Developer’s Manual

What is the point of sales

Table of Contents

Introduction	2
Getting started	2
System requirements	2
External elements	2
Developer's guide	3
Section 1: Introduction	3
Section 2: Use Case 1: System Shutdown and Startup	4
Section 3: Use Case 2: User Management	5
Section 4: Use Case 3: Process Sale	6
Section 5: Use Case 4: Process Rental	10
Section 6: Use Case 5: Process Return	11
Section 7: Use Case 6: Graphical Interface	12
Section 8: Use Case 7: Data Analytics	13
Section 9: Use Case 8: Multiple Users	13
Glossary	14

Introduction

What is the Point of Sales? terminal, hereby referred to as the POS, is a point of sale terminal developed by the genius team of Clayton Barber, Brandon Barton, Max Hasselbusch , Eric Metcalf, and Declan Brennan. It is a software deployable package, allowing the end consumer to use hardware they already own. This manual serves to educate future developers about the core functionality and how to approach the source code.

Getting Started

System requirements: The POS system has limited physical requirements for the devices it will be deployed on. The customer must have:

- Internet connectability, either through a hard connection or wifi
- External credit card reader
- Ten digit data entry pad
- A computer that has java 7 installed.
- the jar file that one can receive through emailing mah318@lehigh.edu

External elements: The one external element linked to in our system is the tax calculator. It allows for any customer to connect to the tax module respective to their state of operation. In the appendix we have inserted the api which we assume that the tax calculator has. The UML diagram for the tax calculator class is shown below:

Developer's guide

Section 1:Introduction to Our POS System as a whole

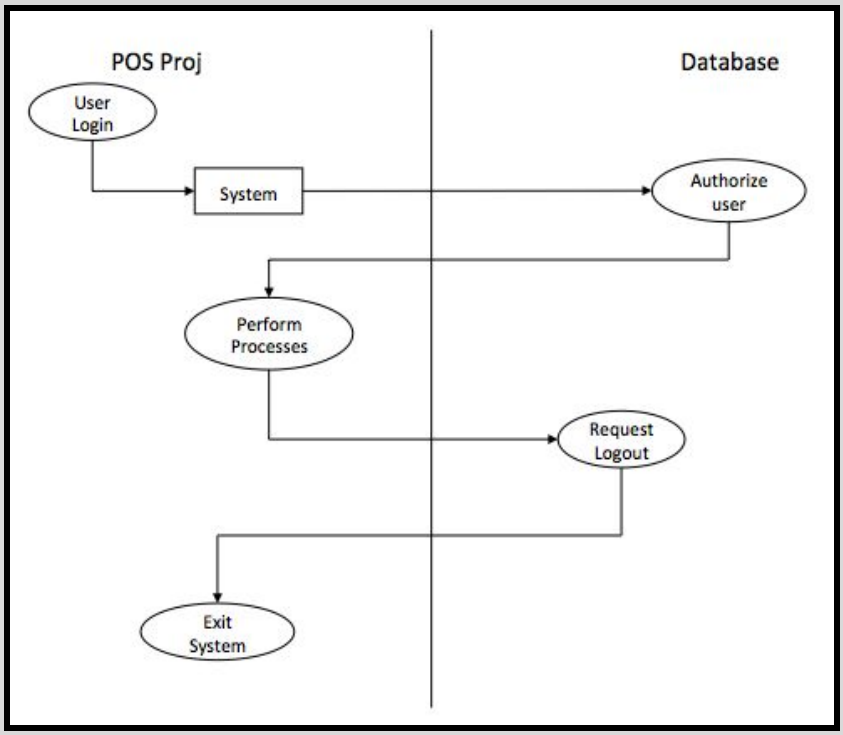
Looking at our system, we set up many different classes and designed a system that reduces low coupling and successfully implements all the necessary use cases. In this section we will point out the most important classes and focus on them. We will give an overview about how to best go about debugging. Then we will step through each of the use cases and demonstrate how we implement each of the different use cases. There are so many classes that we will not go through and explain them all but the execution is described throughout the code. This overview will help you limit the amount of time needed to learn the code base.

The first class that executes the different functions is a class called PointOfSale. This class holds the main function. This main function initiates the database connection along with initializing a few of the different classes that will be use for later. For example, this class creates a sys class which is the main class to create the different objects used later. The point of sale class then creates a login class which then performs the next set of instructions. When do login is created, many of the different gui components are created. Then after login is executed and the database is updated, login calls sys frame which sets up the gui for the system and user management. If the user management is called, then it takes the user into the code that authorized users are able to perform the necessary actions. This will be described more in our user management use case description. If the user does not go into the user management, then the sales gui object comes up. This will allow users to pick through gui components and successfully and build many different. They are able to process the different sales during this frame. Uses many of the different objects such as item, money, Sale, return, sale, and sales frame. Many of the other classes are also used and their descriptions are given in the code. We will explain more about the implementation during the different use case documents.

Section 2: Use Case 1: System Shutdown and Startup

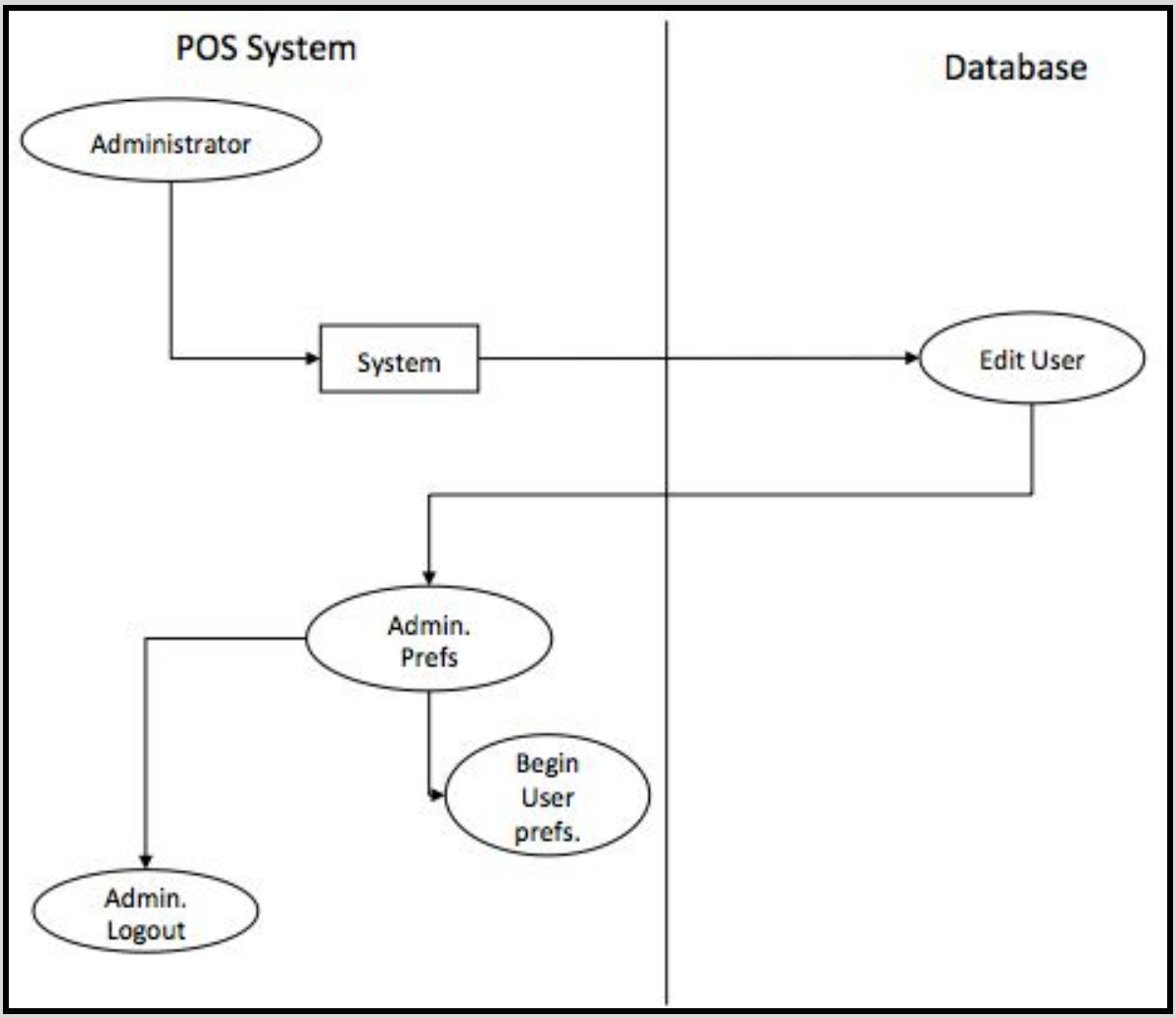
The PoS starts up after running the Makefile found inside /pos after extracting it from pos.zip within the Downloads folder. Running this Makefile only works on a Mac, so if a Windows computer is in use, run pos.jar found in /pos/dist There are two frames that can be accessed after login: the user frame and the admin frame. To log into the user frame, enter *buschmaster* as the username and *1234* as the password. To access the admin frame, enter *sickmedic* as the username and *0632* as the password. Because the database contains all login information, the PoS must be connected to the internet to login under a username and password. If the user attempts to log in without an internet connect, the override key of *0248* can be entered to access the system. System shutdown will occur anytime the user frame or admin frame (whichever is currently being used) is closed.

Before login, a connection to the database will be established and the Sys and LoginNew objects will be created. Inside LoginNew, a login frame will be initialized with which the user will interact. If a connection cannot be established, OSys and Login objects will be created. Both of these objects function the same as Sys and LoginNew expect that only a user frame will be created. Once proper credentials are entered, either the PointOfSale class method doWork() or doAdminWork() will be called, depending upon which credentials are entered. These methods initialize the main functions of the PoS and at this point system startup is complete.



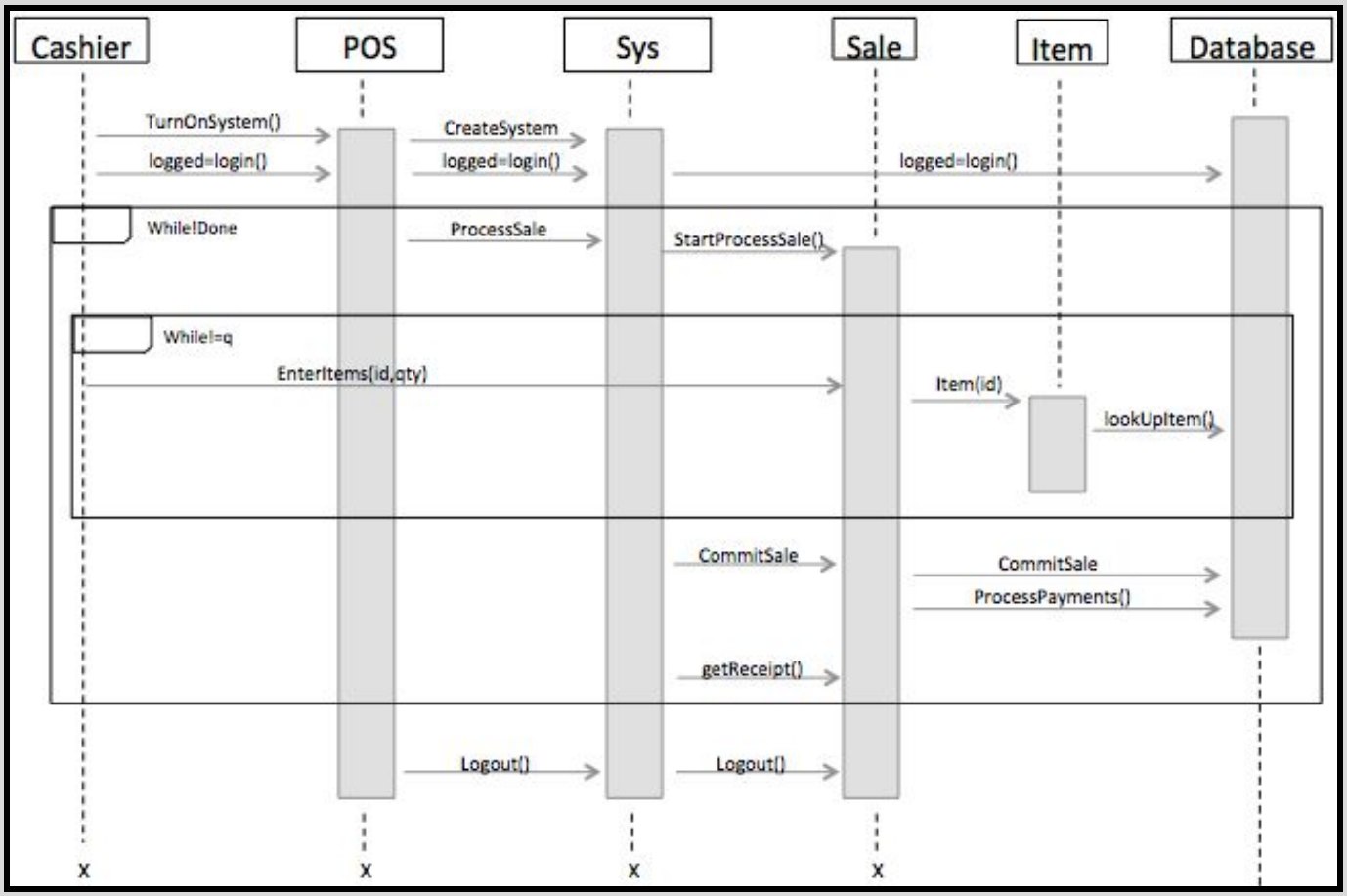
Section 3: Use Case 2: User Management

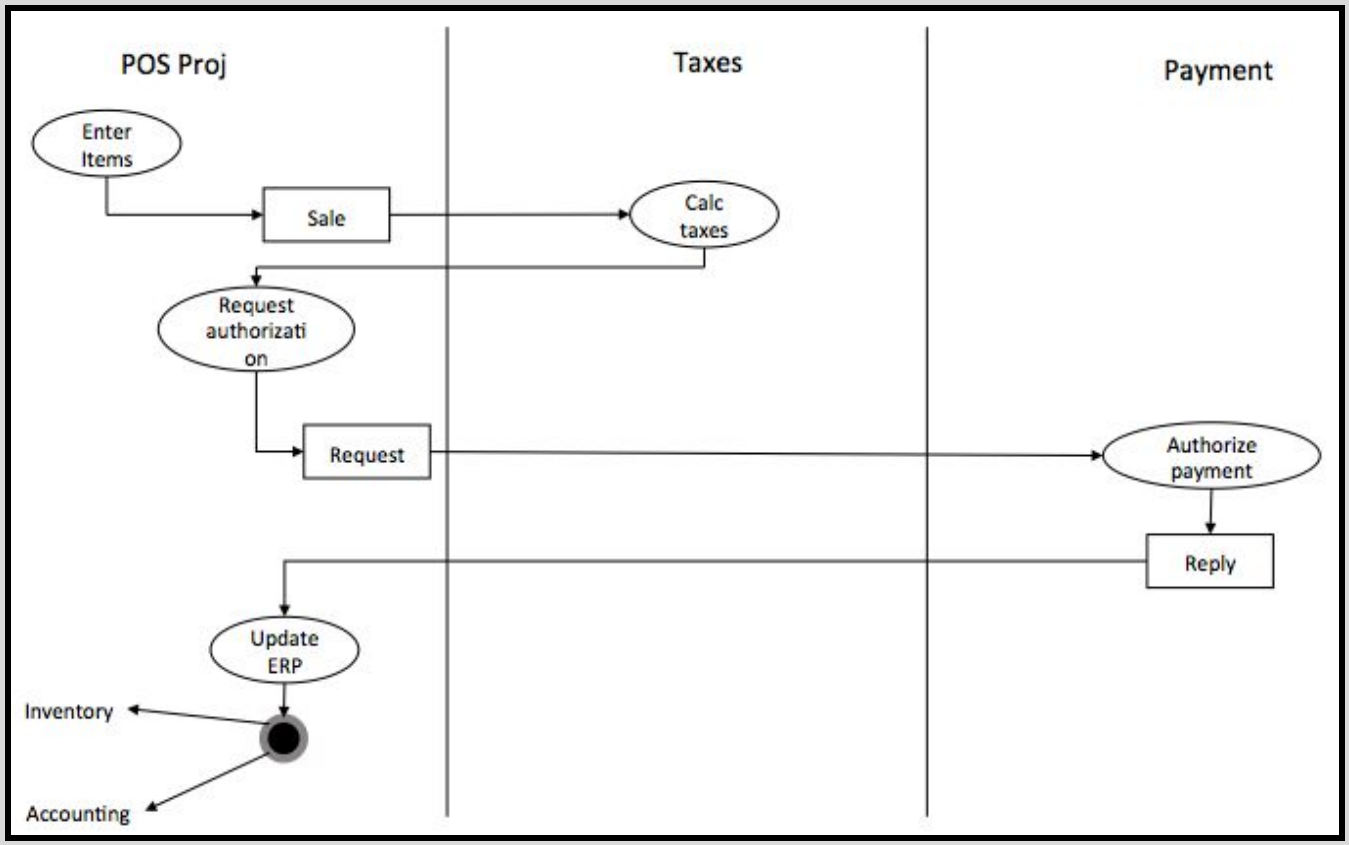
Since we implemented user logins with separate IDS, we needed to find a way to be able to update the different database so that different users can use the system. When considering this use case, we also wanted to make sure that many of the other users can also use this feature at once. I will talk about this more in the multiple users use case section. The idea of our user management system is that we want an authorized user to be able to easily add and delete users from the database. During the login section, the user has the ability to login as an authorized user. They are able to go in and manage the different transactions and different users. We built this through a basic gui interface that allows the user to have many different options. The different classes used are, AdministrativeFunctions, RemoveEmplyeeInfo, and EditEmployeeInfo. Here are the use case diagrams and the data views for the different diagrams.

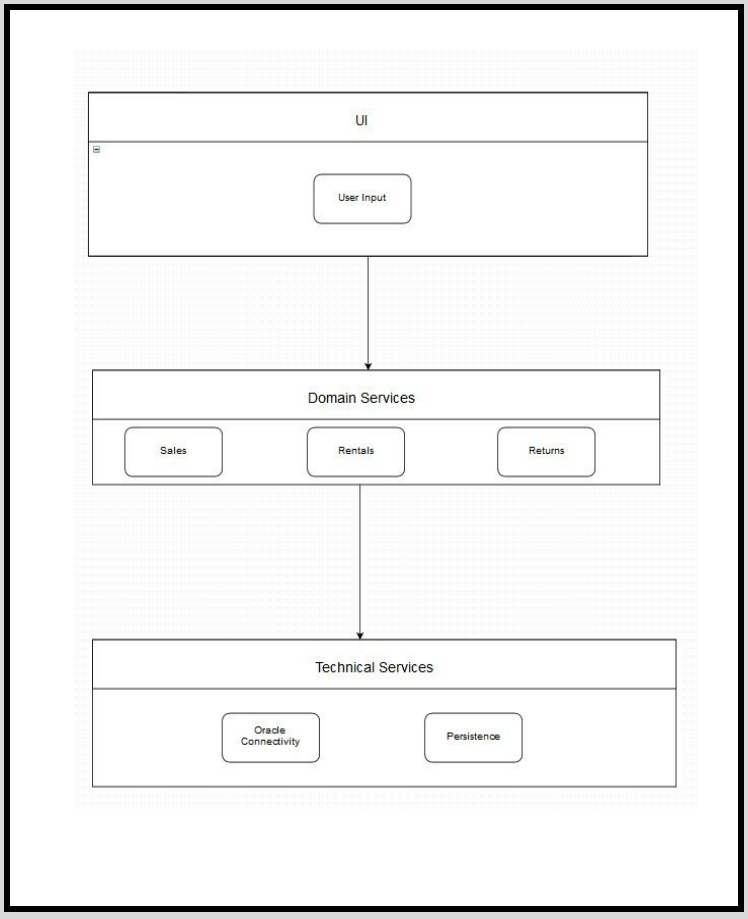


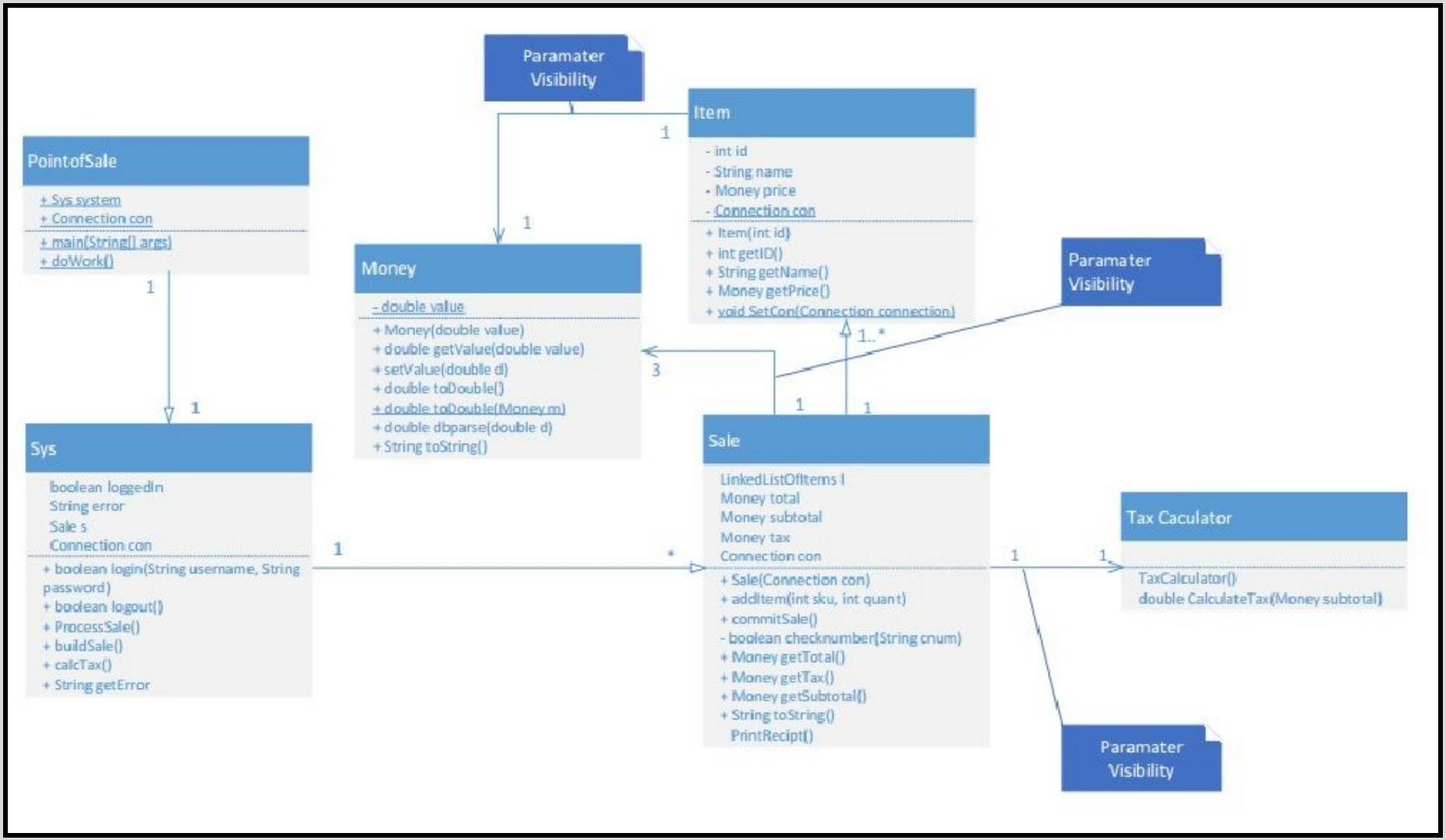
Section 4: Use Case 3: Process Sale

The core functionality of our system is the process sale funtion. A store manager is able to scan all of the items a customer wishes to purchase, query their respective data base from the number provided on the bar codes, assess the tax burden and with holdings of the order, and then process the order with either cash or credit card. Below, we will walk through how we implemented each step in this process. Building the cart. The process begins with the customer pressing the “add item” button. this initializes a sale object. From here, each press of the “add item” button creates a new item object and adds it to the sale object’s linkedlist. This process continues until the user presses the checkout button. A receipt is printed for the user, and the POS queries the database and adjusts the inventory levels, sales data, and tax data.


Cashier
POS
Sys
Sale
Item
Database
TurnOnSystem()
logged=login()
CreateSystem
logged=login()
logged=login()
While!Done
ProcessSale
StartProcessSale()
While!=q
EnterItems(id,qty)
Item(id)
lookUpItem()
CommitSale
CommitSale
ProcessPayments()
getReceipt()
Logout()
Logout()
X
X
X
X

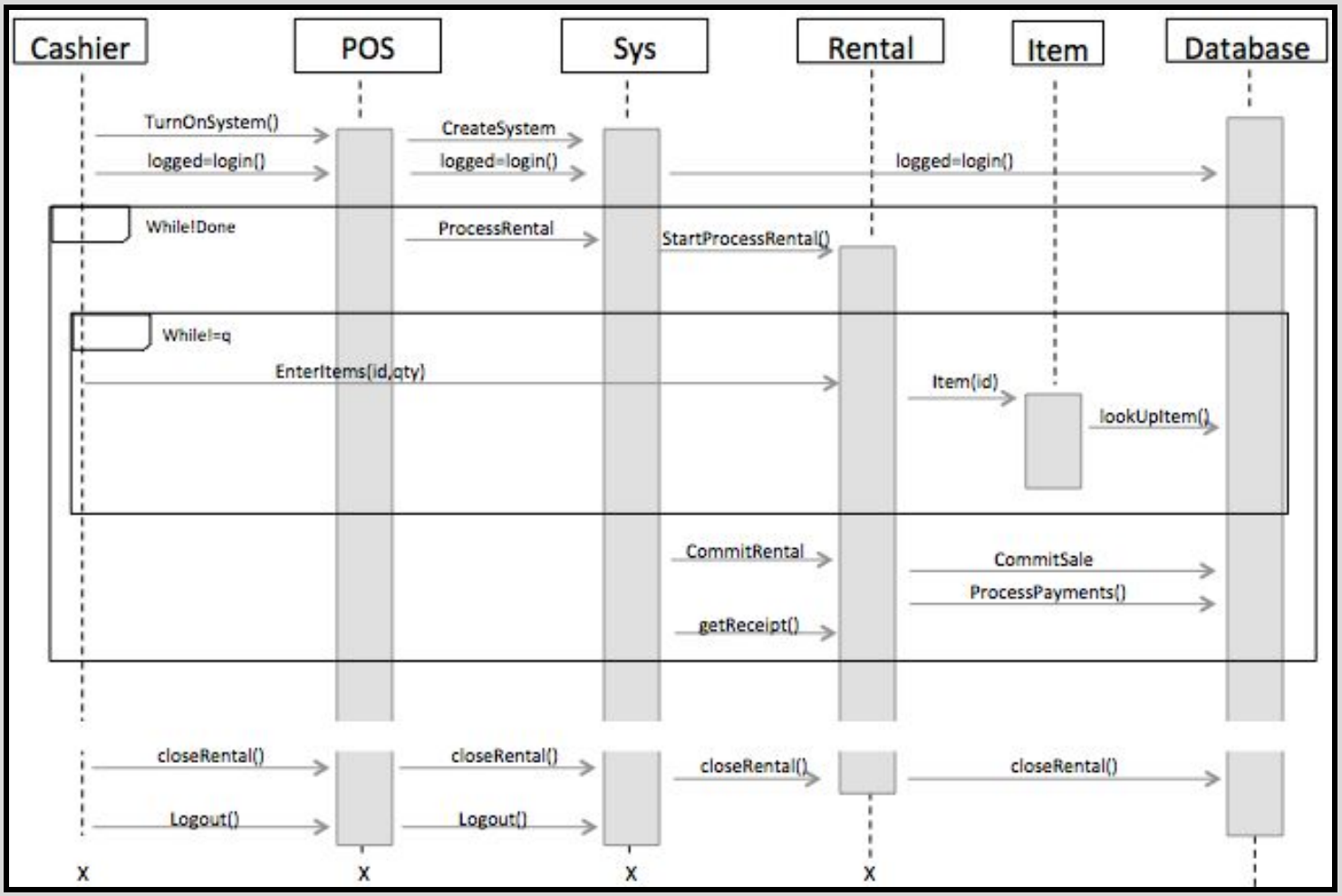

POS Proj
Taxes
Payment
Enter Items
Sale
Calc taxes
Request authorizati on
Request
Authorize payment
Reply
Update ERP
Inventory
Accounting

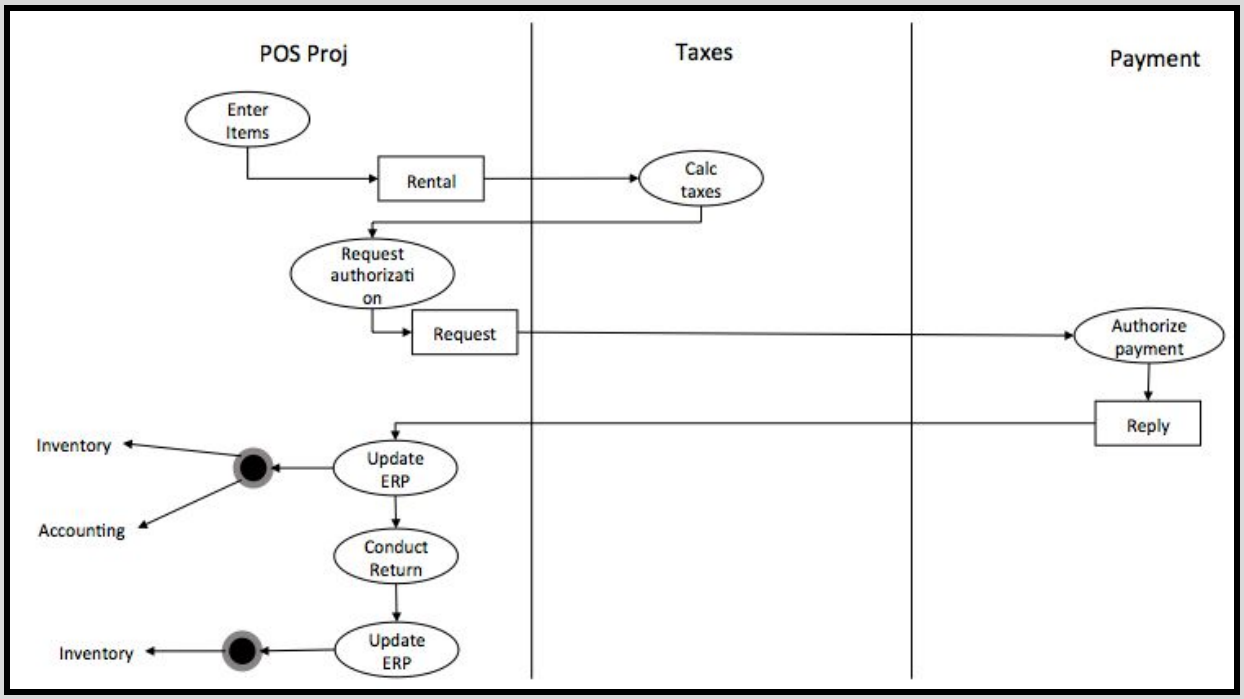

UI
User Input
Domain Services
Sales
Rentals
Returns
Technical Services
Oracle Connectivity
Persistence


Paramater
Visibility
Item
- int id
- String name
- Money price
- Connection con
+ Item(int id)
+ int getID()
+ String getName()
+ Money getPrice()
+ void SetCon(Connection connection)
PointofSale
+ Sys system
+ Connection con
+ main(String[] args)
+ doWork()
Money
- double value
+ Money(double value)
+ double getValue(double value)
+ setValue(double d)
+ double toDouble()
+ double toDouble(Money m)
+ double dbparse(double d)
+ String toString()
Paramater
Visibility
Sys
boolean loggedIn
String error
Sale s
Connection con
+ boolean login(String username, String password)
+ boolean logout()
+ ProcessSale()
+ buildSale()
+ calcTax()
+ String getError
Sale
LinkedListOfItems l
Money total
Money subtotal
Money tax
Connection con
+ Sale(Connection con)
+ addItem(int sku, int quant)
+ commitSale()
- boolean checknumber(String cnum)
+ Money getTotal()
+ Money getTax()
+ Money getSubtotal()
+ String toString()
PrintRecipt()
Tax Caculator
TaxCalculator()
double CalculateTax(Money subtotal)
Paramater
Visibility
1
1
1
1
1..*
3
1
1
1
*
1
1

Section 5: Use Case 4: Process Rental

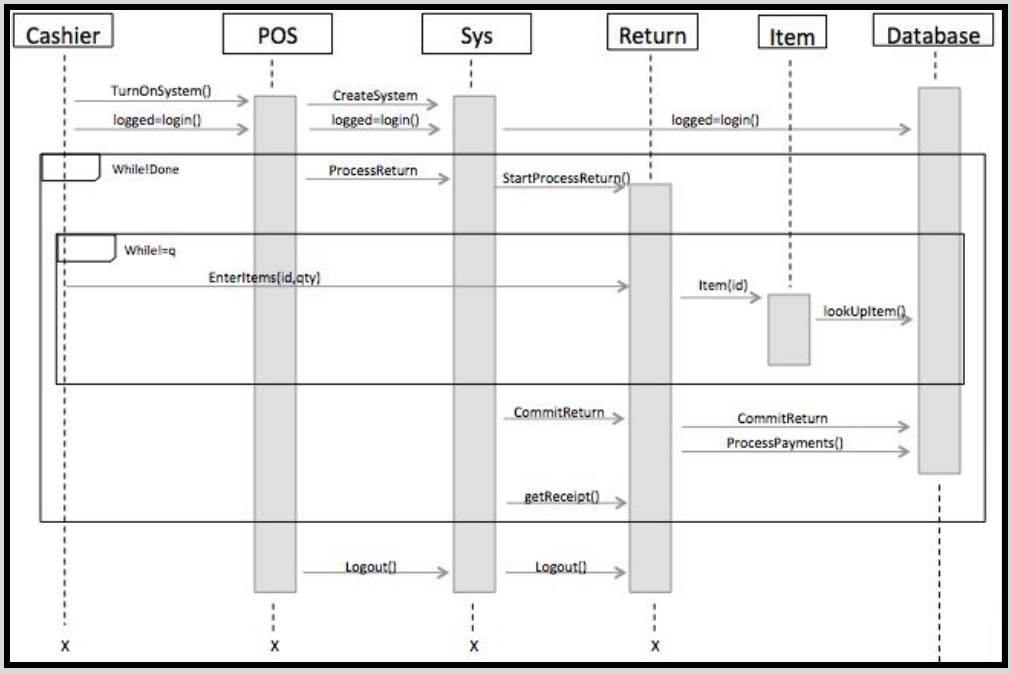
Adding a rental item is done inside the AddItem class. Once the OK button is pressed on the Add Item frame, the program will check to see if the rental checkbox on the selected. If it is selected, the Sale member function addItem() will add the item to the sale as a rental rather than a normal sale. AddItem() will add the item to a LinkedListOfItems instance that holds all sales, either normal items or rentals. At checkout, the database will be updated with the rentals. To return a rental on the date provided at checkout, the user will select the "Rental Return" button which will create NewReturn object. Once the user enters a rental ID (RID), the database will be updated.

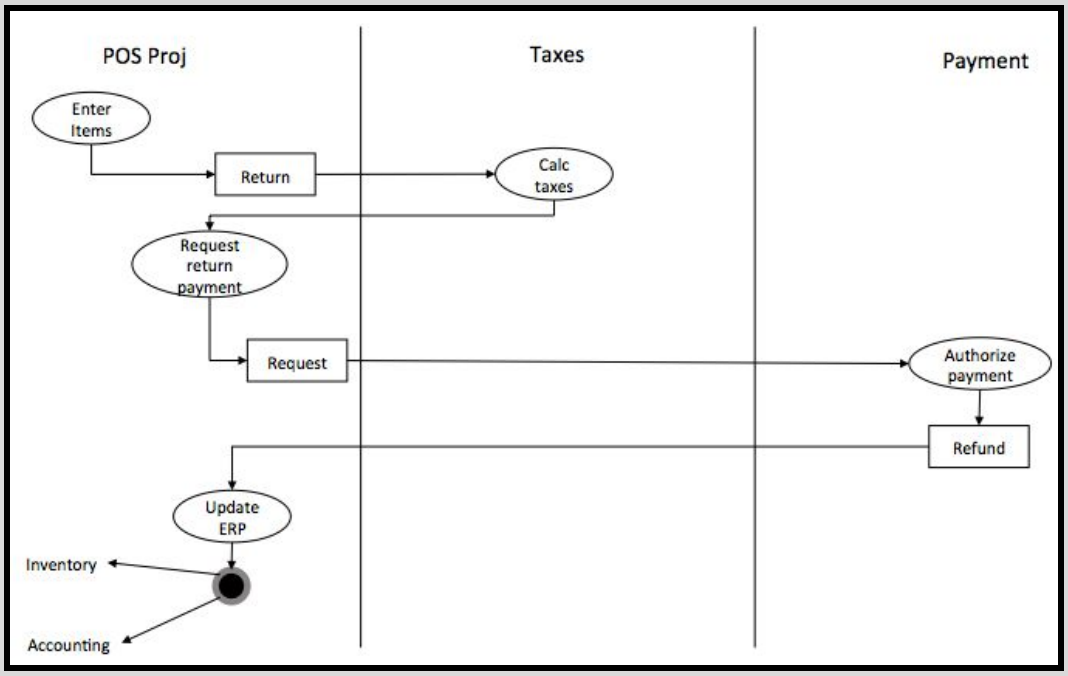




Section 6: Use Case 5: Process Return

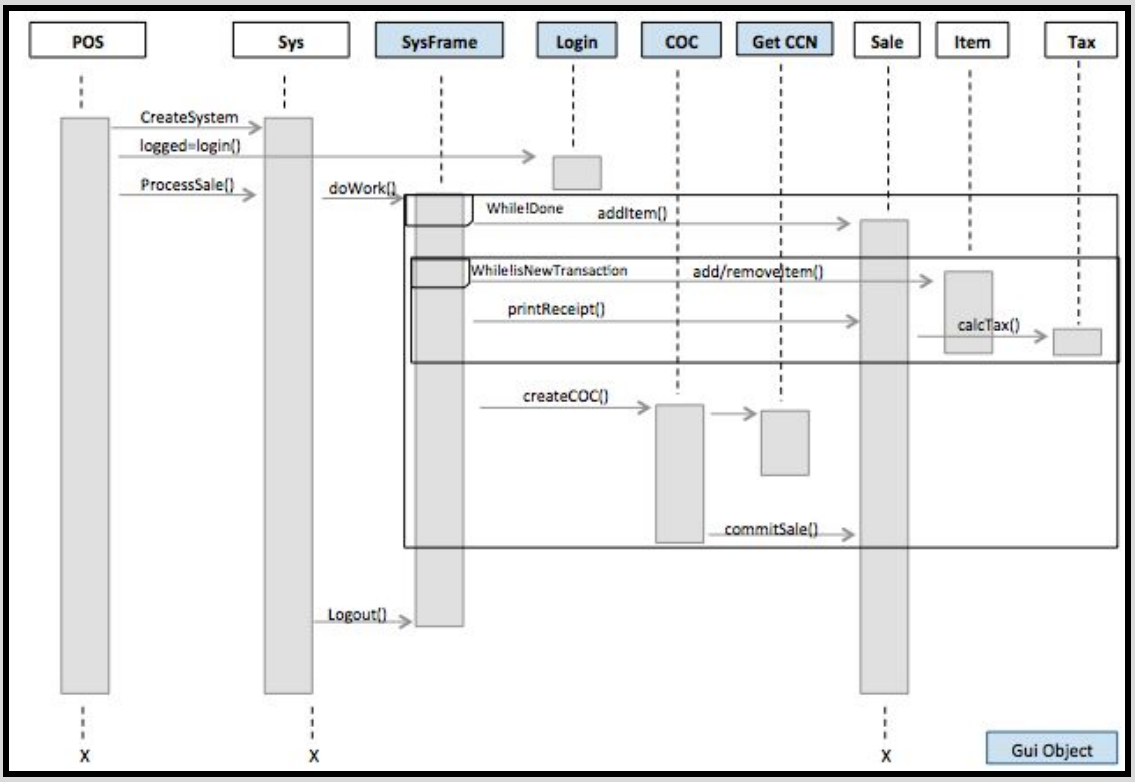
Completing a return is done through the Admin Frame once the user presses the "Void Transaction" button and a VoidFrame object is created. Once the user enters the transaction ID (TID), the system will detect if the transaction was previously completed with a credit card or with cash. If a credit card was used, a RefundCredit object will be created and if cash was used, a TenderCash object will be created. These are simple classes that complete the returns.





Section 7: Use Case 6: Graphical Interface

The user interface is developed using the java swing framework. This allows for a streamlined, visually appealing, and easy to use process for the customer. Most classes inside the PoS extend javax.swing, further emphasizing the importance of the GUI within the system. The diagram below visually represents the GUI interactions for the process sale use case.



Section 8: Use Case 7: Data Analytics

Admins are able to access the database to analyze sales data, inventory levels and revenue.Inside the database, the PoS data is rolled up into a multidimensional cube with two faces. One face contains CONDUCTS, TRANSACTION, and CONTAINS for sales data and the other has CONTAINS, INVENTORY and ITEM for inventory levels.

Section 9: Use Case 8: Multiple Users

The cloud based approach to our data storage, and our local handling of network faults provide a robust fault tolerant system that can be run on many machines simultaneously. This is enabled by the lack of temporary storage on connected scenarios. All terminals pull from the same cloud based store, making it easy to support parallel operations as all dependencies are fetched in real time from the database to prevent the violation of P/L and SPL constraints.

Glossary

No.	Term	Definition/Description
1	Actor	Participant in a use case. Actors can have three roles: primary (the focus of the use case), supporting (an actor which makes the use case possible), and offstage (important but not a direct player in the use case).
2	Administrator	An employee who can add users to and remove users from the POS system
3	COB	Close of Business
4	Commit	A database term representing the completion of a write to the database
5	Credentials	User ID/Username and password
6	Data	Files (or tuples) representing the relevant information to the store
7	Database	A tuple-based warehouse designed to hold data relevant to the POS system
8	Edit User	The POS system has a feature that allows a user to edit his/her information and credentials.
9	Fault	An error which causes system to crash or undergone some kind of paradigm shift
10	Fault Tolerant	Able to gracefully handle different errors
11	Inventory	A numeric representation of the number of items currently in stock
12	Network	A socket connected to the internet that allows the validation of payment information, and the caching of transactional data
13	New User Account	A new user account is created to give an employee access to the POS system's features. The account can be logged into with user credentials.

14	POS	Point of Sales
15	Power Down Option	Feature within the POS that allows a user to shutdown the system.
16	Recovered Data	The system saves data on a consistent basis in an attempt to eliminate work loss following a system fault. When the system becomes responsive again, it will try to recover data that may have been lost due to the fault.
17	Remove User	The POS system allows for its administrator to remove previously added user accounts from the system. This will prevent the employee connected with the deleted user account from logging in and using the system.
18	Restore to Previous State	After a system crash or fault, the POS system will attempt to restore recovered data and allow the user to resume work from the most recently saved state.
19	SKU	"Stockkeeping Unit" - functions as an item identifier for inventory control.
20	Technical Support	Any support available to employees or cashiers to assist in dealing with and troubleshooting POS system issues.
21	Transaction	An instance of exchange involving goods (to be sold/rented) and some form of payment (cash, debit/credit cards)
22	Transaction Log	A temporary file representing transactions which have yet to be written to the database
23	Unresponsive	The POS system does not respond to user input or commands
24	Write	The commitment of an update to the database (a change)
25	XML	Extensible Markup Language