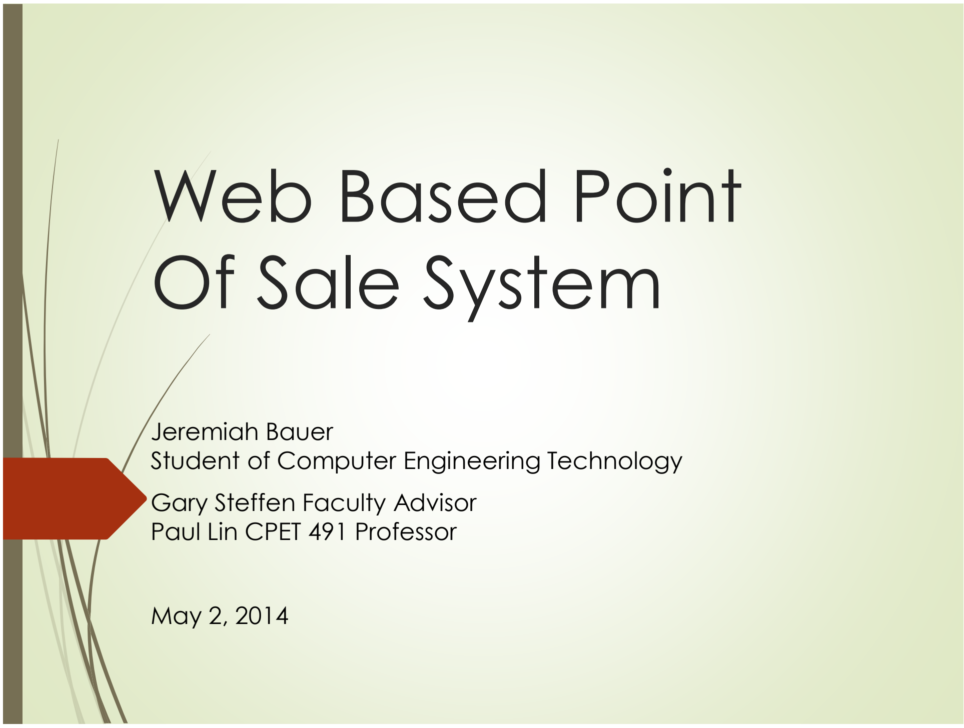
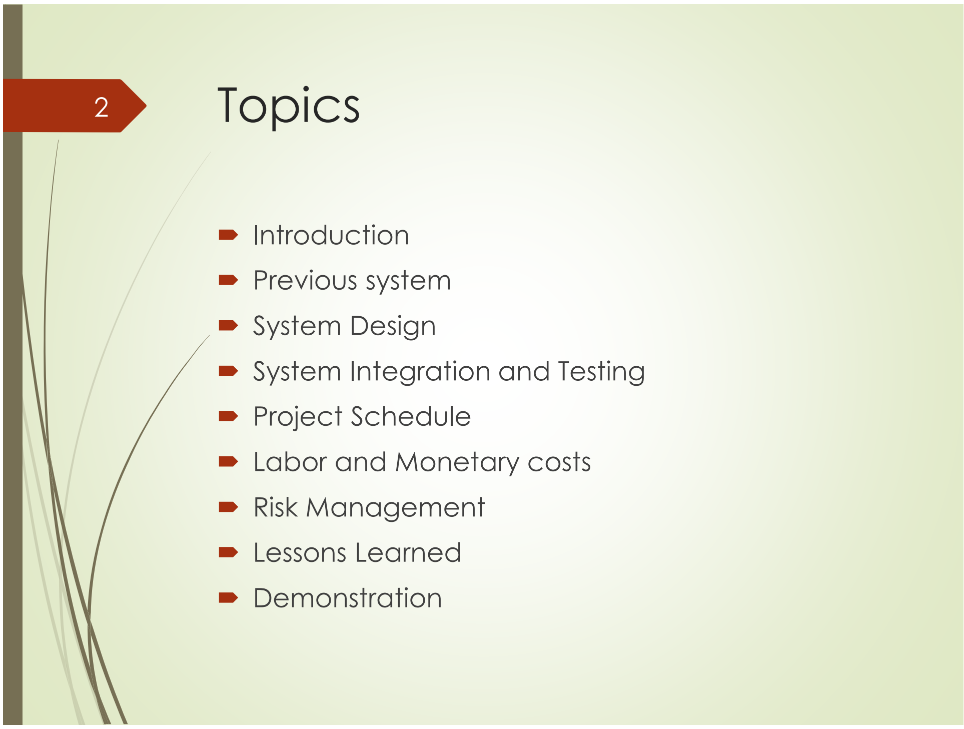


Web Based Point Of Sale System

Jeremiah Bauer
Student of Computer Engineering Technology

Gary Steffen Faculty Advisor
Paul Lin CPET 491 Professor

May 2, 2014



2

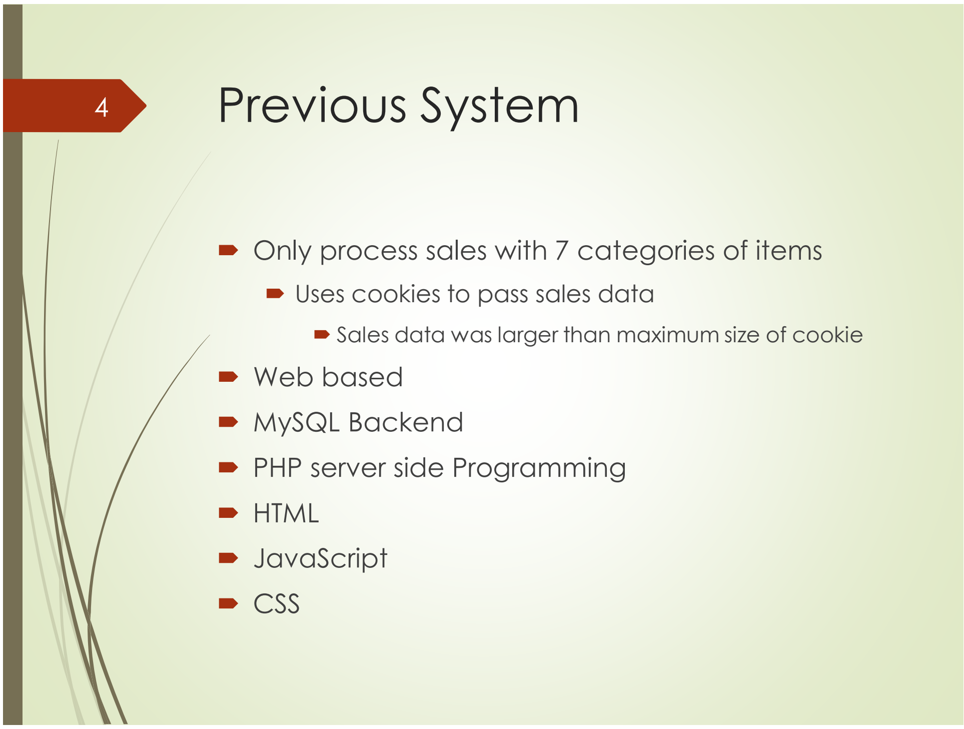
Topics

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

3

Introduction

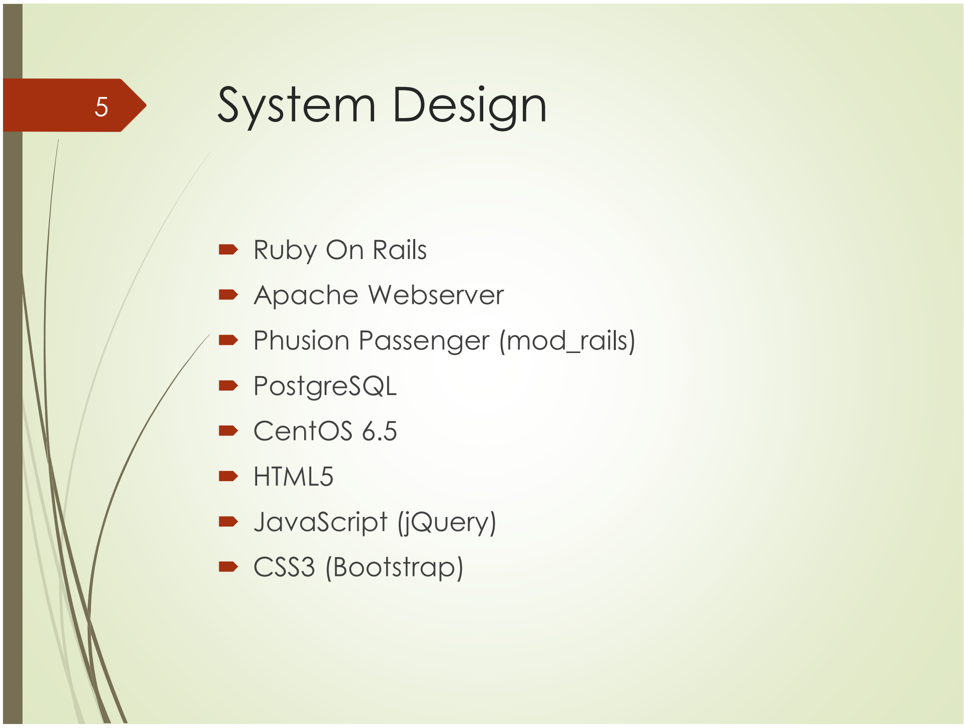
- Sponsored By Dan’s Pies in North Webster, IN
  - Opened in 1990
  - Small business opened it’s first retail location in 2010
- Project was a success
- Completed on time and under budget
- Meets all requirements set forth in phase 1



4

Previous System

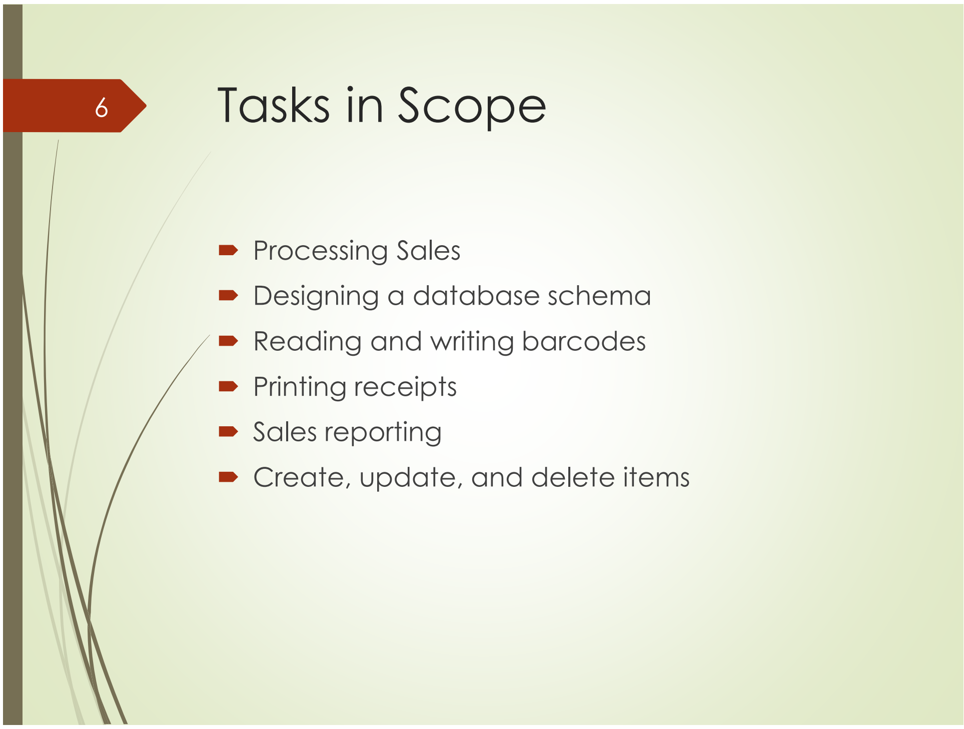
- Only process sales with 7 categories of items
  - Uses cookies to pass sales data
    - Sales data was larger than maximum size of cookie
- Web based
- MySQL Backend
- PHP server side Programming
- HTML
- JavaScript
- CSS



5

System Design

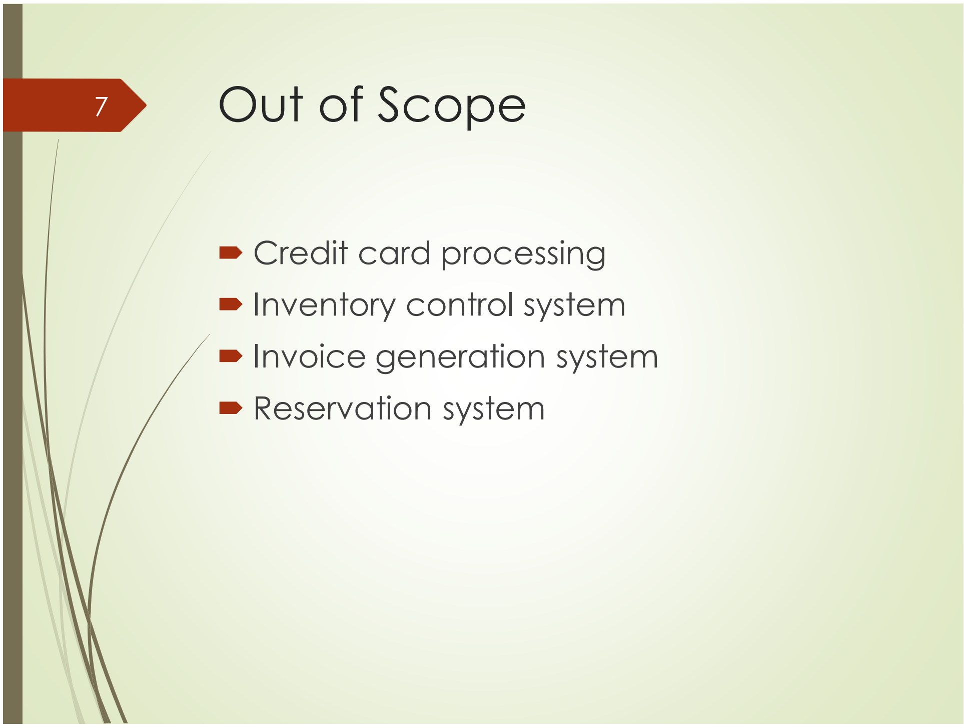
- Ruby On Rails
- Apache Webserver
- Phusion Passenger (mod_rails)
- PostgreSQL
- CentOS 6.5
- HTML5
- JavaScript (jQuery)
- CSS3 (Bootstrap)



6

Tasks in Scope

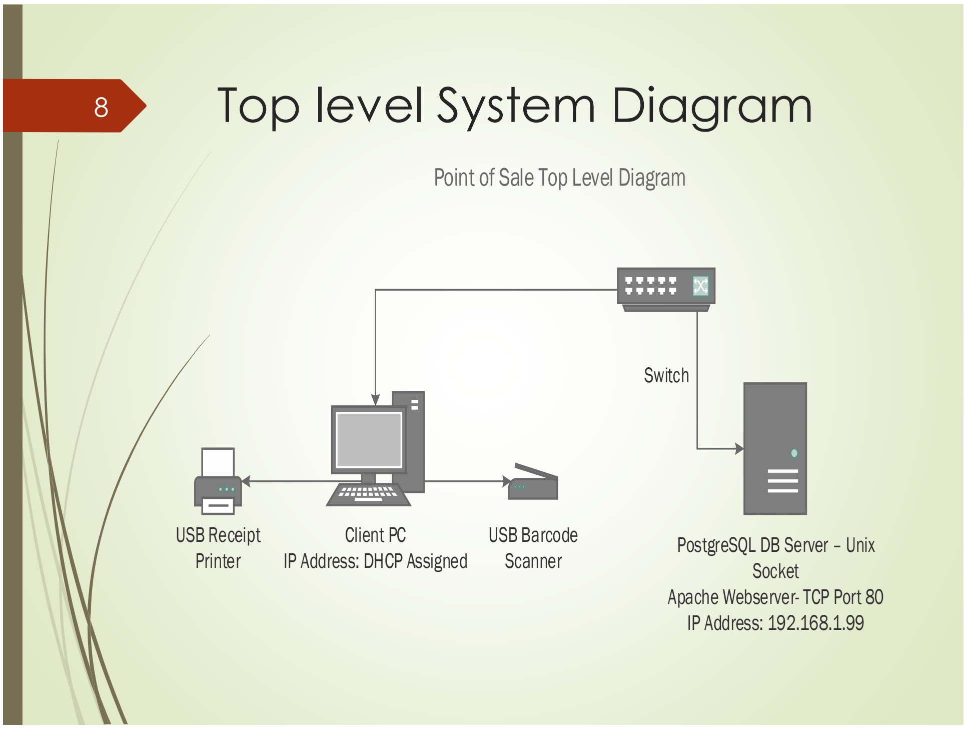
- Processing Sales
- Designing a database schema
- Reading and writing barcodes
- Printing receipts
- Sales reporting
- Create, update, and delete items



7

Out of Scope

- Credit card processing
- Inventory control system
- Invoice generation system
- Reservation system



8

Top level System Diagram

Point of Sale Top Level Diagram

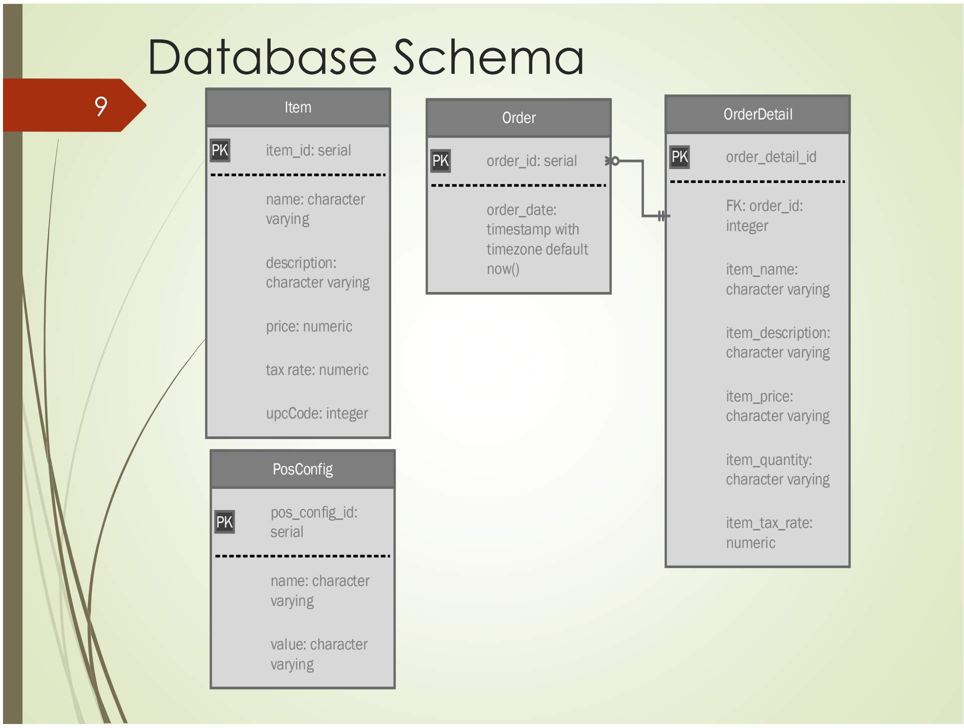
Switch

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



9

Database Schema

Item

PK item_id: serial

name: character varying

description: character varying

price: numeric

tax rate: numeric

upcCode: integer

PosConfig

PK pos_config_id: serial

name: character varying

value: character varying

Order

PK order_id: serial

order_date: timestamp with timezone default now()

OrderDetail

PK order_detail_id

FK: order_id: integer

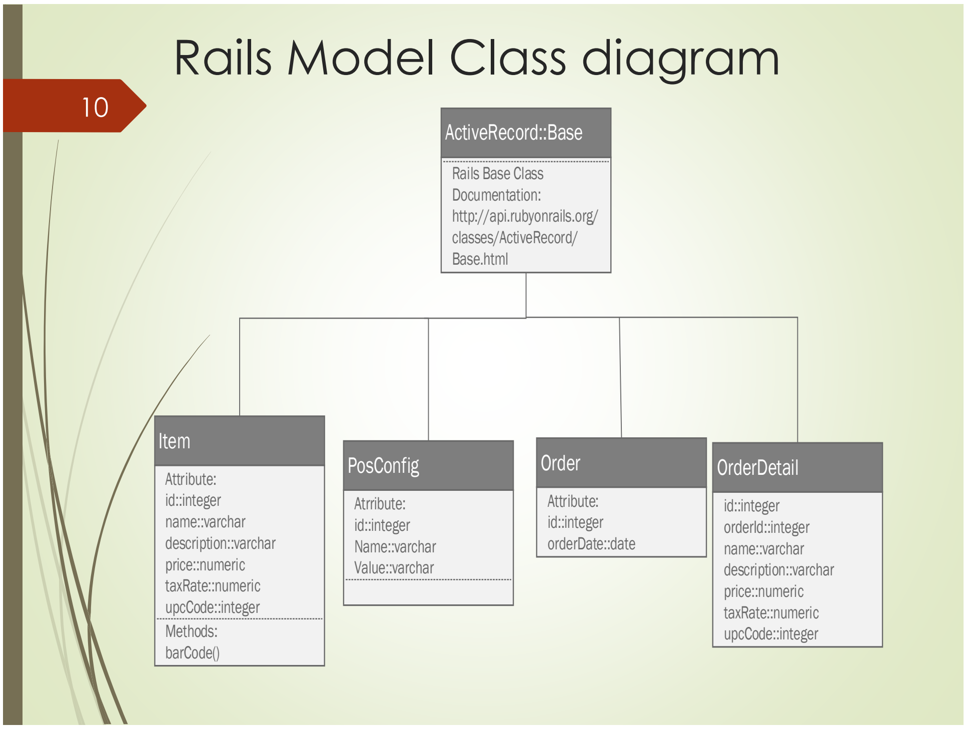
item_name: character varying

item_description: character varying

item_price: character varying

item_quantity: character varying

item_tax_rate: numeric



10

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

Atrribute:
id::integer
Name::varchar
Value::varchar

Order

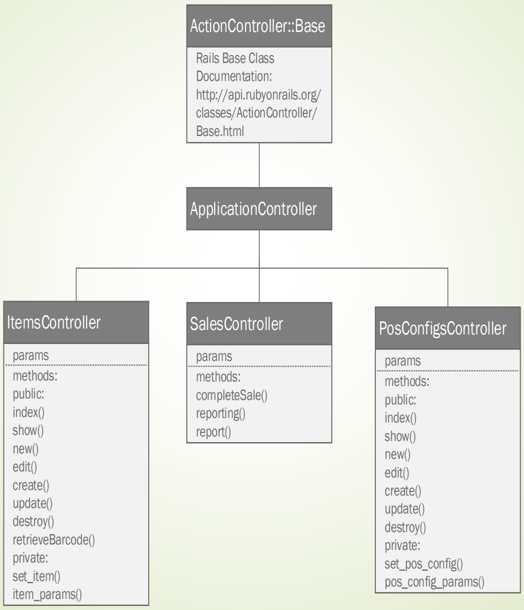
Attribute:
id::integer
orderDate::date

OrderDetail

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

11

Rails Controller Class Diagram



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

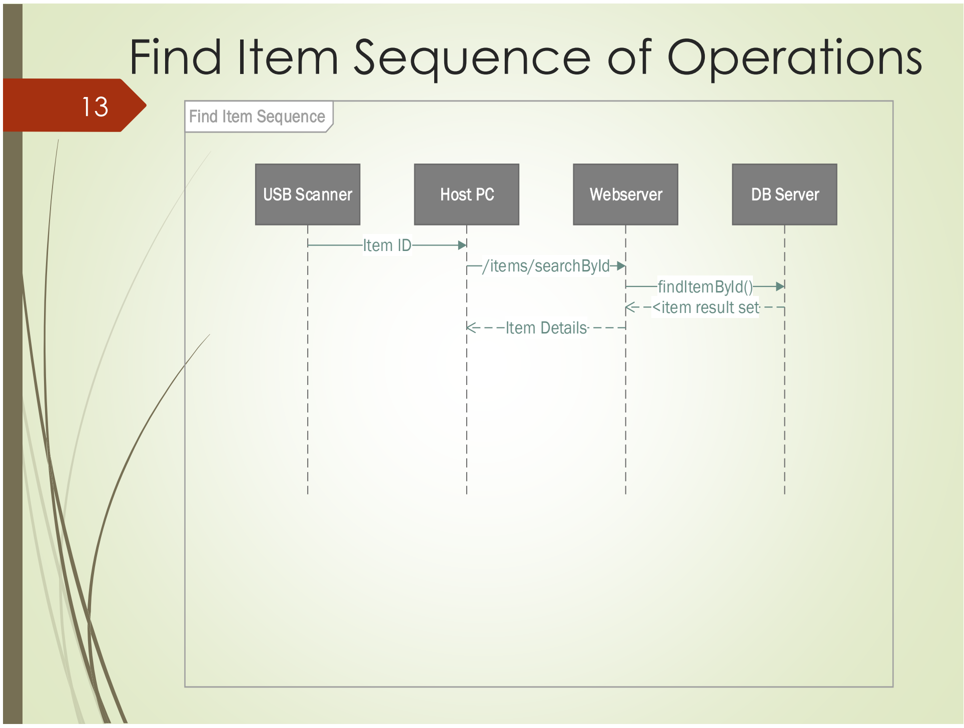
params

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

12

Rails Views

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



13

Find Item Sequence of Operations

Find Item Sequence

USB Scanner

Host PC

Webserver

DB Server

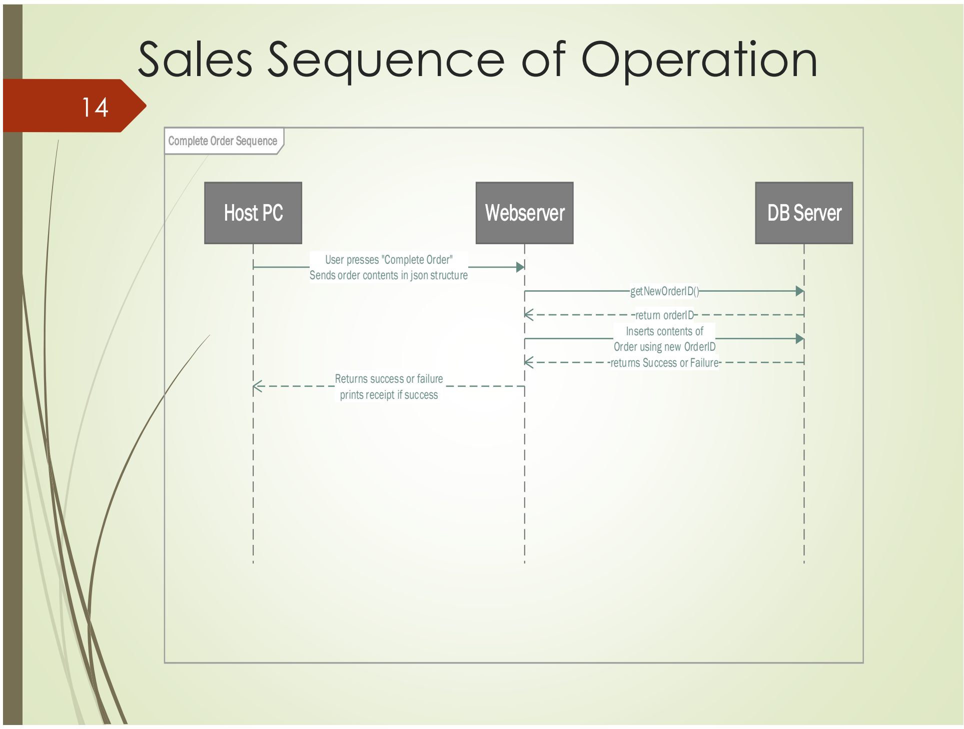
Item ID

/items/searchById

findItemById()

<item result set

Item Details



14

Sales Sequence of Operation

Complete Order Sequence

Host PC

Webserver

DB Server

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

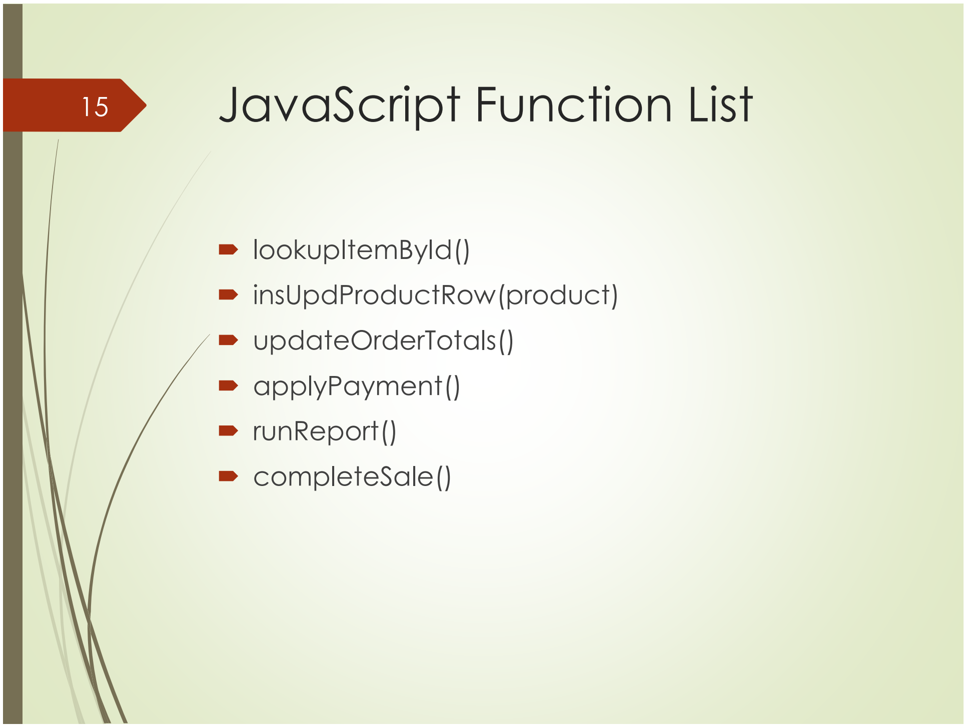
getNewOrderID()

return orderID

Inserts contents of
Order using new OrderID

returns Success or Failure

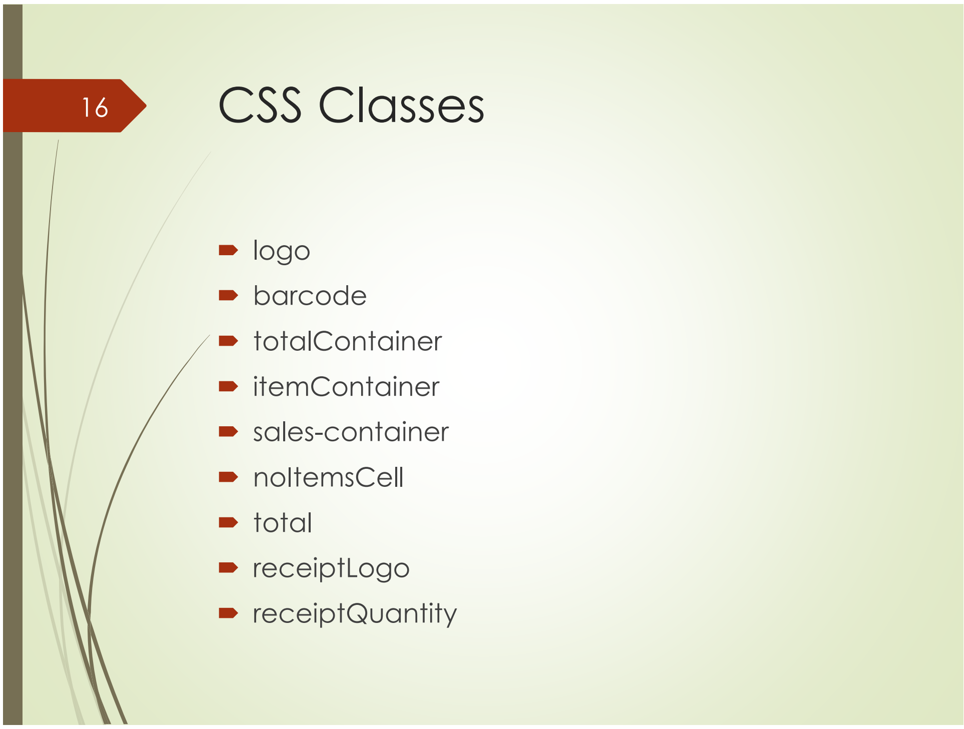
Returns success or failure
prints receipt if success



15

JavaScript Function List

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



16

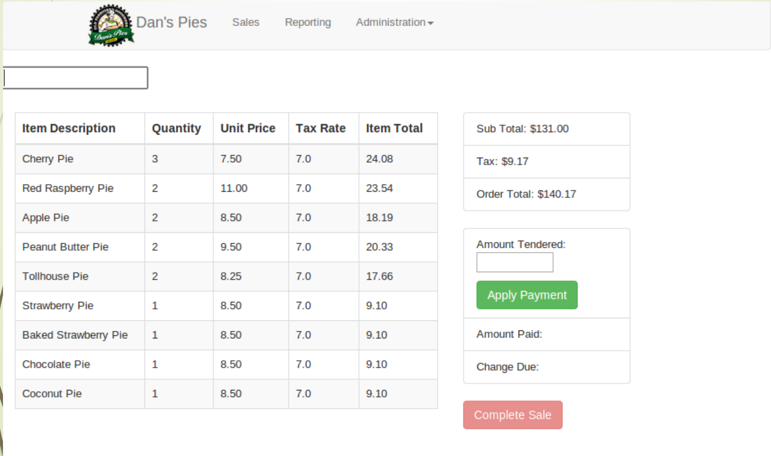
CSS Classes

- 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.



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:

Amount Paid:

Change Due:

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.

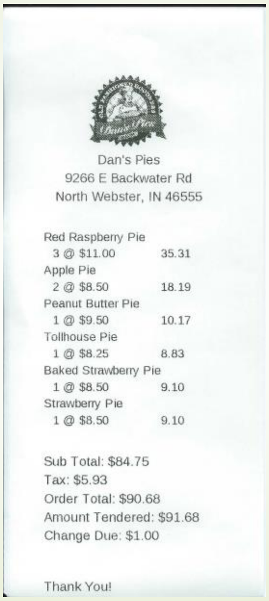
19

Receipt Printing

- 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

20

Sample Receipt



```
          Dan's Pies
     9266 E Backwater Rd
   North Webster, IN 46555

Red Raspberry Pie
  3 @ $11.00          35.31
Apple Pie
  2 @ $8.50           18.19
Peanut Butter Pie
  1 @ $9.50           10.17
Tollhouse Pie
  1 @ $8.25           8.83
Baked Strawberry Pie
  1 @ $8.50           9.10
Strawberry Pie
  1 @ $8.50           9.10

Sub Total: $84.75
Tax: $5.93
Order Total: $90.68
Amount Tendered: $91.68
Change Due: $1.00

Thank You!
```

21

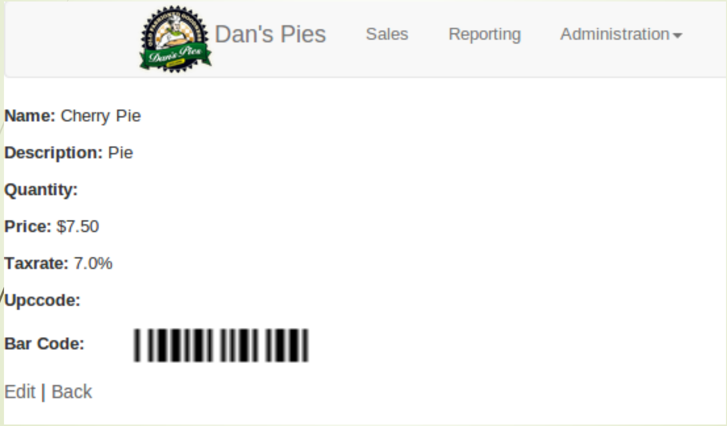
Barcode Generation

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

```
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
```

22

Sample Barcode



Name: Cherry Pie

Description: Pie

Quantity:

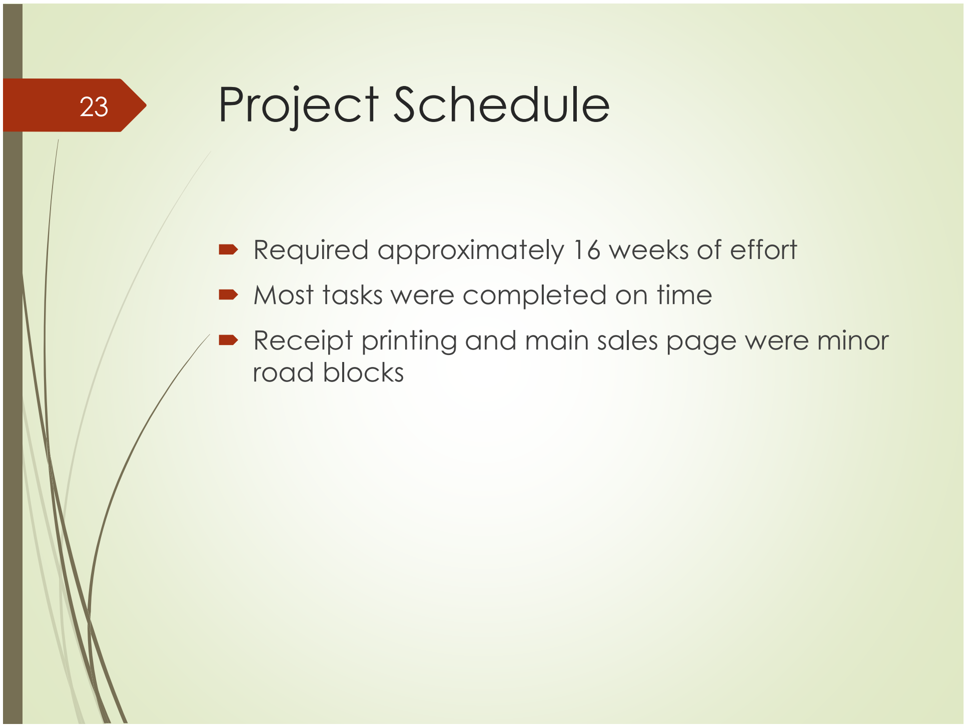
Price: $7.50

Taxrate: 7.0%

Upccode:

Bar Code:

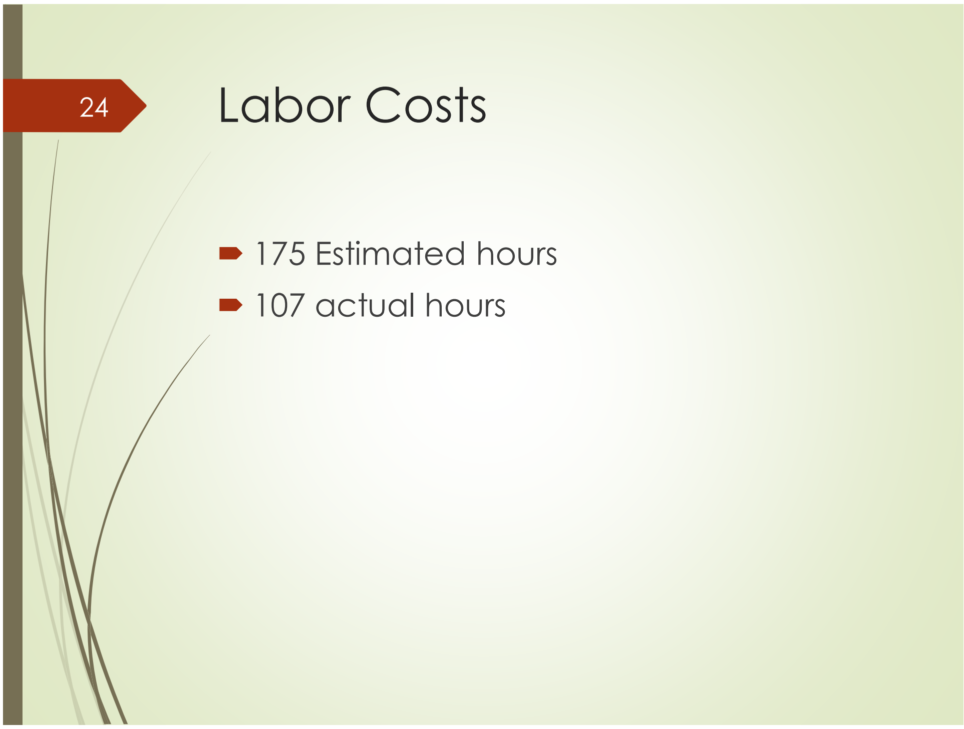
Edit | Back



23

Project Schedule

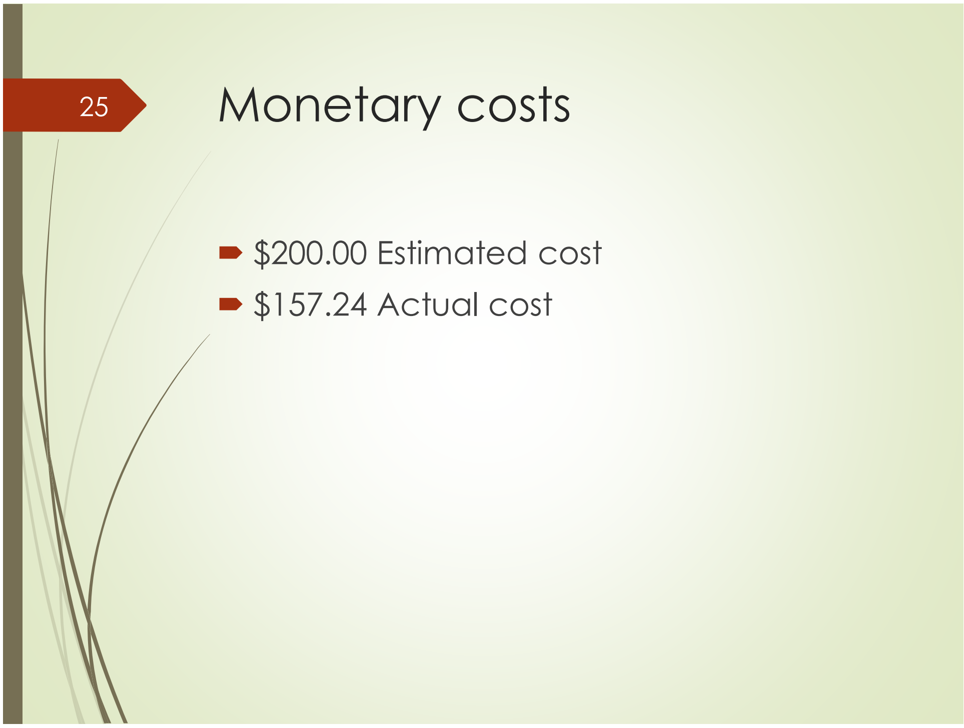
- Required approximately 16 weeks of effort
- Most tasks were completed on time
- Receipt printing and main sales page were minor road blocks



24

Labor Costs

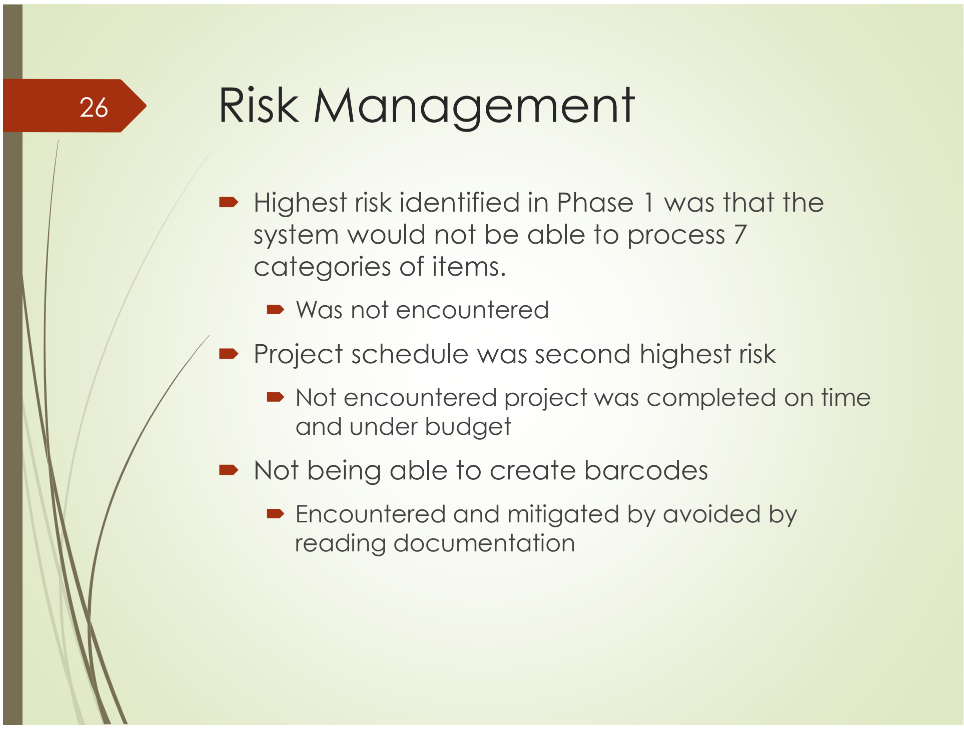
- 175 Estimated hours
- 107 actual hours



25

Monetary costs

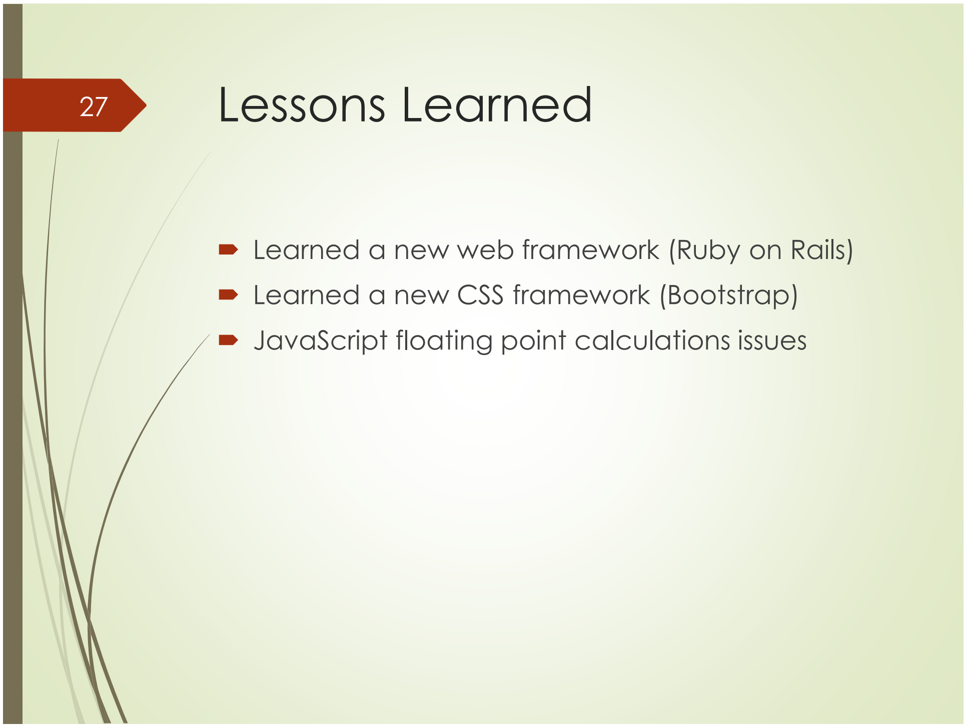
- $200.00 Estimated cost
- $157.24 Actual cost



26

Risk Management

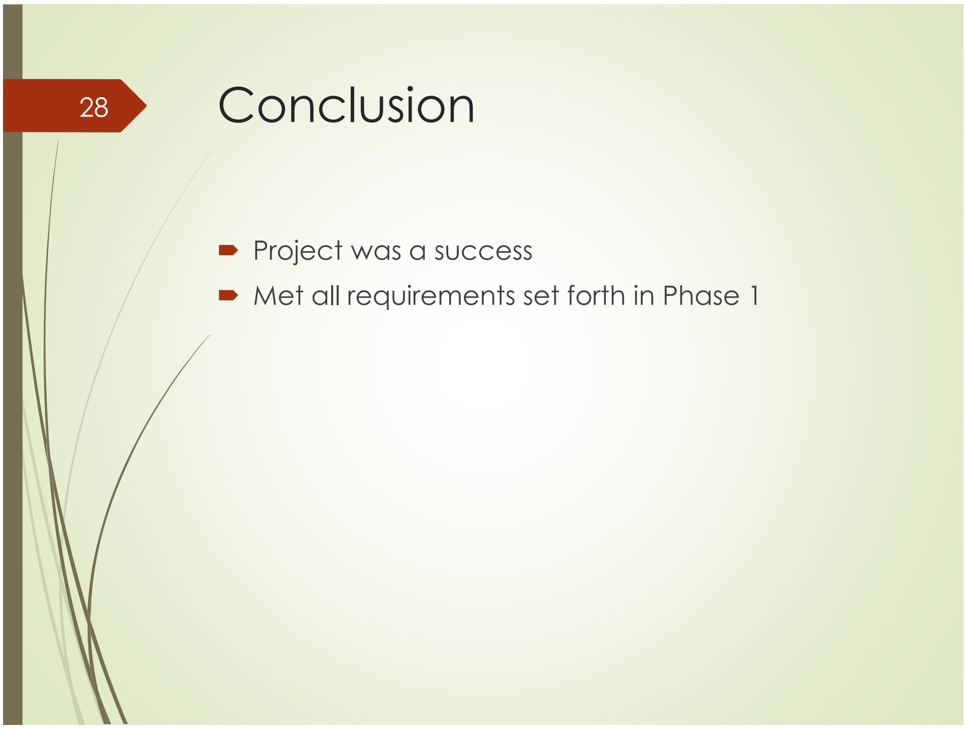
- Highest risk identified in Phase 1 was that the system would not be able to process 7 categories of items.
    - Was not encountered
- Project schedule was second highest risk
    - Not encountered project was completed on time and under budget
- Not being able to create barcodes
    - Encountered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



27

Lessons Learned

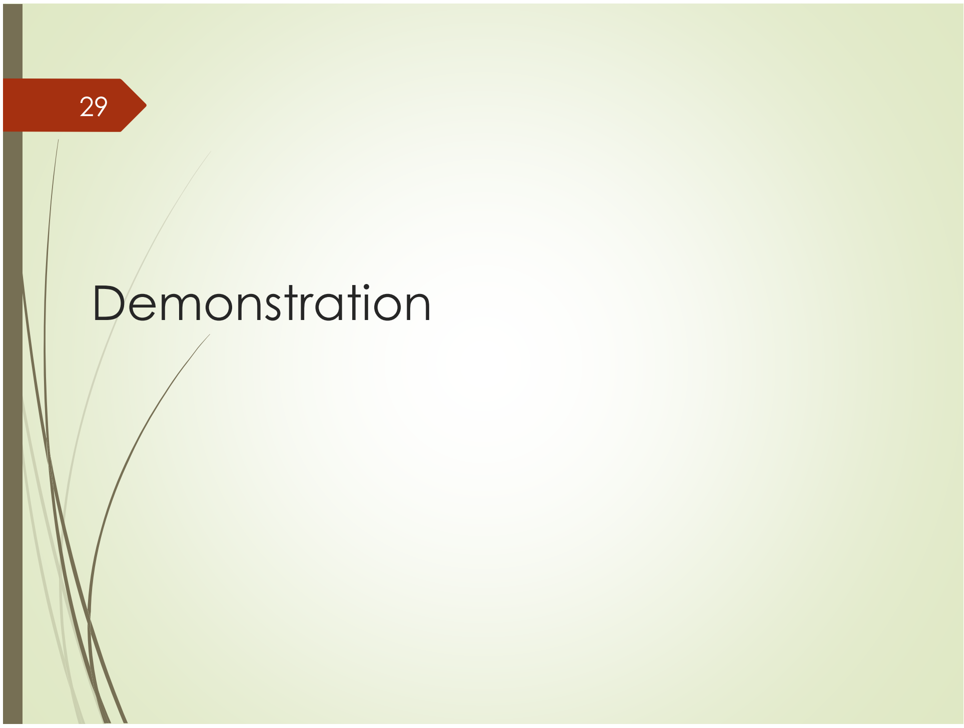
- Learned a new web framework (Ruby on Rails)
- Learned a new CSS framework (Bootstrap)
- JavaScript floating point calculations issues



28

Conclusion

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



29

Demonstration