Campbell, Giarusso, Morisette, Menke

CS 2720 – Software enginnering  Spring, 2016; Dr. J. Phillip East

Point of sale system

Table of Contents

[System Documentation 2](#_Toc449011863)

[System-Level Overview 2](#_Toc449011864)

[Component Functionality and Interfaces 3](#_Toc449011865)

[Known Problems 3](#_Toc449011866)

[Database Design 3](#_Toc449011867)

[Configuration Settings 5](#_Toc449011868)

[Design Considerations 5](#_Toc449011869)

[Customer Report 5](#_Toc449011870)

[Overview of Deliverable 5](#_Toc449011871)

[Comparison to customer request 5](#_Toc449011872)

[Core Requirements 5](#_Toc449011873)

[Additional Requirements 6](#_Toc449011874)

[Guide to Setup 6](#_Toc449011875)

[User Accounts 6](#_Toc449011876)

[Supplier Information 6](#_Toc449011877)

[Product Information 7](#_Toc449011878)

[Inventory Input 7](#_Toc449011879)

[Guide to Use 7](#_Toc449011880)

[Users 7](#_Toc449011881)

[Manager 7](#_Toc449011882)

[Cashier 7](#_Toc449011883)

[Stocker 7](#_Toc449011884)

[Guide to Maintenance 8](#_Toc449011885)

[System Execution and Testing 8](#_Toc449011886)

[Web-Based 8](#_Toc449011887)

[Local Host 8](#_Toc449011888)

[Source Code Overview 8](#_Toc449011889)

# System Documentation

## System-Level Overview

The



*Terminology note: The following terms have a standardized meaning, which is held consistent throughout the document.*

* *Item – Refers to an individual item itself. A single box of matches.*
* *Product – Refers to a type of item. Matches.*
* *Transaction – A single transaction with a customer, which may include one or many items.*
* *Item\_Sold – Refers to an individual item that has been sold.*
* *Discount – A temporary change in price, commonly reflecting a special price of some kind.*

## Component Functionality and Interfaces

## Known Problems

## Database Design

The POS Database contains 7 tables for the various types of information, implemented through a SQL-Alchemy Wrapper with classes that inherit from the SQL-Alchemy Model class, all found in the file Models.py. Basic configuration settings direct SQL-Alchemy to the proper location and type for the production/local implementation. For more information on the proper configuration settings, see the configuration section.

|  |  |
| --- | --- |
| Data Types (Python) | |
| int | integer (primitive) |
| Float | floating point number |
| String-xx | string, *xx* specifies max number of characters |
| DATE | Instance of datetime.date (python module, see datetime documentation) |

All tables are user-accessible through the database dropdown menu to perform basic row additions, deletions, and edits. All fields in all tables are required (“nullable=false”). The complete framework is listed below. Each table column is listed along with its data type (listed *right)*. Fields in bold are user-entered, fields not in bold are automatically generated, most commonly from other data tables. Required columns from other data tables are underlined. Finally, columns’ foreign keys are listed in italics. These prohibit row deletion if other rows in other tables are dependent or row addition if the appropriate dependency does not already exist. References to other tables are written as

*other\_table.column\_in\_other\_table.* Capitalization is used for readability here; all source code variables and methods in the database are lowercase.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Items | | | | |
| ID (int)  *Primary Key* | **Product\_ID (int)**  ***Foreign Key: products.id*** | **Inventory\_Cost**  **(Float)** | Expiration\_Date (DATE)  from products.id | Author\_ID (int)  *Foreign Key: users.id* |

|  |  |  |  |
| --- | --- | --- | --- |
| Users | | | |
| ID (int)  *Primary Key* | **Name**  **(String-20)** | **Password**  **(String-400)** | **Permissions (int)** |
| Note: Permissions (int) is a categorical variable:  1 – Manager; 2 – Cashier; 3 - Stocker | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Products | | | | | | |
| ID (int)  *Primary Key* | **Name (String-20)** | **Supplier\_ID (int)**  ***Foreign Key: suppliers.id*** | Inventory\_Count (int)  from items | **Min\_Inventory (int)** | **Shelf\_Life (int)** | **Standard\_Price (Float)** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Items\_Sold | | | | | |
| ID (int)  *Primary Key* | **Item\_ID (int)**  **from items.id** | Product\_ID (int)  from products.id | **Price\_Sold (Float)** | Inventory\_Cost (Float)  from items.inventory\_cost | **Transaction\_ID (int)**  ***Foreign Key: transactions.id*** |

|  |  |  |
| --- | --- | --- |
| Suppliers | | |
| ID (int) *Primary Key* | **Name (String-20)** | **Email (String-40)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transactions | | | | |
| ID (int)  *Primary Key* | **Cust\_Name (String-20)** | **Cust\_Contact (String-40)** | **Payment\_Type (int)** | **Date (DATE)**  **(defaults to today)** |
| Note: Payment\_type (int) is a categorical variable:  1 – Cash; 2 – Credit; 3 – Debit; 4 – Check; 5 – Other | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Discounts | | | | |
| ID (int)  *Primary Key* | **Product\_ID (int)**  ***Foreign Key: products.id*** | **Start\_Date (DATE)** | **End\_Date (DATE)** | **Discount (Float)** |
| Note: Discount must be a number between 0 and 1 (inclusive), representing the “percent-off”. For example, Discount = 0.3 would signify 30% off. Entering 0 would not change the price; 1 would make the item free. | | | | |

### Configuration Settings

Explain SQL-Alchemy basics, where to find the config settings, how to switch between local/production settings.

## Design Considerations

# Customer Report

## Overview of Deliverable

The Point-Of-Sale System (POS) is a web-application that provides merchants with the ability to store and track inventory and sales in a robust and detailed custom database. Unlike most existing systems, POS stores individualized inventory information, eliminating complicated accounting issues in cost-of-sale calculations. It provides a detailed interface to stored data, helping to reduce regular technical maintenance requirements. Perhaps most importantly, it focuses around storing core data, allowing customers to easily add individualized functionality in future updates as required.

*Terminology note: The following terms have a standardized meaning, which is held consistent throughout the document.*

* *Item – Refers to an individual item itself. A single box of matches.*
* *Product – Refers to a type of item. Matches.*
* *Transaction – A single transaction with a customer, which may include one or many items.*
* *Item\_Sold – Refers to an individual item that has been sold.*
* *Discount – A temporary change in price, commonly reflecting a special price of some kind.*

## Comparison to customer request

POS meets and exceeds all customer requirements. Support for requirements not specifically addressed below can be found in referenced sections.

### Core Requirements

* Handle a sale, looking up product prices, adding items to a receipt, and updating inventory values (*see Guide to Use: Cashier*)
* Generate need report when inventory for a product reaches a preset value (*see Guide to Setup and Guide to Use: Manager*)
* Record audit trail data (*see Guide to Use: Manager*)
* Receive incoming products (*see Guide to Use: Stocker*)
* Add or remove products from the inventory (*see Guide to Setup*)
* Update inventory values (see *Guide to Use: Manager*)
* Prepare reports, including sales-audit and inventory-worth (see *Guide to Use: Manager*)
* Handle temporary price changes (discounts) over a specified period of time (see *Guide to Use: Manager*)

### Additional Requirements

* Log of transactions including: list of items sold, name of buyer, buyer’s contact information, type of payment (see *Guide to Use: Manger*)
* Revenue and profit tracking for previous day, week, and month (see *Guide to Use: Manager*)
* Minimum capacity for approximately 50 different products (see *Guide to Setup*)
* Web interface (see *Guide to Use*)

## Guide to Setup

*Congratulations on your new Point-Of-Sale System (POS)!*

By this point in the process, your system should be running on a web-server with both a web interface and a database, which you can verify by navigating to the URL provided by technical support when they installed your POS. Contact technical support if these steps are not completed. This guide will walk you through the following steps in system setup:

* User accounts
* Supplier information
* Product information
* Inventory input

### User Accounts

When you first log on to POS, you will have to use the default username/password given to you by technical support. If technical support did not give you a default, use the manufacturer standard default. Username: admin Password: admin

Because default information is not secure, you’ll first want to select Databases tab in the navbar and select Users from the dropdown. From here, you can create a new user by inputting the required information and selecting add. The first user you create should have manager permissions (by entering 1 in the permissions window). See *Guide to Use: Users* for more information regarding user permissions.

Next logout from the default account and login using your new username and password. Now you can delete the default user and add other users from the same window used to add the initial user.

### Supplier Information

While logged in to an account with manager permissions, select Databases in the navbar and Suppliers from the dropdown. Here, you’ll need to input a name and contact for suppliers of inventory you currently have or expect to input in the near future. An example entry may look like this: “Name: Acme Parts Co. Contact: suzyQ@acme.co”

*Note: Contact information is saved as a string of plain text, so you can enter up to 40 characters of email, phone, or other information you would like to store.*

### Product Information

While logged in to an account with manager permissions, select Databases in the navbar and Products from the dropdown. Here, you’ll need to input critical information for all products in your inventory (remember these are types, like matches, not individual items, like a box of matches). For each product type, enter the following information:

* Name: The product name, in 20 characters or less
* Supplier\_ID: The ID number of that product’s supplier, taken from the just-completed suppliers table
* Min\_Inventory: A whole number of how many of this item you would like to have in stock at all times
* Shelf\_Life: A whole number of days indicating how many days an item lasts on the shelf before it is expired and must be thrown out
* Standard\_Price: A decimal number indicating the standard price you intend to charge for the product, $1.11

### Inventory Input

While logged in to an account with manager or stocker permissions, select Inventory in the navbar. Here, you’ll need to input your current inventory (you’ll also use this window to add new inventory in the future). For each item (remember, an item is an individual item such as a box of matches), enter the following information:

* Product\_ID: The ID number of that item’s type of product, taken from the just-completed products table
* Inventory\_Cost: A decimal number indicating the price paid for the item from the supplier, used for accounting/audit purposes.

After inventory input is complete, your POS is ready for use! Read on to learn how to handle day to day operations with you POS.

## Guide to Use

### Users

### Manager

### Cashier

### Stocker

## Guide to Maintenance

Regular day-to-day operations should center around the Cashier, Inventory, Reports, and Database: Discounts windows. However, periodically, you may need to update other tables or correct erroneous entries. In general, reference the *Guide to Setup* for these revisions, employing the edit and delete functions of these windows in addition to the add function.

If user-maintenance causes errors or impairs functionality, reference the *System Documentation: Database Design* section for more information. If problems persist, contact technical support.

# System Execution and Testing

For instructor evaluation and grading, we have deployed a production version of the Point-of-Sale System (POS) to <http://sailingsales.pythonanywhere.com/>. A description of this is under the Web-Based section below. Additionally, the more technically minded user may wish to experiment with deploying their own version of the system to a local host, which is detailed under the Local Host section. Such experimentation is for those familiar with Python source code and Flask-based applications only, as it is very technical in nature.

## Web-Based

To demo POS for day-to-day operations, we have deployed a production version to <http://sailingsales.pythonanywhere.com/>. This simulates a client who has already completed the *Guide to Setup*. For evaluation and grading, we recommend re-following the *Guide to Setup* to observe adding and editing to different tables. Next, we recommend following the *Guide to Use* to observe the day-to-day functionality. Following these guides will cover all customer design requirements and fully exercise the system.

## Local Host

Alternatively, POS may be deployed to a local host by the technically inclined user. However, we must emphasize that this is a complicated process recommended for individuals familiar with Python source code and Flask-based web applications. While POS is generally robust and resilient, it was designed for deployment to a web-server by technical support. We make no guarantees regarding the functionality of local host deployments of POS.

To deploy POS locally, first download all necessary source files from <https://github.com/gioGats/CS2720-SE/archive/master.zip> and unzip the file.

Next, prepare the local environment. POS requires Python 3.5 or higher and a list of additional modules. Ensure first that Python 3.5 is installed and functional, and then that each additional module listed in master/Flask-App/requirements.txt is installed and functional.

For those users new to Python, we recommend using the Pip module to install further modules. Most newer installs come with Pip by default, but the use of Pip from a command-line or terminal interface is operating system dependent.

* A Pip download is available at: https://pypi.python.org/pypi/pip
* Pip documentation is available at: https://pip.pypa.io/en/stable/

Once Pip is installed, enter the following command in the command-line/terminal while in the master/Flask-App directory

pip install -r requirements.txt

After all additional modules have been installed, you must initialize databases by running the python file master/Flask-App/db\_create.py. This may be done through command-line/terminal input, the shell provided with Python, or other development environment.

Finally, run the python file master/Flask-app/app.py. This will deploy POS to the local host. Enter the IP address displayed in the python shell into a web browser (usually 127.0.0.1:5000). This will be a fully functioning local deployment of POS, with default “simulated” database entries as listed in master/Flask-App/db\_create.py

# Source Code Overview

The source code for the Point-of-Sale System (POS) is available for perusal and printing through the zip-file, at <https://github.com/gioGats/CS2720-SE/archive/master.zip>

Directory Structure (with file contents)