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.

2 Functional Requirements

2.1 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

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.

2.2 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.

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.

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.

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.

2.3 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.

2.4 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.

2.5 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.

3 Design

This section is still in the works

4 Implementation

4.1 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.

5 YardDatabase Class Reference

`#include <ys_database.h>`

Collaboration diagram for YardDatabase:

5.1 Detailed Description

This is the main database backend which does all translation from OO calls to SQL/ODBC.

YardInvType(p. 10)

Public Member Functions

YardDatabase (const wxString &dsn, const wxString &name, const wxString &pass)

bool **connect** ()

vector< **YardInvType** > **InvSearchKeyword** (const unsigned long &sku)

Find all inventory matches of keyword search.

vector< **YardInvType** > **InvGet** (unsigned int num, unsigned int offset)

Get a batch of inventory items.

5.2 Member Function Documentation

5.2.1 vector<YardInvType> YardDatabase::InvGet (unsigned int *num*, unsigned int *offset*)

Get a batch of inventory items.

num The number of items to get.

offset The item index to start at.

A std::vector of **YardInvType**(p. 10) objects

5.2.2 vector<YardInvType> YardDatabase::InvSearchKeyword (const unsigned long & *sku*)

Find all inventory matches of keyword search.

This could be dangerous, need to limit all returns to some set value (or configured value).

keyword A text string to search for.

A std::vector of **YardInvType**(p. 10) objects

The documentation for this class was generated from the following files:

ys_database.h

ys_database.cpp

[width=60pt]classYardDatabase$_{coll_{graph}}$

6 YardDBType Class Reference

`#include <ys_dbtype.h>`

Inheritance diagram for YardDBType:

6.1 Detailed Description

Abstract base class for datebase objects in **YardSale**(p. 16).

Jesse Lovelace

Public Member Functions

YardDBType (const **YardDBType** &obj)

The documentation for this class was generated from the following file:

ys_dbtype.h

[width=57pt]classYardDBType$_{inherit_{graph}}$

7 YardEmployee Class Reference

`#include <ys_employee.h>`

7.1 Detailed Description

YardEmployee is the employee managment screen for **YardSale**(p. 16).

Depending on access level, users may insert/modify employee information via this screen.

Jesse Lovelace

Public Member Functions

YardEmployee (wxWindow ∗parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wxRESIZE_BORDER)

The documentation for this class was generated from the following files:

ys_employee.h

ys_employee.cpp

8 YardInventory Class Reference

`#include <ys_inventory.h>`

8.1 Detailed Description

The inventory screen.

Jesse Lovelace

1.5

Public Member Functions

YardInventory (wxWindow ∗parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wxRESIZE_BORDER)

void **OnExitButton** (wxCommandEvent &event)

Exit button handler.

8.2 Member Function Documentation

8.2.1 void YardInventory::OnExitButton (wxCommandEvent & *event*)

Exit button handler.

event The event being passed in.

The documentation for this class was generated from the following files:

ys_inventory.h

ys_inventory.cpp

9 YardInvType Class Reference

`#include <ys_inv_type.h>`

Inheritance diagram for YardInvType: Collaboration diagram for YardInv-Type:

9.1 Detailed Description

The **YardSale**(p. 16) Inventory Type is a OO representation of the datebase inventory table.

Jesse Lovelace

1.3

YardDBType(p. 6)

Public Member Functions

YardInvType (const **YardInvType** &obj)

Copy constructor.

YardInvType & **operator=** (const **YardInvType** &obj)

wxString **GetBarCode** () const

wxString **GetDescription** () const

wxString **GetDepartment** () const

unsigned long **GetQuantOnHand** () const

unsigned long **GetQuantOnOrder** () const

unsigned long **GetReorderLevel** () const

wxString **GetItemType** () const

float **GetItemWeightLbs** () const

float **GetTaxType** () const

long int **GetVendorId** () const

float **GetRetailPrice** () const

float **GetWholesalePrice** () const

vector< BulkPricing > **GetBulkPricing** () const

bool **IsOverSized** () const

bool **MustShipFreight** () const

void **SetBarCode** (const wxString &str)

void **SetDescription** (const wxString &str)

void **SetDepartment** (const wxString &str)

void **SetQuantOnHand** (unsigned long num)

void **SetQuantOnOrder** (unsigned long num) void **SetReorderLevel** (unsigned long num)

void **SetItemType** (const wxString &str)

void **SetItemWeightLbs** (float num)

void **SetTaxType** (float num)

void **SetVentorId** (long int num)

void **SetRetailPrice** (float num)

void **SetWholesalePrice** (float num)

bool **AddBulkPrice** (const BulkPricing &price)

bool **RemoveBulkPrice** (unsigned int level)

void **SetOverSized** (bool cond)

void **SetShipFreight** (bool cond)

Friends

class **YardDatabase**

The documentation for this class was generated from the following files:

ys_inv_type.h

ys_inv_type.cpp

[width=57pt]classYardInvType$_{inherit_{graph}}$

[width=57pt]classYardInvType$_{coll_{graph}}$

10 YardLogin Class Reference

`#include <ys_login.h>`

10.1 Detailed Description

This is the customized login screen for **YardSale**(p. 16).

Jesse Lovelace

1.3

Public Member Functions

YardLogin (wxWindow ∗parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wx-STAY_ON_TOP|wxRESIZE_BORDER)

void **OnExitButton** (wxCommandEvent &event)

Exit button handler.

void **OnLogin** (wxCommandEvent &event)

10.2 Member Function Documentation

10.2.1 void YardLogin::OnExitButton (wxCommandEvent & *event*)

Exit button handler.

event The event being passed in.

The documentation for this class was generated from the following files:

ys_login.h

ys_login.cpp

11 YardMain Class Reference

`#include <ys_main.h>`

11.1 Detailed Description

YardMain is the main menu screen for **YardSale**(p. 16), it displays graphical buttons for accessing the inventory, employees, sales, etc.

Jesse Lovelace

1.6

Public Member Functions

YardMain (wxWindow ∗parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wxRESIZE_BORDER|wxFRAME_NO_TASKBAR)

Constructor.

void **OnLogout** (wxCommandEvent &event)

Logout button event handler.

void **OnMax** (wxCommandEvent &event)

Maximize handler.

void **OnInventory** (wxCommandEvent &event)

Inventory button handler.

void **OnSale** (wxCommandEvent &event)

Sales buttons.

void **OnEmployee** (wxCommandEvent &event)

Employee button handler.

11.2 Constructor & Destructor Documentation

11.2.1 YardMain::YardMain (wxWindow ∗ *parent*, wxWindowID *id*, const wxString & *title*, const wxPoint & *pos* = wxDefaultPosition, const wxSize & *size* = wxDefaultSize, long *style* = wxRESIZE_BORDER|wxFRAME_NO_TASKBAR)

Constructor.

Make these compiled into the binary.

The documentation for this class was generated from the following files:

ys_main.h

ys_main.cpp

12 YardSale Class Reference

`#include <yardsale.h>`

12.1 Detailed Description

This is the main application object.

Jesse Lovelace

1.8

Public Member Functions

virtual bool **OnInit** ()

This is the function were top level windows are created.

12.2 Member Function Documentation

12.2.1 bool YardSale::OnInit () `[virtual]`

This is the function were top level windows are created.

True if application initialized ok

The documentation for this class was generated from the following files:

yardsale.h

yardsale.cpp

13 YardSaleScreen Class Reference

`#include <ys_sale.h>`

13.1 Detailed Description

This is the main sale screen.

It contains the current transaction information.

Jesse Lovelace

1.3

Public Member Functions

YardSaleScreen (wxWindow *parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wxRESIZE_BORDER)

void **OnExitButton** (wxCommandEvent &event)

Exit button event handler.

The documentation for this class was generated from the following files:

ys_sale.h

ys_sale.cpp

14 YardSplash Class Reference

`#include <ys_splash.h>`

14.1 Detailed Description

Eye-candy splash screen.

Jesse Lovelace

1.6

Public Member Functions

YardSplash (wxWindow *parent, wxWindowID id, const wxString &title, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=wxSTAY_ON_TOP)

void **OnClick** (wxMouseEvent &event)

Mouse click event handler.

void **OnTimer** (wxTimerEvent &event)

Timer event handler.

void **Bump** (unsigned int amount=10)

Called to nudge progress bar over.

14.2 Member Function Documentation

14.2.1 void YardSplash::Bump (unsigned int *amount* = 10)

Called to nudge progress bar over.

amount Number of pixels to nudge

14.2.2 void YardSplash::OnClick (wxMouseEvent & *event*)

Mouse click event handler.

This handler is for testing only and will be removed

The documentation for this class was generated from the following files:

ys_splash.h

ys_splash.cpp

15 Test Plan