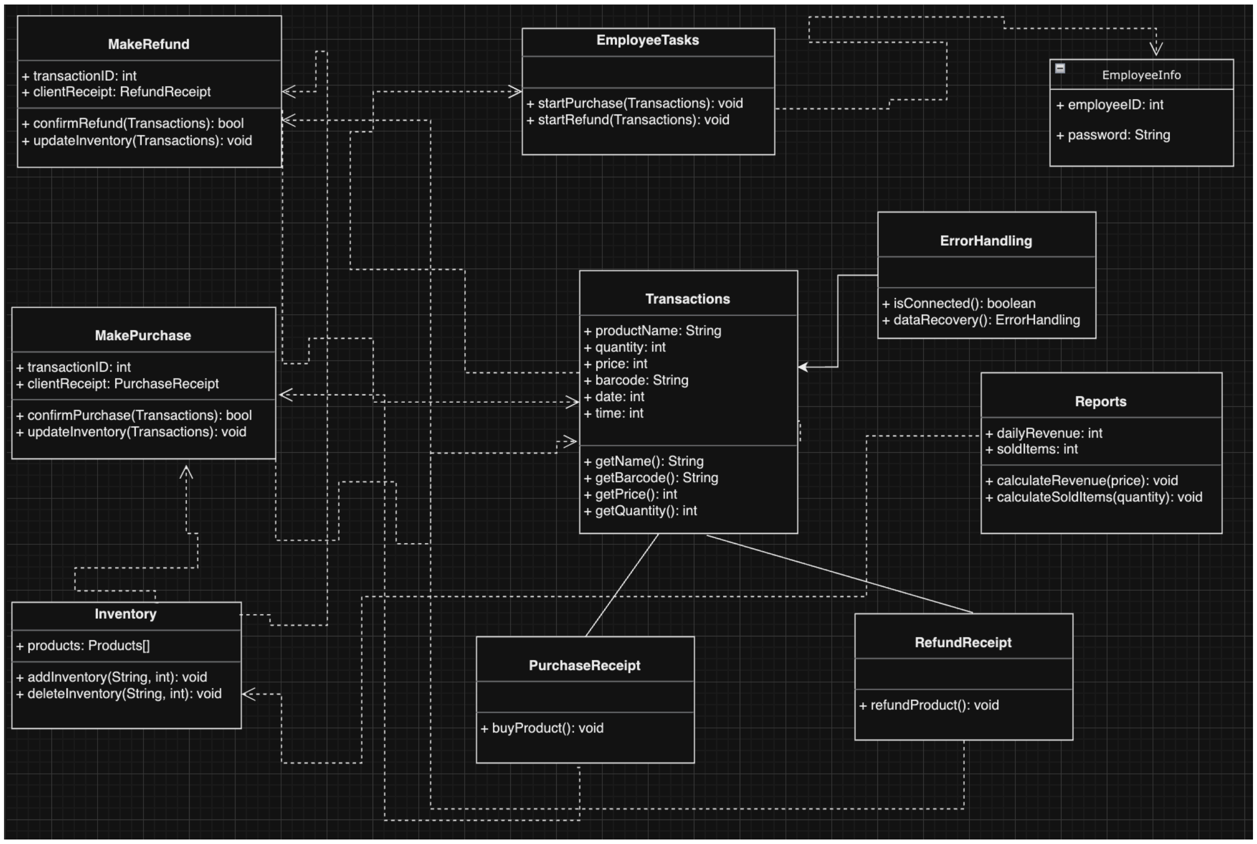
CS 250

27 October 2023

Shreya Sridharan, Krish Jhaveri, Erick Hernandez

1. Software Design Specification

1.1 UML Diagram



1.2 UML Diagram Description

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.

This system must also include a class that will handle errors by recognizing if the network is connected, or if data has been deleted. The operations isConnected() and dataRecovery() both complete the respective error handling requirements. isConnected will check if the network is connected before any action is performed, and will display a corresponding error message to the user. dataRecovery() will implement data recovery software to recover any deleted files or in cases where the file system is severely damaged or goes missing, data recovery tools will search for the file signatures to reconstruct the data.

2. Verification Test Plan

System Test Cases

Feature: System Compatibility

1. **Test set 1: Accessing the website and app from different devices/web browsers**

→ verify that the website and the application can be accessed by different devices and web browsers.

Testing:

→Open the website on Firefox, Chrome, Safari, and Bing and ensure that it opens.

→Download the application on both Android and iOS and ensure that it works as expected.

2. **Test set 2: Verify that all the functionalities work as expected across all devices**

→ After ensuring that the application and website can open on all devices and platforms, ensure that the three main components of the system function as expected (transaction processing, inventory management, report generation).

Testing:

→Add an item to a transaction and complete a purchase transaction on both android, iOS, and the search engines.

→Print a report of transaction history.

→Add an item to the inventory management system and then delete it.

Feature: Concurrent Usage

1. **Test set 1: Simultaneous transaction processing**

→ Verify that employees are able to perform several transactions at once.

Testing:

→ Complete two purchase transactions at the same time from two different devices.

→ Complete two return transactions at the same time from two different devices.

→ Ensure that the transaction report reflects the user/device, time of transaction.

2. **Test set 2: Monitoring the system performance and response times**

→ If several transactions are being processed at the same time, does the response time slow down?

Testing:

→ Complete to purchase transactions at the same time and compare speed to complete the transaction to a transaction completed on a single device.

→ Increase the number of devices simultaneously completing the transaction to ensure speed is consistent.

Feature: Error Handling and Recovery

1. **Test set 1: Network Disruptions**

→ How does the system respond to network disruptions from the mobile application.

Testing:

→ Perform a purchase transaction and disable network connectivity. The system must display an error message that the transaction was not completed and inform the user to try again.

→ Attempt to generate a report without network connectivity. The system should display an error message or alert that the network cannot be reached and the report cannot be generated.

2. **Test set 2: Data loss or corruption**

→ How does the system respond to any data loss or corruption of any data files?

Testing:

→ Initiate a data backup, verify that the backup process completes without any errors. Ensure the backup files are in the correct backup location.

→ Delete a specific dataset to simulate data loss. Initiate a data recovery process. Verify that the lost data is successfully recovered.

Unit Test Cases

Feature: Inventory Management

1. **Test set 1:Adding items inside the inventory**

→ Verify that the system can add items to the inventory database accurately.

Testing:

→ Add a new clothing item with details (name, size, color, price, quantity).

→ Ensuring that the item is correctly stored in the database or not.

→ Validate that the item's information matches the data inputted.

2. **Test set 2:Calculating Discounts**

→ Ensuring if the discount calculation works properly or not.

Testing:

→ Input a variety of items with different discount rules (e.g., buy one get one, percentage discounts).

→ Verify that the system calculates the total price correctly.

→ Confirm that discounts are applied according to the rules.

3. **Test set 3:Updating Inventory**

→ Confirm that the system helps update the items in the inventory per time.

Testing:

→Modifying prices and quantity and picture if needed of the existing items.

→Verify that the changes are saved in the inventory.

→Test updating items that do not exist and confirm proper error handling.

Integration Test Cases

Feature: Printing Receipt

1. **Test set 1: Printing purchase receipt.**

→ Verify the receipt prints effectively and has the correct purchase information.

Testing:

→ Buy a product noticing all the details.

→ Verify the receipt includes name, price, quantity, date and time.

→ Confirm the product is deleted from the inventory.

2. **Test set 2: Printing refund receipt.**

→ Verify the receipt prints effectively and has the correct refund information.

Testing:

→ Refund a product noticing all the details.

→ Verify the receipt includes name, price, quantity, date and time.

→ Confirm the product is added to the inventory.

Feature: Calculating Reports

1. **Test set 1: Calculate Revenue**

→ Calculate all the revenue using the price data provided from the inventory.

Testing:

→ Add all the sales from products bought in the last hour.

→ Subtract all the sales from products refunded in the last hour.

→ Calculate revenue according to the purchase and refund transactions.

→ Confirm the report has a parallel to the inventory.

2. **Test set 2: Calculate Items Sold**

→ Calculate all the items sold using the quantity data provided from the inventory.

Testing:

→ Add the quantity from products bought in the last hour.

→ Subtract the quantity from products refunded in the last hour.

→ Calculate items sold according to the purchase and refund transactions.

→ Confirm the report has a parallel to the inventory.