**To: P. Lin, G. Steffen**

**From: Jeremiah Bauer**

**Date: 2/24/2014**

**Re: Web Based Point of Sale System Design Project Report**

**Report Outline**

* Report Summary
* Problems and Corrections
* Status of the System
* Timeline for Completion
* Programming and Development Environment
* Hardware Information
* Project Costs
* Supporting Information
	+ Top Level System Diagram
	+ Cashier Use Case
	+ Object Controller UML Diagram
	+ Object Model UML Diagram
	+ Find Item Operation Sequence
	+ Complete Order Sequence Diagram
	+ Database Schema Diagram
	+ Sales Page UI Design
	+ Completed Item Maintenance Page Example
	+ Configuration Page Example
	+ Gantt Chart

**Report Summary**

In this report you will find problems and corrections, status of the system, a description of the programming and development environment, hardware information, and project costs. The design and construction of the system is moving along smoothly if not a little behind schedule due to being employed full time. I have every confidence that I will be able to finish the system by the presentation deadline.

**Problems and Corrections**There has only been one major issue encountered during development. I am able to generate barcodes for items both inside and outside of the application. However, I am unable to get them to show up in the web browser, this is simply a case of me not reading and comprehending the documentation behind sending binary data to the browser with Ruby on Rails.

After discussing with Dan’s Pies it was determined that users do not need to log in. Therefore the “Create User Login Page” and “Create User Administration Pages” tasks where dropped. This simplified the database schema and simplified the “Sales Page”. This functionality could be added at a later date if requested.

**Status of the System**

The system is currently under active development. The development is progressing slightly behind schedule due to my full time employment and other course load as can be seen in the Gantt Chart in Table 7. I fully expect to catch up on development and be ahead of schedule by the end of spring break. The objects for the ORM model have been designed and can be seen in Figure 4. These objects allow the application to store data in the PostgreSQL database schema seen in Figure 7. The controller objects have also been created which provide URL access to the ORM objects. These objects can be seen in Figure 3.

The item maintenance page has been completed as seen in Figure 9. The only functionality missing is displaying the barcode to the user when the “Show Barcode” link is clicked, currently when this link is clicked an exception is thrown and displayed to the user. This exception is due to not understanding completely how to pass dynamically generated binary data to the web browser with Ruby on Rails. This functionality will be completed with a few more hours of development time.

The sales screen has been designed and is described in Figure 8 and the sequence of operations required to find an item and add it to the order has been developed. This sequence is described in Figure 5. After all the items have been added to the order the user will click the “Complete Order” button and the operations described in Figure 6 will be executed. This page will be the next page to be finished.

The store configuration page has been completed and can be viewed in Figure 10 below. This screen allows the business to change values that will be used throughout the application.

**Programming and Development Environment**

The programming language that is being used for server side development is Ruby. Ruby is an interpreted, dynamically typed, scripting language that combines the best from Python and Perl. A web framework called Ruby on Rails is being used to speed up development. The database being used to store information is called PostgreSQL an open source relational database. The webpages generated by Ruby on Rails will use HTML, CSS, and JavaScript. The webserver that will be used for deployment is the Apache Webserver with the Phusion Passenger (mod\_rails) plugin.

The development environment is hosted on a CentOS 6.5 virtual machine. CentOS is a Linux distribution based on Red Hat Enterprise Linux. The editor being used is vim with the rails plugin. All software is open source and free to distribute and use.

**Table 1 Summary of Software and Versions**

|  |  |  |
| --- | --- | --- |
| **Software Name** | **Major Version** | **Purpose** |
| Vim | 7.2 | Text Editor |
| Rails.vim | 5.0 | Vim rails integration |
| Ruby | 2.0 | Server Side Scripting |
| Ruby on Rails | 4.0 | Server Side Web Framework |
| PostgreSQL Database Server | 9.3 | Relational Database Server |
| Apache HTTPD Webserver | 2.4 | Web server |
| Phusion Passenger (mod\_rails) | 4.0 | HTTPD ruby plugin |
| CentOS | 6.5 | Host OS |

Ruby gems are third party software that provide extra functionality to the Ruby programming language. Below is a list of these gems that are required for the application to function. All of these gems are licensed as open source software and are free to distribute.

**Table 2 Summary of Gems and Versions**

|  |  |  |
| --- | --- | --- |
| **Gem Name** | **Version** | **Purpose** |
| Rails | 4.0.2 | Web Application Framework |
| sass-rails | 4.0.1 | Style sheet generator |
| uglifier | 1.3.0 | JavaScript compressor |
| coffee-rails | 4.0.0 | Coffeescript compiler (JavaScript minilanguage) |
| jquery-rails | 3.0.4 | Provides the jQuery Javascript framework to the application |
| turbolinks | 2.1.0 | Web link library |
| jbuilder | 1.5.3 | JSON parser |
| therubyracer | 0.12.0 | Server side JavaScript engine |
| Barby | 0.5.1 | Barcode generator |
| chunky\_png | 1.30 | PNG graphic library |
| sdoc | 0.3.20 | Generates documentation from rdoc comments |
| Pg | 0.17.1 | Provides connectivity to PostgreSQL |
| Postgres | 0.8.1 | Provides connectivity to PostreSQL |

**Hardware Information**

The client PC being used is a net-top computer that is powered by an Intel Atom D510 processor and 2 gigabytes of RAM. This is enough processing power to host the barcode scanner and receipt printer. The server is powered by an Intel Core 2 Duo processor and 8 gigabytes of RAM. The barcode scanner is a generic brand USB barcode scanner purchased online. The receipt printer is a USB Epson ReadyPrint T20 Direct Thermal Printer. This printer was chosen because a Linux driver is available from Epson.

**Table 3 Hardware Summary**

|  |  |
| --- | --- |
| **Hardware Name** | **Function** |
| Intel Atom D510 Net-top | Client PC |
| Intel Core 2 Duo Server | Webserver and Database Server |
| Generic USB Barcode Scanner | Scan product barcodes |
| Epson ReadyPrint T20 Direct Thermal Printer | Print receipts |

**Project Costs**

**Table 4 Project Material Costs**

|  |  |
| --- | --- |
| **Material Costs** |   |
| Description | Cost |
| Receipt Printer | $144.98 |
| Total | $144.98 |

All hardware has been ordered and will be received by 2/28/2014.

**Table 6 Labor Costs To Date**

|  |  |  |  |
| --- | --- | --- | --- |
| Labor Costs |   |   |  |
| Resource | Description  | Estimated Hours | Actual Hours |
| Jeremiah Bauer | Determine Final Requirements | 5 | 2 |
| Jeremiah Bauer | Document System Architecture | 5 | 2 |
| Jeremiah Bauer | Document Unit Tests | 5 | 2 |
| Jeremiah Bauer | Model Database | 5 | 6 |
| Jeremiah Bauer | ORM Development | 5 | 2 |
| Jeremiah Bauer | Design User Interface Theme | 3 | 3 |
| Jeremiah Bauer | Main Sales Page Development | 20 | 1 |
| Jeremiah Bauer | Product Administration Page Development | 5 | 3 |
| Jeremiah Bauer | Barcode Generation Development | 5 | 3 |
| Jeremiah Bauer | Barcode Reading Development | 5 | 1 |
| Jeremiah Bauer | Receipt Printing Development | 3 |  |
| Jeremiah Bauer | Final Code Development | 2 |  |
| Jeremiah Bauer | Unit Test Development | 10 |  |
| Jeremiah Bauer | Setup Test System | 10 |  |
| Jeremiah Bauer | Deploy Code to Test Server | 2 |  |
| Jeremiah Bauer | Preliminary Functional Testing | 8 |  |
| Jeremiah Bauer | Write Verification Testing Plans | 15 |  |
| Jeremiah Bauer | Verification Testing | 5 |  |
| Jeremiah Bauer | Install Required Server Software At Dan's Pies | 5 |  |
| Jeremiah Bauer | Setup New Users on System | 5 |  |
| Jeremiah Bauer | Train Users On New System | 10 |  |
| Jeremiah Bauer | Construct Power Point | 5 |  |
| Jeremiah Bauer | Build and Insert Diagrams | 1 |  |
| Jeremiah Bauer | Review Presentation Requirements | 1 |  |
| Jeremiah Bauer | Make Final Power Point | 1 |  |
| Jeremiah Bauer | Write rough draft | 10 |  |
| Jeremiah Bauer | Proof-read and edit rough draft | 1 |  |
| Jeremiah Bauer | Write final report | 1 |  |
| Dan Bauer | User Acceptance Testing | 10 |  |
|  | Total: | 168 | 25 |

**Supporting Information**

The information below is to further clarify the design and behavior of the system.



Figure 1. Top Level System Diagram of a LAN-based Point of Sales System

Figure 2. Cashier Use Case



Figure 3. Object Controller UML Diagram



Figure 4. Object Model UML Diagram



Figure 5. Find\_Item() Operation Sequence



Figure 6. Complete Order Sequence



Figure 7. Database Schema



Figure 8 Sales Page UI Design



Figure 9 Completed Item Maintenance Page



Figure 10 Configuration Page



Table 7 Gantt Chart