Web Based Point Of Sale System

Jeremiah Bauer
Student of Computer Engineering Technology

Gary Steffen Faculty Advisor
Paul Lin CPET 491 Professor

May 2, 2014

2

Topics

- Introduction
- Previous system
- System Design
- System Integration and Testing
- Project Schedule
- Labor and Monetary costs
- Risk Management
- Lessons Learned
- Demonstration

Introduction

3

- Sponsored By Dan's Pies in North Webster, IN
  - Opened in 1990
  - Small business opened it's first retail location in 2010
- Project was a success
- Completed on time and under budget
- Meets all requirements set forth in phase 1

Previous System

4

- Only process sales with 7 categories of items
  - Uses cookies to pass sales data
    - Sales data was larger than maximum size of cookie
- Web based
- MySQL Backend
- PHP server side Programming
- HTML
- JavaScript
- CSS

System Design

5

- Ruby On Rails
- Apache Webserver
- Phusion Passenger (mod_rails)
- PostgreSQL
- CentOS 6.5
- HTML5
- JavaScript (jQuery)
- CSS3 (Bootstrap)

Tasks in Scope

6

- Processing Sales
- Designing a database schema
- Reading and writing barcodes
- Printing receipts
- Sales reporting
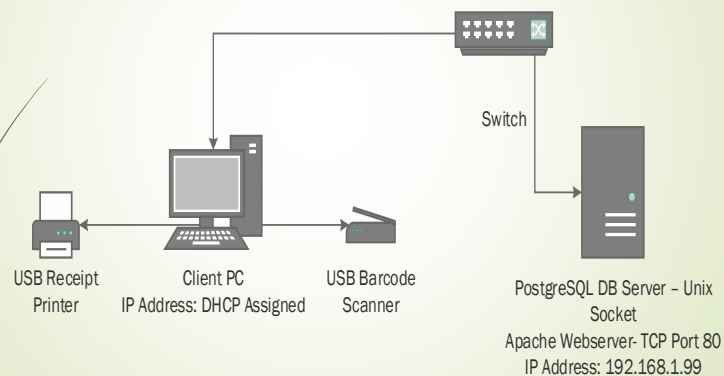- Create, update, and delete items

Out of Scope

7

- Credit card processing
- Inventory control system
- Invoice generation system
- Reservation system
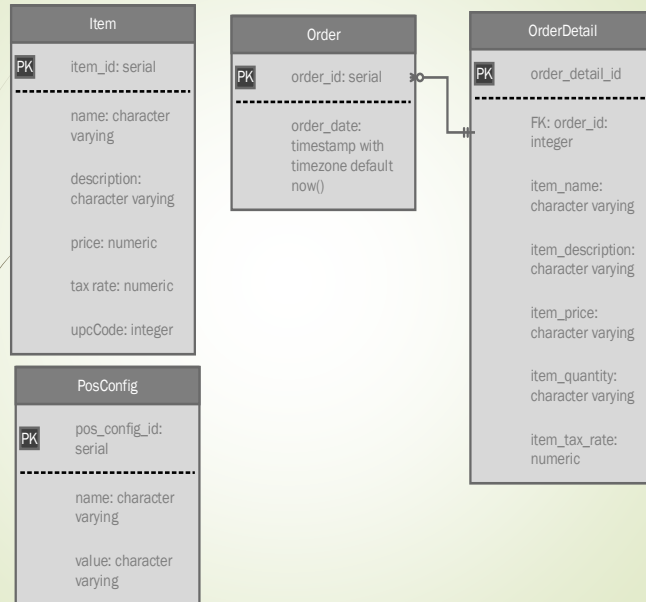
Top level System Diagram
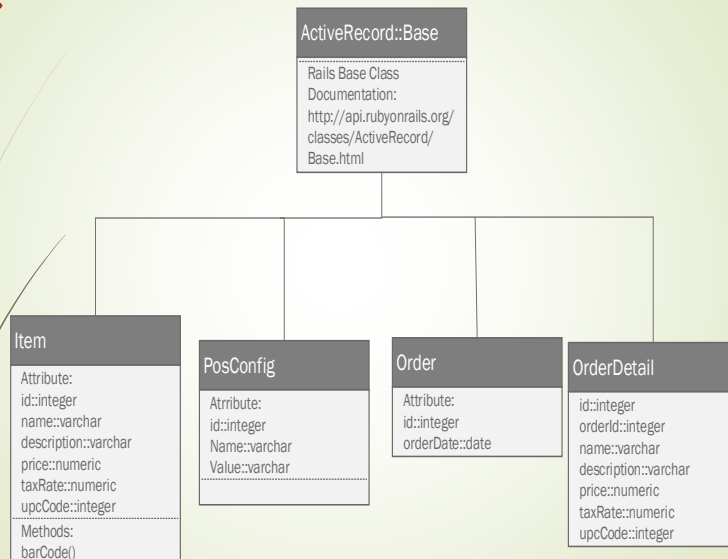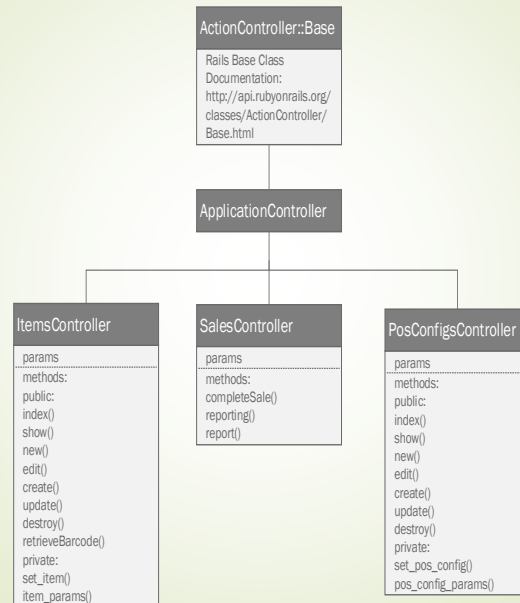
8

Point of Sale Top Level Diagram

Switch

USB Receipt
Printer

Client PC
IP Address: DHCP Assigned

USB Barcode
Scanner

PostgreSQL DB Server – Unix
Socket
Apache Webserver- TCP Port 80
IP Address: 192.168.1.99

Database Schema

Item

PK	item_id: serial
	name: character varying
	description: character varying
	price: numeric
	tax rate: numeric
	upcCode: integer

Order

PK	order_id: serial
	order_date: timestamp with timezone default now()

OrderDetail

PK	order_detail_id
	FK: order_id: integer
	item_name: character varying
	item_description: character varying
	item_price: character varying
	item_quantity: character varying
	item_tax_rate: numeric

PosConfig

PK	pos_config_id: serial
	name: character varying
	value: character varying

Rails Model Class diagram

ActiveRecord::Base

Rails Base Class Documentation:
http://api.rubyonrails.org/classes/ActiveRecord/Base.html

Item

Attribute:
id::integer
name::varchar
description::varchar
price::numeric
taxRate::numeric
upcCode::integer
Methods:
barCode()

PosConfig

Atribute:
id::integer
Name::varchar
Value::varchar

Order

Attribute:
id::integer
orderDate::date

OrderDetail

id::integer
orderId::integer
name::varchar
description::varchar
price::numeric
taxRate::numeric
upcCode::integer

Rails Controller Class Diagram

11

ActionController::Base

Rails Base Class
Documentation:
http://api.rubyonrails.org/
classes/ActionController/
Base.html

ApplicationController

ItemsController

params
methods:
public:
index()
show()
new()
edit()
create()
update()
destroy()
retrieveBarcode()
private:
set_item()
item_params()

SalesController

params
methods:
completeSale()
reporting()
report()

PosConfigsController

params
methods:
public:
index()
show()
new()
edit()
create()
update()
destroy()
private:
set_pos_config()
pos_config_params()

Rails Views

12

- Welcome Landing Page
- Application Layout
- Application Header Layout
- Item New/Edit/Show
- Configuration New/Edit/Show
- Sales Page
- Reporting Page

Find Item Sequence of Operations

13

Find Item Sequence

| USB Scanner | Host PC | Webserver | DB Server |

Item ID →

/items/searchById →

findItemById() →

←‑‑<item result set‑‑

←‑‑‑Item Details‑‑‑
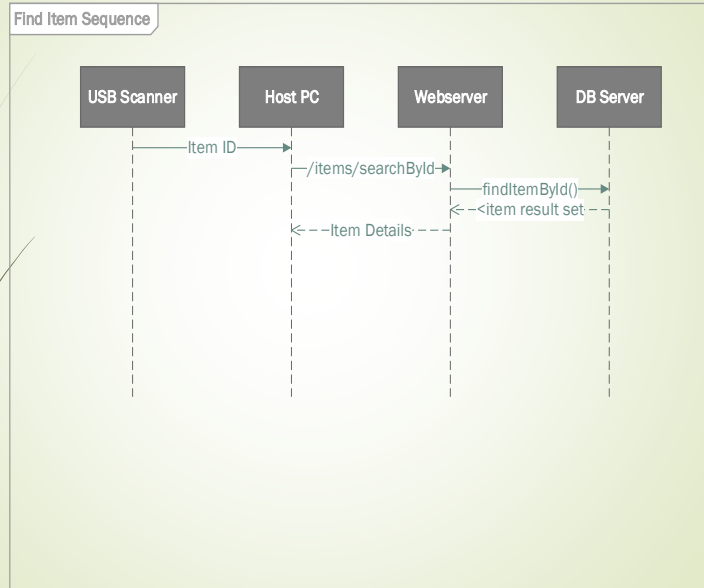
Sales Sequence of Operation

14

Complete Order Sequence

| Host PC | Webserver | DB Server |

User presses "Complete Order"
Sends order contents in json structure →

getNewOrderID() →

←‑‑‑return orderID‑‑‑

Inserts contents of
Order using new OrderID →

←‑‑returns Success or Failure‑‑

Returns success or failure
prints receipt if success ←‑‑

JavaScript Function List

15

- lookupItemById()
- insUpdProductRow(product)
- updateOrderTotals()
- applyPayment()
- runReport()
- completeSale()

CSS Classes

16

- logo
- barcode
- totalContainer
- itemContainer
- sales-container
- noItemsCell
- total
- receiptLogo
- receiptQuantity

17 | Most important requirement

- The system shall be able to process a sale with more than 7 items.

Dan's Pies Sales Reporting Administration ▼

Item Description	Quantity	Unit Price	Tax Rate	Item Total
Cherry Pie	3	7.50	7.0	24.08
Red Raspberry Pie	2	11.00	7.0	23.54
Apple Pie	2	8.50	7.0	18.19
Peanut Butter Pie	2	9.50	7.0	20.33
Tollhouse Pie	2	8.25	7.0	17.66
Strawberry Pie	1	8.50	7.0	9.10
Baked Strawberry Pie	1	8.50	7.0	9.10
Chocolate Pie	1	8.50	7.0	9.10
Coconut Pie	1	8.50	7.0	9.10

Sub Total: $131.00

Tax: $9.17

Order Total: $140.17

Amount Tendered:

Apply Payment

Amount Paid:

Change Due:

Complete Sale

18 | Secondary Requirements

- The system shall be able to generate bar codes by using a Ruby image processing library.
- The system shall be able to print a receipt.
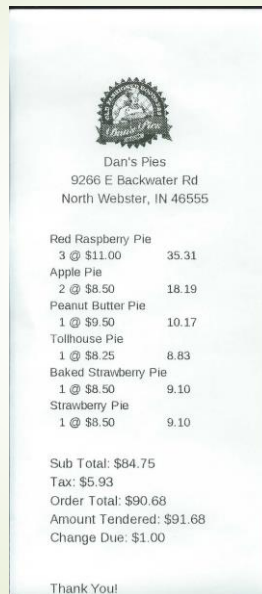- The system shall be able to generate a sales report.

Receipt Printing

19

- Uses a USB Epson ReadyPrint T20 Direct Thermal Printer
- Cost $157.24
- Uses JavaScript, HTML, and CSS to generate the receipt
- Created only after sale is successfully inserted into database

Sample Receipt

20

Barcode Generation

21

- Uses barcode encoding 39
- USB Barcode Scanner
- Generated using the "barby" barcode library
- Generated based on the items id in the items table

```ruby
def generateBarcode()

    barcode = Barby::Code39.new(self.id.to_s)
    fileName = "public/" + self.id.to_s + ".png"

    File.open( fileName , 'w' ){|f|

       f.write barcode.to_png

    }

    return fileName

end
```

Sample Barcode

22

Dan's Pies   Sales   Reporting   Administration ▾

Name: Cherry Pie
Description: Pie
Quantity:
Price: $7.50
Taxrate: 7.0%
Upccode:
Bar Code: ‖‖‖‖‖ ‖‖ ‖‖‖
Edit | Back

Project Schedule

23

- Required approximately 16 weeks of effort
- Most tasks were completed on time
- Receipt printing and main sales page were minor road blocks

Labor Costs

24

- 175 Estimated hours
- 107 actual hours

Monetary costs

25

- $200.00 Estimated cost
- $157.24 Actual cost

Risk Management

26

- Highest risk identified in Phase 1 was that the system would not be able to process 7 categories of items.
  - Was not encountered
- Project schedule was second highest risk
  - Not encountered project was completed on time and under budget
- Not being able to create barcodes
  - Encountered and mitigated by avoided by reading documentation

27 Lessons Learned

- Learned a new web framework (Ruby on Rails)
- Learned a new CSS framework (Bootstrap)
- JavaScript floating point calculations issues

28 Conclusion

- Project was a success
- Met all requirements set forth in Phase 1

29

Demonstration