Requirements Definition

0.1 Introduction

YardSale is an open source Point of Sale system that is being designed by A.S Logic Systems to revolutionize the way Point of Sale systems are implemented in today's market. Current implementations of Point of Sale systems are fraught with a number of issues; including often being extremely overpriced, difficult to administer, dated in their functionality, and lacking necessary operations. YardSale was designed to alleviate these problems and allow for extensibility as the market of retail sale expands in the future.

Being that YardSale is an open source project with little funding, the initial niche market will target small, locally owned retail stores. This type of market will allow for extensive on-site research, as many of these businesses should be willing to work with us to improve on the functionality of their existing POS system. In addition, because A.S. Logic has decided to take the open source route, the software itself will be freely available to all who wish to use it, the only expense that will come into play is if the company wishes to enlist our services in setup, troubleshooting, support, expansion, or customization.

0.2 Functional Requirements

Overview

This section entails all of the minimal requirements set by the AS Logic Systems development team for the OpenPOS hereafter referred to as YardSale.

YardSale at its most minimal functionality level should accomplish the following goals:

- Manage Inventory Data
- Manage Customer Data
- Manage Employee Data
- Perform Transactions

- Provide Logging Facilities
- Report on Data
- User Level Access Rights

Each of these tasks is described in depth in their corresponding sub sections. There is planned functionality above and beyond this basic featureset, although without these functions other functionality can not be added.

0.2.1 Database Management

The goal of the database management tasks is to add, modify, and delete information in the database. The database backend will run on a MySQL server and all relevant data will be stored in its own table or set of tables. More information on the database design can be found in the database design document.

Inventory Management

The inventory management system will have an interface for the stock employees to enter new shipments. It will also have an interface for the management employees to define new items for sale. The most common use of the inventory management system will be integrated into the checkout system. When customers are checking out, inventory will be populated to the checkout screen from the database, and then decremented on purchase from their quantity in the database.

Customer Management

The customer management system will work in a similar fashion to the inventory management system. It will have only one level of functionality though, the ability to define and update new customer data. There will also be a small interface with the database during the checkout of a customer which will allow for the selection of a customer from the customer table.

Employee Management

The employee management system here again will also function similarly to the previous two in that it has hooks into the database for employees. It will allow an authorized user the ability to add and modify employee information as well as disable employee accounts.

Among the more specific functionality of each of these sections would be to allow the user to retrieve any item stored in either inventory, customer, or employee tables given a search criteria.

0.2.2 Transactions

Transactions in YardSale are basically a 4 step process. The first step in the process is to select a customer to do business with or select a cash ”quick” customer. After the customer is selected their basic information will be displayed on the main checkout screen. The user will will then either select items from the inventory from a hierarchy, scan them in with the barcode scanner, or manually type in their information in lookup fields to checkout a customer. A running total will be kept as well as tax calculations for the sale. After all items are accounted for a checkout screen will be called. This will allow the user to select the customer’s preferred method of payment and also provide the user with the ability to calculate the correct amount of change.

All transactions will be maintained with a table in the database that will link related elements of the transaction together for later reporting.

0.2.3 Logging

Many program logging features will be provided inherently due to the database backend. Among the logging facilities will be the following items:

- Employee Time Tracking
- Transaction Logging
- Shipment Logs
- SQL String Execution Logging

Employee time tracking will be logged based on when the login to their first client and when the logout of their last client. An entry in the Login table will track this for each instance of a session. The transaction logging as was discussed earlier will be maintained in a table of its own tracking related items, customers, and employees with transactions. The shipment logs will be an offshoot of a transaction in that some will be shipped. If so a log of the shipping information will be available as it relates to the customer and the carrier. As a debugging feature a SQL String Execution Log facility will also be included. For every SQL string used on the database, there will be a stored procedure that adds it to the log table for later searching and debug use. This feature will likely be disabled in the final deliverable.

0.2.4 Reporting

There will be a wide variety of reporting features available also as a result of the database backend. The following basic reporting functionality will be available in the first iteration of YardSale.

- Payroll for Employees
- Top Sales for Inventory Items

- Top Sales for Employees
- Top Sales for Customer
- Revenue Reporting given any time frame
- Hourly Employee Log Reporting

All of the reports will be outputted to a pdf via LaTeXtype setting. This will provide ease of portability and a wide range of flexibility in reporting.

0.2.5 User Level Access Rights

YardSale has three different levels of user interactivity. There are Managers, Sales Associates, and Administrative users.

Sales Associate Functionality

Sales associates are likely going to be the primary users of the YardSale system. However they will have a limited set of functionality when they login. Their privileges will be limited to basic customer management and transaction processing. They have no real need for full inventory capabilities nor do they need the ability to manage or browse employee information.

When a sales associate logs into YardSale they will see the following options:

- Transaction Processing
- Customer Management

Manager Functionality

The managers primary function in YardSale is to make sure that all of the inventory items, customers, and employees are correctly maintained. They are also the only primary user on the system allowed to run reporting functions.

When a manager logs in to YardSale they will have a full set of functionality available to them from the main screen inclusive of the following:

- Inventory Management
- Transaction Processing
- Customer Management
- Employee Management
- Reporting

Administrative Functionality

The administrative user is basically a built in manager user. This user will have full rights to the system and the password will be able to unlock other accounts in the system that are locked due to loss of password. The administrative user will likely never be used in day to day use of YardSale, however it is provided as a safety to always allow the access of encrypted data in case of total loss of management capability.

0.3 Non Functional Requirements

The main non functional requirements for YardSale are its ability to run on any Operating System or Architecture.

0.4 External Dependencies and Interfaces

0.5 Preliminary Design

0.5.1 Major Modules

0.5.2 System Architecture

0.6 Preliminary Project Task Plan

0.6.1 Milestones

0.6.2 Team Member Roles and Responsibilities

Design

This section is still in the works

Implementation

0.6.3 Class Documentation

Please note the following section was auto-generated. We are working on tweaking this output a little, but currently it is sectioned wrong.

Test Plan