$\odot \int$ Sale
Final Report

Group Number: 06

Version	1.0
Print Date	21/10/2012 23:59
Release Date	21/10/2012
Release State	Final
Approval State	Pending
Approved by	Chris, Dylan, Lasath, Vincent
Prepared by	Chris, Dylan, Lasath, Vincent
Reviewed by	Chris, Dylan, Lasath, Vincent
Confidentiality Category	Public

Document Revision Control

Version	Date	Authors	Summary of Changes
0.1	21/10/2012	Vincent	Created initial report structure
0.5	21/10/2012	Team	Added content

Contents

1 Executive Summary

2 Requirements

Our requirements reflect on the core business scope we set for this project: to develop a point of sale system which has functionality that will assist businesses in managing the logistics involved in selling their products or services.

This is broken down into 4 main core functionality of the system; Accurate Stock Control management, Instantaneous Customer Services, Sales and inventory reporting and lastly, a Safe and encrypted system . This is reflected in our goal requirements and is largely unchanged since the initial requirements report.

However, many domain level and product level requirements were revised over the course of this project. The main reason is to include more detail into our requirements to better reflect our system, and secondly to add in new and innovative feature to our system for a more enjoyable experience for our stakeholders while using our system.

In the requirements listed below, the number colour show if it is a functional requirement (Black) or a non functional requirement (Blue). There are also brackets next to the description to label the requirement to be either a core part of the system ([core]) or an extension of a system ([extension]). In the given prototype, all extensions have been implemented, but clients have an option to opt out various extension functionality and the core part of the system will still work without omission. Following that is the page number link to the event-B code and the Ruby on Rails code in the appendix.

Key:
black = Functional Requirements
Blue = Non-Functional Requirements

2.1 Goal-Level Requirements

Table 1: Table of Goal-Level Requirements

ReqID	Requirement	Short Description	EB	RR
GL-1	To build a system that will manage Stock Control	**[Core]** The system must have the ability to alter the stock levels, and relocate stock to the correct locations, somewhat autonomously.	10	10
GL-2	To build a system that will provide users with functionality to support Customer Service	**[Core]** The system must provide a set of features which will enable the user to perform task associated with customer service	10	10
GL-3	To build a system capable of reporting	**[Extension]** The system will generate different kinds of reports including productivity and sales	10	10
GL-4	Maintain business functionality	**[Core]** The system must be intuitive and secure, allowing multiple levels of authentication with minimal learning curve to maximise profits.	10	10

2.2 Domain-Level Requirements

Table 2: Table of Domain-Level Requirements

ReqID	Requirement	Short Description	EB	RR
DN-1.1	The system should provide the capability to modify the current stock data.	**[Core]** The system will be able to modify quantities of each particular stock. This includes creation and deletion of products, changing product details, changing location stock levels.	10	10
DN-1.2	To provide a system which can log damage, loss, and theft	**[Extension]** Essentially staff can put in the affected stock and its state, by which the system will record it and make necessary updates to stock states. Also any stolen or missing items can be resolved by staff if they are recovered.	10	10
DN-1.3	Support faults and returns of Products in the system.	**[Core]** Sometimes manufacturers can ship faulty products. The system should be able log when such an event occurs and assist in returning such products.	10	10
DN-1.4	Handle reordering and relocation of stock.	**[Core]** When floor stock levels for any product falls below a specified threshold, the system should automatically be able to request extra stock from a warehouse or another location.	10	10
DN-1.5	The system must handel various stock locations.	**[Core]** The system should support the creation, modification of various stock levels in the business. This includes backroom, warehouse etc.	10	10
DN-2.1	Must support Orders and Sales throughout the system.	**[Core]** This deals with the inventory side of orders and sales. Ability to create and handle orders/sales within the system, while update stock levels and income into the system.	10	10
DN-2.2	The system must be capable of refunding and or exchange items within the system	**[Core]** This requirement allows such processes as exchange of stock for store credit, updates stock level as appropriate, refunds for returns, and the ability to recalculate a customer's total bill.	10	10
DN-2.3	The system must be able to process payments, Billings and Transactions	**[Core]** The payments system will be outsourced however out system must be able to provide the appropriate information and update the appropriate revenue while maintaining confidentiality.	10	10
DN-2.4	Provide a system which allows for individual customer accounts	**[Extension]** The system will allow for the creation of customer accounts, and support adjustments of customer details. It will also support a loyalty program and apply various discounts.	10	10

continued on next page...

Table 2: Table of Domain-Level Requirements *Continued*

ReqID	Requirement	Short Description	E	R
DN-3.1	Include the ability to report on stocks	**[Extension]** Ability to report on stock quantities, report on how much stock has gone in and out of a location, alert for high and low stock levels.	10	10
DN-3.2	The system must have the ability to provide sales reports for management.	**[Extension]** Sales reports are generated from current stock levels, as well as history of sales, and supplier orders, sales in particular period based on product category, leading and trailing product sales and profitability, total sales based on location.	10	10
DN-3.3	The system must have the ability to report on system users.	**[Extension]** User reports includes employee reports and also customer reports. It is also able to generate employee detail reports and various other useful reports regarding users.	10	10
DN-4.1	Support user authentication and multiple levels of authorisation	**[Core]** The system will include functionality to allow users of the system to authenticate and contain various levels of access control.	10	10
DN-4.2	Provide user support.	**[Extension]** Enable users to access documentation and support for the system on demand.	10	10
DN-4.3	The system must be reasonable in its response times to given actions	**[Extension]** The system must be able to respond quick enough that the business benefits from the use of the POS.	10	10
DN-4.4	The system must be stable in its completed state	**[Core]** Both in terms of system crashes, bugs and misinformation. This essentially outlines that the system must work as expected.	10	10
DN-4.5	The system will provide backup	**[Extension]** The system is able to provide both onsite and offsite backup for various data used in the system.	10	10

2.3 Product-Level Requirements

Table 3: Table of Product-Level Requirements

ReqID	Requirement	Short Description	EB	RR
PD-1.1.1	Ability to add/remove stock from a location.	**[Core]** Stock can be rearranged from different locations i.e. when stock levels are low on the floor stock should be moved from the store rooms or the warehouse.	10	10
PD-1.1.2	Add new products to the database	**[Core]** When the store decides to sell a new product, the staff should be able to enter the product into the system, and record any relevant details.	10	10
PD-1.1.3	Update a products details	**[Core]** The products recorded in the system should be editable. For example, current stock levers, unit price, product description, etc.	10	10
PD-1.1.4	Remove a product from the system	**[Core]** If the store decides to discontinue the sale of a particular product, functionality to remove it will be provided so that the system will cease to manage the stock.	10	10
PD-1.1.5	System should allow change in product's to be activation status	**[Core]** Authorised Staff member should be able to activate or deactivate a product.	10	10
PD-1.2.1	Log an item as lost or stolen	**[Extension]** Ability to log if any item that is managed by the system is lost or stolen. This information can then be included in the various reports that are generated by the system.	10	10
PD-1.2.2	Resolve an item previously reported as lost	**[Extension]** If a lost or stolen item is found, the the system will be able to take that data and cancel any actions it may have commenced in response to it being missing.	10	10
PD-1.3.1	Ability to report faulty or damaged items received from suppliers	**[Extension]** If an item is received from a supplier is found to be faulty, then allow such an instance to be logged within the system so that it can be dealt with appropriately.	10	10
PD-1.3.2	Warranties and repairs for sold items	**[Extension]** Log and track when an item is brought back for repairs and include any current warranty status.	10	10
PD-1.4.1	Function to order new stock from supplier	**[Core]** When stock is below the threshold for warehouse stock, a purchase order must be placed with the respective supplier.	10	10
PD-1.4.2	Ability to request stock from other locations	**[Core]** When stock is below the threshold at a particular location (e.g. on the floor, in back store room, or from the warehouse), the system must be able to relocate it to the relevant place.	10	10

continued on next page...

Table 3: Table of Product-Level Requirements *Continued*

ReqID	Requirement	Short Description	E	R
PD-1.4.3	Ability to edit and cancel a stock order	**[Core]** If an order is placed within the system, an authorised staff member can edit the order while the order is still in progress or even cancel the order overall. For example A spot sale of item X was very well received by customers and sells out quickly. The duty manager raises an urgent replenishment request for item X through the PoSWare system, which then sets in train an extraordinary delivery.	10	10
PD-1.4.4	Allow stock level thresholds to be set	**[Core]** Allow an authorised user to set the stock level threshold for an item. For example, item X should have a minimum threshold of m and a maximum threshold of n on the store's floor shelves.	10	10
PD-1.5.1	Ability to add new stock location	**[Core]** Stock location can be created when new warehouse/ store is used. Authorised staff should be able to create new stock location and record any relevant details.	10	10
PD-1.5.2	Ability to edit stock location	**[Core]** Stock location's name,threshold amount and other details can be modified .	10	10
PD-1.5.3	Ability to delete stock location	**[Core]** Stock location can be deleted, but stock location must have 0 stock left in order for it to be able to be deleted.	10	10
PD-2.1.1	The system will allow customers to place products in an cart	**[Core]** Customers can place a set of products in the cart for purchasing.	10	10
PD-2.1.2	The system will be able to process the sale of goods and updating the appropriate stock levels	**[Core]** When a product is sold, the system will reduce stock levels of the particular product. If stock level then falls below a predetermined threshold, triggers relevant actions within the system.	10	10
PD-2.1.3	The system will calculate total purchasing price of stock	**[Core]** Calculates the cost of the purchased items in stock, including the ability to account for any specials on the item being purchased.	10	10
PD-2.1.4	The system will able to operate by multiple users in multiple terminals	**[Core]** the system should be able to support multiple users accessing the database at the same time.	10	10
PD-2.1.5	The system should allow user to edit or remove products from carts	**[Core]** Users can edit or remove individual products from the cart list before the transaction is gone through. This includes changing the amount, or removing a product from the order.	10	10
PD-2.2.1	Refund provision for returned stock	**[Core]** When stock is returned and is still in purchasable condition, it may be added back to the current stock.	10	10
PD-2.2.2	The system will handle exchange of stock for store credit	**[Extension]** The value of the item may be credited to a users account or next purchase after a valid return of the product.	10	10

continued on next page...

Table 3: Table of Product-Level Requirements *Continued*

ReqID	Requirement	Short Description	E	R
PD-2.2.3	The system will handle exchange of stock for cash refund	**[Core]** The item may be returned and exchanged for cash where applicable.	10	10
PD-2.3.1	The system will have a customer payment system for orders and sales	**[Core]** The payment will be validated and then recorded as a transaction within the system.	10	10
PD-2.3.2	The system will be able to update revenue as sales are made	**[Core]** Records of the sales and transactions are consolidated within the system.	10	10
PD-2.3.3	The system will be able to update tax(GST) as sales are made	**[Core]** tax will be calculated and apply to sales and ordering.	10	10
PD-2.4.1	The system will be able to allocate membership discounts to appropriate customers	**[Extension]** Where applicable for certain loyalty memberships discounts will be applied to their transactions.	10	10
PD-2.4.2	The system will handle customer account creation	**[Extension]** Users will be able to create a new account for a customer.	10	10
PD-2.4.3	The system will allow the revision of a customers details of customer account	**[Extension]** Customers with accounts will be able to edit their contact details, as well as any subscriptions and discounts within their account. System also have the ability to change the discount level for users.	10	10
PD-2.4.4	The system will allow cancellation of customer account	**[Extension]** Customers also have the option of deleting or deactivating their account if needed be.	10	10
PD-2.4.4	The system will have the functionality to remove a customer	**[Extension]** If a customer wishes to no longer take part in any programs offered by the store, there should be a way to disable that customer account in the system.	10	10
PD-3.1.1	The system allows reporting on loss/damages/theft based on cause	**[Extension]** When an item is reported as lost, stolen or damaged, there should also be a way of reporting the exact cause and (optionally) who is responsible so that it may be included in reports generated by the system.	10	10
PD-3.1.2	The system needs to be capable of generating reports based on products	**[Extension]** At the request of a manager (or anyone with sufficient privileges), the system should be able to generate a report outlining the amount of products in inventory.	10	10
PD-3.2.1	The system needs to be capable of generating reports based on sales	**[Extension]** At the request of a manager (or anyone with sufficient privileges), the system should be able to generate a report outlining the amount of sales each product has.	10	10
PD-3.2.2	The system needs to be capable of generating financial reports	**[Extension]** At the request of a manager (or anyone with sufficient privileges), the system should be able to generate a standard financial report outlining the revenue and profit of the company/ individual store.	10	10

continued on next page...

Table 3: Table of Product-Level Requirements *Continued*

ReqID	Requirement	Short Description	E	R
PD-3.3.1	The system needs to be capable of generating reports based on customers	**[Extension]** At the request of a manager (or anyone with sufficient privileges), the system should be able to generate a report outlining the customer, their amount purchased and membership type.	10	10
PD-3.3.2	The system needs to be capable of generating reports based on employees	**[Extension]** At the request of a manager (or anyone with sufficient privileges), the system should be able to generate a report outlining the employees of the business along with the number of sales they made and amount of sales they made.	10	10
PD-4.1.1	User Authentication and creation	**[Core]** Ability for a user to be created and also easily login to the system with their credentials so that their authorisation level may be determined.First user (usually owner) should be created by default	10	10
PD-4.1.2	Provide various levels of access control to the system.	**[Core]** Create ACLs to restrict functionality to specified groups of users. For example, a customer should not be able to modify the price of a product.	10	10
PD-4.1.3	Allow modification of access rights	**[Core]** The rights defined in the previous requirement should be modifiable by someone with sufficient rights. For example, if a cashier gets promoted to a manager, they will now have access to more functions within the system.	10	10
PD-4.2.2	The system includes help documentation outlining its operation	**[Core]** Provide a useful interface in such a way that help is accessible at any point while using the system.	10	10

2.4 Design-Level Requirements

Table 4: Table of Design-Level Requirements

ReqID	Requirement	Short Description	EB	RR
DZ-2.1.1.1	Barcode recognition	**[Core]** The barcode recognition system must comply with the ISO/IEC 15426-1 (linear) or ISO/IEC 15426-2 (2D).	10	10
DZ-4.2.2.1	Easily accessible help button	**[Core]** The system should have built in support, and should have an intuitive way of allowing users to access it from any point within the system.	10	10

6 Discussion of possible physical deployment

7 Assessment of development process

8 Project management

9 Special features

10 Reflections and Introspection

11 Appendix

11.1 Ruby on Rails code

11.1.1 Product Model

```
class Product < ActiveRecord::Base
  has_many :stock_levels, :dependent => :destroy
  has_many :stock_locations, :through => :stock_levels
  has_many :sale_items
  has_many :sales, :through => :sale_items
  has_many :transactions, :through => :sales
  has_many :supplier_stock_orders
  belongs_to :supplier
  attr_accessible :cost, :description, :name, :price, :barcode, :supplier, :brand, :size, :ac


  validates :name, :description, :price,:brand, :size, :cost, :barcode, :supplier, :presence

  validates :price, :cost, :numericality => {:greater_than_or_equal_to => 0}

  validates :barcode, :uniqueness => true

  validates_associated :stock_levels

  def total_stock
      stock_levels.sum(&:quantity)
  end

  def total_sold
    sale_items.sum(&:quantity)
  end

  def total_on_order
    supplier_stock_orders.where(:status => ['Created', 'Processed']).sum(&:quantity)
  end

  def revenue
    sale_items.sum(&:quantity).to_f * price.to_f
  end

  def total_cost
    sale_items.sum(&:quantity).to_f * cost.to_f
  end

end
```

11.1.2 Refund Model

```
class Refund < ActiveRecord::Base
  belongs_to :sale_item
  belongs_to :checkout_user, :class_name => 'User'
```

```
  has_one :products, :through => :sale_items
  has_one :sale, :through => :sale_items

  attr_accessible :quantity, :reason

  validate :stock_not_already_returned

  def stock_not_already_returned
    @refunds = Refund.find_all_by_sale_item_id(sale_item.id)
    @quantity_returned = @refunds.sum(&:quantity)
    @quantity_available = sale_item.quantity - @quantity_returned
    if sale_item.quantity == 0
      errors.add(:base, 'Product was not found in sale')
    elsif quantity > @quantity_available
      errors.add(:quantity, "of item return is too high. There are only #{@quantity_available}
    end
  end

end
```

11.1.3 Sales Model

```
class Sale < ActiveRecord::Base
    has_many :sale_items, :dependent => :destroy
    has_many :products, :through => :sale_items
    has_many :transactions

    belongs_to :customer, :class_name => 'User'
    belongs_to :checkout_user, :class_name => 'User'

    before_save :check_customer

    attr_accessible :customer, :checkout_user, :discount, :status, :updated_at

    #Event-B: transactionInProcess  members TRANSACTIONTYPE
    #         axm3: partition(TRANSACTIONTYPE, {ADDINGTOCART},{CHECKINGOUT},{FINISHED})
    #Comment: These was just renamed but serves the exact same person.

    validates :status,
       :inclusion  => { :in => [ 'Adding to Cart', 'Checking Out', 'Finished'],
       :message    => "%{value} is not a valid status" }


    # Various other methods.
    def total
      sale_items.sum(&:sub_total)
    end

    def amount_paid
      transactions.sum(&:amount)
```

```
    end

    def discount
      if customer && customer.discount
        customer.discount / 100 * total
      else
        0
      end
    end

    def change_given
      [amount_paid - total, 0].max
    end

    def check_customer
      if customer.nil?
        customer = User.find_by_email('default@pos.com')
      end
    end
end
```

11.1.4 SaleItem Model

```
class SaleItem < ActiveRecord::Base
  #Event-B: CART = PRODUCT
  #Summary: This class represents the cart set from our model
  belongs_to :sale
  belongs_to :product

  attr_accessible :sale, :product, :quantity, :sub_total
  validates_presence_of :sale, :product, :quantity

  validate :product_is_active
  validate :deduct_stock
  before_destroy :restore_stock

  def stock_level
    if @stock_level.nil?
      @stock_level = StockLevel.find_by_product_id_and_stock_location_id(product, StockLocati
    end
    return @stock_level
  end

  def product_is_active
    errors.add(:product, "is not active") unless product.active == 't'
  end

  def deduct_stock
    if quantity_changed? and quantity_was
      stock_level.quantity += quantity_was
    end
```

```
    stock_level.quantity -= quantity
    if stock_level.quantity >= 0
      stock_level.save!
    else
      errors.add(:quantity, "cannot exceed current floor stock")
      return false
    end
  end

  def restore_stock
    stock_level.quantity += quantity
    stock_level.save
  end
end
```

11.1.5 StockLevel Model

```
class StockLevel < ActiveRecord::Base
  #Event-B: productmaxthreshold  products  (STOCK_LOCATION  )
  #Event-B: productlevels  products (STOCK_LOCATION  )

  belongs_to :product
  belongs_to :stock_location
  attr_accessible :quantity, :threshold, :product, :stock_location

  # Event-B: p,l  p activeProducts  l  STOCK_LOCATION  productmaxthreshold(p)(l)  productthres
  # Validate that stock_level is above miniumum threshold, and automatically reorder if quant
  validates :quantity, :threshold, :numericality => {:greater_than_or_equal_to => 0}
  after_save :automatic_reorder




  def below
   quantity < threshold
  end

  def automatic_reorder
    if below
      if stock_location.previous_location
        #stock transfer
        exsting_transfer = StockTransfer.find_by_product_id_and_stock_location_id_and_complet
        if exsting_transfer
          exsting_transfer.quantity = threshold - quantity
          exsting_transfer.save!
        else
          StockTransfer.create!(
            :product => product,
            :stock_location => stock_location,
            :quantity => threshold - quantity,
            :complete => false
```

```
          )
        end
      else
        #supplier stock order
        existing_order = SupplierStockOrder.find_by_product_id_and_status(product, 'Created')
        if existing_order
          existing_order.quantity = threshold - quantity
          existing_order.save!
        else
          SupplierStockOrder.create!(
            :product => product,
            :quantity => threshold - quantity,
            :status => 'Created'
          )
        end
      end
    end
  end
end
```

11.1.6 StockLocation Model

```
class StockLocation < ActiveRecord::Base
  has_many :stock_levels, :dependent => :destroy
  has_many :products, :through => :stock_levels

  has_one :previous_location
  belongs_to :previous_location, :class_name => 'StockLocation'

  attr_accessible :id,:name, :previous_location

  validates :name, :presence => true, :uniqueness => true

end
```

11.1.7 StockTransfer Model

```
class StockTransfer < ActiveRecord::Base
  belongs_to :product
  belongs_to :stock_location
  attr_accessible :product, :stock_location, :complete, :quantity

   validates :product, :stock_location, :quantity, :presence => true
   validates :quantity, :numericality => {:greater_than => 0}

end
```

11.1.8 Supplier Model

```
class Supplier < ActiveRecord::Base
```

```
  has_many :products
  attr_accessible :contact_number, :contact_person, :name

  validates :name, :contact_number, :contact_person, :presence => true
  validates :name, :uniqueness => true

end
```

11.1.9 SupplierStockOrder Model

```
class SupplierStockOrder < ActiveRecord::Base
  belongs_to :product
  attr_accessible :product, :quantity, :status

  validates :status,
       :inclusion  => { :in => [ 'Created', 'Processed', 'Completed'],
       :message    => "%{value} is not a valid status" }

  validates :product, :quantity, :status, :presence => true
  validates :quantity, :numericality => {:greater_than => 0}


  def total
   quantity * product.cost
  end
end
```

11.1.10 Transaction Model

```
class Transaction < ActiveRecord::Base
  belongs_to :sale

  attr_accessible :amount, :approved, :method, :sale
  validates_presence_of :amount, :approved, :method, :sale

  validates :amount, :exclusion =>
   {:in => [0], :message => "Must be non-zero amount." }

  validates :method,  :inclusion => { :in => ['Cash', 'Other']}

  validates :amount, :if => Proc.new {method == 'Cash'},
   :inclusion => {:in => [0.05, 0.10, 0.20, 0.50, 1.00, 2.00, 5.00, 10.00, 20.00, 50.00, 100.

  validates_acceptance_of :approved, :accept => true
end
```

11.1.11 User Model

```
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
```

```
  devise :database_authenticatable, :token_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :timeoutable, :validatable

  has_many :sales_checkout, :class_name => 'Sale', :foreign_key => "checkout_user_id"
  has_many :sales_customer, :class_name => 'Sale', :foreign_key => "customer_id"

  attr_accessible :role, :name, :postcode, :discount, :membership, :active, :email, :password

  #Event-B: {Stock_ctx_R0} axm2: partition(USER_PRIVILEGE,{Cashier},{Stock_Control},{Manager}
  #Comment: Default was added as a result of merging the users and members
  validates :role,
    :inclusion  => { :in => [ 'Owner', 'Manager', 'Stock Control', 'Cashier', 'Default'],
    :message    => "%{value} is not a valid status" }


  #Event-B: {StockControl_R4} grd: userPrivileges(user)  {Stock_Control, Manager, Owner, Cash
  #Comment: Used as guards throughout the model
  def can_checkout
    role == "Owner" or role == "Manager" or role == "Stock Control" or role =="Cashier"
  end

  #Event-B: {StockControl_R4} grd: userPrivileges(user)  {Stock_Control, Manager, Owner}
  #Comment: Used as guards throughout the model
  def can_manage_stock
    role == "Owner" or role == "Manager" or role == "Stock Control"
  end

  #Event-B: {StockControl_R4} grd: userPrivileges(user)  {Manager, Owner}
  #Comment: Used as guards throughout the model
  def can_report
    role == "Owner" or role == "Manager"
  end


  #Various Methods
  def num_sales
    sales_checkout.count
  end

  def num_purchases
    sales_customer.count
  end

  def total_sales
    sales_checkout.sum(&:total)
  end

  def total_purchases
    sales_customer.sum(&:total)
  end

end
```

11.1.12 ApplicationController

```
class ApplicationController < ActionController::Base
  before_filter :authenticate_user!
  protect_from_forgery
end
```

11.1.13 HomeController

```
class HomeController < ApplicationController
  def index
  end

  def help
  end
end
```

11.1.14 ProductsController

```
class ProductsController < ApplicationController
  skip_before_filter  :authenticate_user!, :only => [:index, :show]
  # GET /products
  # GET /products.json
  def index
    @products = Product.order("name")

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @products }
    end
  end

  # GET /products/1
  # GET /products/1.json
  def show
    @product = Product.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @product }
    end
  end

  # GET /products/new
  # GET /products/new.json
  def new
    @product = Product.new

    StockLocation.all.each do |s|
```

```
    @product.stock_levels.build(:stock_location => s)
  end

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @product }
  end
end

# GET /products/1/edit
def edit
  @product = Product.find(params[:id])

  StockLocation.all.each do |s|
    if !StockLevel.exists?(:product_id => params[:id].to_i, :stock_location_id =>s.id)
      @product.stock_levels.build(:stock_location => s)
    end
  end

end

# POST /products
# POST /products.json
# Event-b: NewProduct
def create
  #raise params.inspect
  @supplier = nil
  if(params[:product][:supplier] != "")
  @supplier = Supplier.find_by_id(Integer(params[:product][:supplier]))
  end

  params[:product][:supplier] = @supplier

  # Event-b: grd1:  product  PRODUCTproducts
  @product = Product.new(params[:product])

  params[:stock_level].each do |sl_id, sl|
    @product.stock_levels.build(:stock_location => StockLocation.find(sl_id.to_i), :quantit
  end

  respond_to do |format|
    # Event-b: act1: products  products  {product}
    # Event-b: act2:  productprice(product)  price
    if @product.save
      format.html { redirect_to @product, notice: 'Product was successfully created.' }
      format.json { render json: @product, status: :created, location: @product }
    else
      format.html { render action: "new" }
      format.json { render json: @product.errors, status: :unprocessable_entity }
    end
  end
end
```

```
# PUT /products/1
# PUT /products/1.json
# Event-b: UpdateProduct
def update
  # Event-b: grd1:  product  products
  @product = Product.find(params[:id])

  params[:stock_level].each do |sl_id, sl|
    begin
    @stock_id = StockLevel.find(:first, :conditions => {:product_id => params[:id].to_i, :s
      @product.stock_levels.update(@stock_id, :quantity =>sl[:quantity], :threshold => sl[:
    rescue ActiveRecord::RecordNotFound
      @product.stock_levels.build(:stock_location => StockLocation.find(sl_id.to_i), :quant
    end
  end

  respond_to do |format|
    @supplier = Supplier.find(Integer(params[:product][:supplier]))
    params[:product][:supplier] = @supplier

    # Event-b: act1:  productprice(product)  price
    if @product.update_attributes(params[:product])
      format.html { redirect_to @product, notice: 'Product was successfully updated.' }
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
      format.json { render json: @product.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /products/1
# DELETE /products/1.json
def destroy
  @product = Product.find(params[:id])
  @product.destroy

  respond_to do |format|
    format.html { redirect_to products_url }
    format.json { head :no_content }
  end
end

# GET /products/1/activate
# Event-b: ActivateProduct
def activate
  # Event-b: grd1: product  products
  @product = Product.find(params[:id])

  # Event-b: act1:  activeProducts  activeProducts  {product}
  @product.update_attribute(:active,true)
```

```
    @product.save


    respond_to do |format|
      format.html { redirect_to products_url }
      format.json { head :no_content }
    end
  end

  # GET /products/1/deactivate
  # Event-b: DeactivateProduct
  def deactivate
    # Event-b: grd1: product  products
    @product = Product.find(params[:id])

    # Event-b: act1:  activeProducts  activeProducts  {product}
    @product.update_attribute(:active,false)
    @product.save

    respond_to do |format|
      format.html { redirect_to products_url }
      format.json { head :no_content }
    end
  end

end
```

11.1.15 RefundsController

```
class RefundsController < ApplicationController
  # GET /refunds
  # GET /refunds.json
  def index
    @refunds = Refund.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @refunds }
    end
  end

  # GET /refunds/1
  # GET /refunds/1.json
  def show
    @refund = Refund.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @refund }
    end
  end
```

```
# GET /refunds/new
# GET /refunds/new.json
def new
  @refund = Refund.new
  @refund.sale_item = SaleItem.find(params[:sale_item])

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @refund }
  end
end

# GET /refunds/1/edit
def edit
  @refund = Refund.find(params[:id])
end

# POST /refunds
# POST /refunds.json
def create
  @refund = Refund.new(params[:refund])
  @refund.sale_item = SaleItem.find(params[:sale_item_id])
  @refund.checkout_user = current_user
  @refund.total = (@refund.sale_item.sub_total / @refund.sale_item.quantity) * @refund.quan

  # Update stock levels
  @stock_location = StockLocation.where("previous_location_id is NULL")
  @stock_level = @refund.sale_item.product.stock_levels.find_by_stock_location_id(@stock_lo
  @stock_level.quantity += @refund.quantity

  respond_to do |format|
    if @refund.save and @stock_level.save
      format.html { redirect_to @refund, notice: 'Refund was successfully created.' }
      format.json { render json: @refund, status: :created, location: @refund }
    else
      format.html { render action: "new" }
      format.json { render json: @refund.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /refunds/1
# PUT /refunds/1.json
def update
  @refund = Refund.find(params[:id])
  @refund.total = (@refund.sale_item.sub_total / @refund.sale_item.quantity) * params[:refu

  # Get previous quantity
  @previous_quantity = @refund.quantity

  # Update stock levels
```

```
    @stock_location = StockLocation.where("previous_location_id is NULL")
    @stock_level = @refund.sale_item.product.stock_levels.find_by_id(@stock_location)
    @stock_level.quantity += (@refund.quantity - @previous_quantity)

    respond_to do |format|
      if @refund.update_attributes(params[:refund])
        @stock_level.save

        format.html { redirect_to @refund, notice: 'Refund was successfully updated.' }
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @refund.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /refunds/1
  # DELETE /refunds/1.json
  def destroy
    @refund = Refund.find(params[:id])
    @refund.destroy

    respond_to do |format|
      format.html { redirect_to refunds_url }
      format.json { head :no_content }
    end
  end

  def search
    matches = Sale.where(:id => params[:sale_id])
    if matches.any?
      redirect_to matches.first
    else
      redirect_to refunds_path, alert: 'Invalid Sale ID'
    end
  end
end
```

11.1.16 ReportsController

```
class ReportsController < ApplicationController
  def index
  end

  def sale
    @sales = Sale.where(:status =>"Finished")

    @users = User.all

    @h = LazyHighCharts::HighChart.new('graph') do |f|
```

```
    f.options[:chart][:defaultSeriesType] = "area"
    f.options[:title][:text] = "Sales By Customer"
    f.options[:yAxis][:title][:text] = "Total of Sale"
    f.options[:xAxis] = { :title=>{:text=>"Date"}, :type => 'datetime',:dateTimeLabelFormat



    @users.each do |u|
      f.series(:name=>u.name,
        :data=>u.sales_customer.where(:status =>"Finished").pluck(:updated_at).zip(u.sales_
      #User.find(4).sales_customer.pluck(:updated_at).zip(User.find(4).sales_customer.map(&
        )
    end
  end


  respond_to do |format|
    format.html # sale.html.erb
  end
end

def stock
  @products = Product.all

  respond_to do |format|
    format.html # suppliers.html.erb
  end
end

def financial
  @products = Product.all
  @revenue = @products.sum(&:revenue)
  @cost = @products.sum(&:total_cost)
  @profit = @revenue - @cost

  @gst = @revenue * -0.1
  @tax =  @gst*0.3

  @income = @profit + @gst + @tax


end

def staff
  @users = User.where(:role => [ 'Owner', 'Manager', 'Stock Control', 'Cashier'])

  respond_to do |format|
    format.html # suppliers.html.erb
  end
end

def supplier
```

```
    @supplier_stock_orders = SupplierStockOrder.all

    respond_to do |format|
      format.html # suppliers.html.erb
    end
  end

  def customer
    @users = User.all

    respond_to do |format|
      format.html # suppliers.html.erb
    end
  end
end
```

11.1.17 SalesController

```
class SalesController < ApplicationController
  # GET /sales
  # GET /sales.json
  def index
    @sales = Sale.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @sales }
    end
  end

  # GET /sales/1
  # GET /sales/1.json
  def show
    @sale = Sale.find(params[:id])
    @sale_items = @sale.sale_items

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @sale }
    end
  end

  # GET /sales/new
  # GET /sales/new.json
  def new
    @current_sales = Sale.find_all_by_checkout_user_id_and_status(current_user.id, ['Adding t

    if @current_sales.empty?
      @sale = Sale.new(:checkout_user  => current_user, :status => 'Adding to Cart')
      @sale.save
```

```
    respond_to do |format|
      format.html { redirect_to edit_sale_path(@sale) }
      format.json { render json: @sale}
    end
  else
    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @current_sales }
    end
  end
end

# GET /sales/1/edit
def edit
  @sale = Sale.find(params[:id])

  case @sale.status
  when 'Adding to Cart'
    @sale_item = SaleItem.new({:sale => @sale})
    render @sale.status.parameterize.underscore
  when 'Checking Out'
    @transaction = Transaction.new(:sale => @sale)
    render 'checking_out'
  when 'Finished'
    redirect_to sale_path(@sale), :error => "Can't edit a finished sale"
  end
end

# POST /sales
# POST /sales.json
def create
  @sale = Sale.new(:checkout_user  => current_user, :status => 'Adding to Cart')
  @sale.save

  respond_to do |format|
    format.html { redirect_to edit_sale_path(@sale) }
    format.json { render json: @sale}
  end
end

# PUT /sales/1
# PUT /sales/1.json
def update
  params[:sale][:customer] = User.find_by_id(params[:sale][:customer])
  @sale = Sale.find(params[:id])

  respond_to do |format|
    if @sale.update_attributes(params[:sale])
      format.html { redirect_to edit_sale_path(@sale), notice: 'Sale was successfully updat
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
```

```
      format.json { render json: @sale.errors, status: :unprocessable_entity }
    end
  end
end

def checkout
  @sale = Sale.find(params[:id])

  if @sale.status != 'Adding to Cart'
    redirect_to sales_path, alert: 'You can only proceed to payment from adding to cart'
  end

  @sale.status = 'Checking Out'
  @sale.save!

  respond_to do |format|
    format.html { redirect_to edit_sale_path(@sale) }
    format.json { head :no_content }
  end
end

def complete
  @sale = Sale.find(params[:id])

  if @sale.status != 'Checking Out'
    redirect_to sales_path, alert: 'You can only finish a sale during checkout.'
    return
  end

  if @sale.total > @sale.amount_paid + @sale.discount
    redirect_to edit_sale_path(@sale), alert: 'You must finish payment before completing a
    return
  end

  @sale.status = 'Finished'
  @sale.save!

  respond_to do |format|
    format.html { redirect_to @sale, notice: 'Sale complete.' }
    format.json { head :no_content }
  end
end

# DELETE /sales/1
# DELETE /sales/1.json
def destroy
  @sale = Sale.find(params[:id])
  @sale.destroy

  respond_to do |format|
    format.html { redirect_to sales_url }
    format.json { head :no_content }
```

```
    end
  end
end
```

11.1.18 SaleItemsController

```
class SaleItemsController < ApplicationController
  # GET /sale_items
  # GET /sale_items.json
  def index
    @sale_items = SaleItem.all
    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @sale_items.to_json }
    end
  end

  # GET /sale_items/1
  # GET /sale_items/1.json
  def show
    @sale_item = SaleItem.find(params[:id])

    respond_to do |format|
      format.json { render :json => @sale_item.to_json }
    end
  end

  # GET /sale_items/new
  # GET /sale_items/new.json
  def new
    @sale = params[:sale]
    @sale_item = @sale.sale_items.build

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @sale_item }
    end
  end

  # GET /sale_items/1/edit
  def edit
    @sale_item = SaleItem.find(params[:id])
    @sale = @sale_item.sale
  end

  # POST /sale_items
  # POST /sale_items.json
  def create
    @product = Product.find_by_barcode(params[:sale_item][:product])
    params[:sale_item][:product] = @product;
```

```
    @sale = Sale.find(params[:sale_item][:sale])
    params[:sale_item][:sale] = @sale;

    @sale_item = SaleItem.find_by_sale_id_and_product_id(@sale.id, @product.id)
    if @sale_item.nil?
      @sale_item = SaleItem.new(params[:sale_item])
    else
      @sale_item.quantity += params[:sale_item][:quantity].to_i
    end

    @sale_item.sub_total = @sale_item.quantity * @sale_item.product.price

    respond_to do |format|
      if @sale_item.save
        format.json { render :show }
      else
        format.json { render json: @sale_item.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /sale_items/1
  # PUT /sale_items/1.json
  def update
    @product = Product.find(params[:sale_item][:product])
    params[:sale_item][:product] = @product;
    @sale_item = SaleItem.find(params[:id])

    respond_to do |format|
      if @sale_item.update_attributes(params[:sale_item])
        format.html { redirect_to sale_sale_items_path(@sale_item.sale), notice: 'Sale item w
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @sale_item.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /sale_items/1
  # DELETE /sale_items/1.json
  def destroy
    @sale_item = SaleItem.find(params[:id])
    @sale = @sale_item.sale
    @sale_item.destroy

    respond_to do |format|
      format.html { redirect_to sale_sale_items_path(@sale) }
      format.json { head :no_content }
    end
  end
end
```

11.1.19 StockLevelsController

```
class StockLevelsController < ApplicationController
  skip_before_filter  :authenticate_user!, :only => [:index, :show]
  # GET /stock_levels
  # GET /stock_levels.json
  def index
    @stock_levels = StockLevel.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @stock_levels }
    end
  end

  # GET /stock_levels/1
  # GET /stock_levels/1.json
  def show
    @stock_level = StockLevel.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @stock_level }
    end
  end

  # GET /stock_levels/new
  # GET /stock_levels/new.json
  def new
    @stock_level = StockLevel.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @stock_level }
    end
  end

  # GET /stock_levels/1/edit
  def edit
    @stock_level = StockLevel.find(params[:id])
  end

  # POST /stock_levels
  # POST /stock_levels.json
  def create
    @product = Product.find(Integer(params[:stock_level][:product]))
    @stock_location = StockLocation.find(Integer(params[:stock_level][:stock_location]))

    params[:stock_level][:product] = @product
    params[:stock_level][:stock_location] = @stock_location
```

```
    @stock_level = StockLevel.new(params[:stock_level])

    respond_to do |format|
      if @stock_level.save
        format.html { redirect_to @stock_level, notice: 'Stock level was successfully created
        format.json { render json: @stock_level, status: :created, location: @stock_level }
      else
        format.html { render action: "new" }
        format.json { render json: @stock_level.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /stock_levels/1
  # PUT /stock_levels/1.json
  # Event-b: SetProductLevel
  def update
    #Event-b: grd1: product  activeProducts,
    @stock_level = StockLevel.find(params[:id])

    respond_to do |format|
      #Event-b: act1: productlevels(product)  {Floor  floor,Backroom  backroom,Warehouse  war
      if @stock_level.update_attributes(params[:stock_level])
        format.html { redirect_to @stock_level, notice: 'Stock level was successfully updated
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @stock_level.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /stock_levels/1
  # DELETE /stock_levels/1.json
  def destroy
    @stock_level = StockLevel.find(params[:id])
    @stock_level.destroy

    respond_to do |format|
      format.html { redirect_to stock_levels_url }
      format.json { head :no_content }
    end
  end
end
```

11.1.20 StockLocationsController

```
class StockLocationsController < ApplicationController
  skip_before_filter  :authenticate_user!, :only => [:index, :show]
```

```
# GET /stock_locations
# GET /stock_locations.json
def index
  @stock_locations = StockLocation.order("id DESC")

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @stock_locations }
  end
end

# GET /stock_locations/1
# GET /stock_locations/1.json
def show
  @stock_location = StockLocation.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @stock_location }
  end
end

# GET /stock_locations/new
# GET /stock_locations/new.json
def new
  @stock_location = StockLocation.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @stock_location }
  end
end

# GET /stock_locations/1/edit
def edit
  @stock_location = StockLocation.find(params[:id])

end

# POST /stock_locations
# POST /stock_locations.json
def create
  @previous_location = nil
  if(params[:stock_location][:previous_location_id] != "")
    @previous_location = StockLocation.find(Integer(params[:stock_location][:previous_locat
  end
  params[:stock_location][:previous_location] = @previous_location
  params[:stock_location].delete :previous_location_id


  @stock_location = StockLocation.new(params[:stock_location])
```

```
    respond_to do |format|
      if @stock_location.save
        format.html { redirect_to stock_locations_path, notice: 'Stock location was successfu
        format.json { render json: @stock_location, status: :created, location: @stock_locati
      else
        format.html { render action: "new" }
        format.json { render json: @stock_location.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /stock_locations/1
  # PUT /stock_locations/1.json
  def update
    @previous_location = nil
    if(params[:stock_location][:previous_location_id] != "")
      @previous_location = StockLocation.find(Integer(params[:stock_location][:previous_locat
    end

    params[:stock_location][:previous_location] = @previous_location
    params[:stock_location].delete :previous_location_id

    @stock_location = StockLocation.find(params[:id])

    respond_to do |format|
      if @stock_location.update_attributes(params[:stock_location])
        format.html { redirect_to stock_locations_path, notice: 'Stock location was successfu
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @stock_location.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /stock_locations/1
  # DELETE /stock_locations/1.json
  def destroy
    @stock_location = StockLocation.find(params[:id])
    @stock_location.destroy

    respond_to do |format|
      format.html { redirect_to stock_locations_url }
      format.json { head :no_content }
    end
  end
end
```

11.1.21 StockTransfersController

```
class StockTransfersController < ApplicationController
```

```
# GET /stock_transfers
# GET /stock_transfers.json
def index
  @stock_transfers = StockTransfer.order("complete DESC")

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @stock_transfers }
  end
end

# GET /stock_transfers/1
# GET /stock_transfers/1.json
def show
  @stock_transfer = StockTransfer.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @stock_transfer }
  end
end

# GET /stock_transfers/new
# GET /stock_transfers/new.json
def new
  @stock_transfer = StockTransfer.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @stock_transfer }
  end
end

# GET /stock_transfers/1/edit
def edit
  @stock_transfer = StockTransfer.find(params[:id])
end

# POST /stock_transfers
# POST /stock_transfers.json
def create


  @product = Product.find(Integer(params[:stock_transfer][:product]))
  @stock_location = StockLocation.find(Integer(params[:stock_transfer][:stock_location]))

  params[:stock_transfer][:product] = @product
  params[:stock_transfer][:stock_location] = @stock_location

  @stock_transfer = StockTransfer.new(params[:stock_transfer])

  respond_to do |format|
```

```
    if @stock_transfer.save
      format.html { redirect_to stock_transfers_path, notice: 'Stock transfer was successful
      format.json { render json: @stock_transfer, status: :created, location: @stock_transf
    else
      format.html { render action: "new" }
      format.json { render json: @stock_transfer.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /stock_transfers/1
# PUT /stock_transfers/1.json
def update

  @stock_transfer = StockTransfer.find(params[:id])

  respond_to do |format|
    @product = Product.find(Integer(params[:stock_transfer][:product]))
    @stock_location = StockLocation.find(Integer(params[:stock_transfer][:stock_location]))

    params[:stock_transfer][:product] = @product
    params[:stock_transfer][:stock_location] = @stock_location
    if @stock_transfer.update_attributes(params[:stock_transfer])
      format.html { redirect_to stock_transfers_path, notice: 'Stock transfer was successful
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
      format.json { render json: @stock_transfer.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /stock_transfers/1
# DELETE /stock_transfers/1.json
def destroy
  @stock_transfer = StockTransfer.find(params[:id])
  @stock_transfer.destroy

  respond_to do |format|
    format.html { redirect_to stock_transfers_url }
    format.json { head :no_content }
  end
end


# Event-b: MoveStockToFloor & MoveStockToBackroom
def complete
  @stock_transfer = StockTransfer.find(params[:id])
  @product = @stock_transfer.product
  # Event-b: product  activeProducts
  @locationto = @stock_transfer.stock_location
  @locationfrom = @locationto.previous_location
```

```
    #Mark as complete
    @stock_transfer.update_attribute(:complete,true)
    @stock_transfer.save

    #Event-b : act1: productlevels(product)  productlevels(product) <+ {Floor  (productlevels
    @stock_level_to = StockLevel.find_by_product_id_and_stock_location_id(@product,@locationt
    @stock_level_to.update_attribute(:quantity, (@stock_level_to.quantity + @stock_transfer.q
    @stock_level_to.save

    @stock_level_from = StockLevel.find_by_product_id_and_stock_location_id(@product,@locatio
    @stock_level_from.update_attribute(:quantity, (@stock_level_from.quantity - @stock_transf
    @stock_level_from.save




    respond_to do |format|
      format.html { redirect_to stock_transfers_url }
      format.json { head :no_content }
    end
  end


end
```

11.1.22 SupplierStockOrdersController

```
class SupplierStockOrdersController < ApplicationController
  # GET /supplier_stock_orders
  # GET /supplier_stock_orders.json
  def index
    @supplier_stock_orders = SupplierStockOrder.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @supplier_stock_orders }
    end
  end

  # GET /supplier_stock_orders/1
  # GET /supplier_stock_orders/1.json
  def show
    @supplier_stock_order = SupplierStockOrder.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @supplier_stock_order }
    end
  end
```

```
# GET /supplier_stock_orders/new
# GET /supplier_stock_orders/new.json
def new
  @supplier_stock_order = SupplierStockOrder.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @supplier_stock_order }
  end
end

# GET /supplier_stock_orders/1/edit
def edit
  @supplier_stock_order = SupplierStockOrder.find(params[:id])
end

# POST /supplier_stock_orders
# POST /supplier_stock_orders.json

# Event-b: NewOrder
def create
  # Event-b: product  activeProducts
  @product = Product.find(params[:supplier_stock_order][:product])
  params[:supplier_stock_order][:product] = @product

  #Event-b: act1: orders  orders  {product  quantity}
  #Event-b: act2: orderStatus  orderStatus  {product  Created}
  @supplier_stock_order = SupplierStockOrder.new(params[:supplier_stock_order])

  respond_to do |format|
    if @supplier_stock_order.save
      format.html { redirect_to @supplier_stock_order, notice: 'Supplier stock order was su
      format.json { render json: @supplier_stock_order, status: :created, location: @supplie
    else
      format.html { render action: "new" }
      format.json { render json: @supplier_stock_order.errors, status: :unprocessable_entit
    end
  end
end

# PUT /supplier_stock_orders/1
# PUT /supplier_stock_orders/1.json

#Event-b: EditOrder
def update
  #Event-b: grd1: product  activeProducts
  @product = Product.find(params[:supplier_stock_order][:product])
  params[:supplier_stock_order][:product] = @product

  @supplier_stock_order = SupplierStockOrder.find(params[:id])
```

```
  respond_to do |format|
    #Event-b: act1: orders  orders <+ {product  quantity}
    if @supplier_stock_order.update_attributes(params[:supplier_stock_order])
      format.html { redirect_to @supplier_stock_order, notice: 'Supplier stock order was su
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
      format.json { render json: @supplier_stock_order.errors, status: :unprocessable_entit
    end
  end
end

# DELETE /supplier_stock_orders/1
# DELETE /supplier_stock_orders/1.json
# Event-b: CancelOrder
def destroy
  @supplier_stock_order = SupplierStockOrder.find(params[:id])
  @supplier_stock_order.destroy

  respond_to do |format|
    format.html { redirect_to supplier_stock_orders_url }
    format.json { head :no_content }
  end
end

# GET /supplier_stock_orders/1/process

#Event-b: UpdateOrderToDelivering
def process_order
  @supplier_stock_order = SupplierStockOrder.find(params[:id])


  #Mark as processed
  #Event-b: act1: orderStatus  orderStatus <+ {product  Delivering}
  @supplier_stock_order.update_attribute(:status,"Processed")
  @supplier_stock_order.save




  respond_to do |format|
    format.html { redirect_to supplier_stock_orders_url }
    format.json { head :no_content }
  end
end

# GET /supplier_stock_orders/1/complete
#Event-b: UpdateOrderToComplete & CompleteOrder
def complete
  @supplier_stock_order = SupplierStockOrder.find(params[:id])
  @product = @supplier_stock_order.product
  @stock_location = StockLocation.where("previous_location_id is NULL")
```

```
    #Mark as processed
    #Event-b: act1: orderStatus  orderStatus <+ {product  Completed}
    @supplier_stock_order.update_attribute(:status,"Completed")
    @supplier_stock_order.save

    #Event-b: productlevels(product)  productlevels(product) <+ {Warehouse  (productlevels(pr
    @stock_level = StockLevel.find_by_product_id_and_stock_location_id(@product,@stock_locati
    @stock_level.update_attribute(:quantity, (@stock_level.quantity + @supplier_stock_order.q
    @stock_level.save




    respond_to do |format|
      format.html { redirect_to supplier_stock_orders_url }
      format.json { head :no_content }
    end
  end
end
```

11.1.23 SuppliersController

```
class SuppliersController < ApplicationController
  # GET /suppliers
  # GET /suppliers.json
  def index
    @suppliers = Supplier.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @suppliers }
    end
  end

  # GET /suppliers/1
  # GET /suppliers/1.json
  def show
    @supplier = Supplier.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @supplier }
    end
  end

  # GET /suppliers/new
  # GET /suppliers/new.json
  def new
    @supplier = Supplier.new
```

```
  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @supplier }
  end
end

# GET /suppliers/1/edit
def edit
  @supplier = Supplier.find(params[:id])
end

# POST /suppliers
# POST /suppliers.json
def create
  @supplier = Supplier.new(params[:supplier])

  respond_to do |format|
    if @supplier.save
      format.html { redirect_to @supplier, notice: 'Supplier was successfully created.' }
      format.json { render json: @supplier, status: :created, location: @supplier }
    else
      format.html { render action: "new" }
      format.json { render json: @supplier.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /suppliers/1
# PUT /suppliers/1.json
def update
  @supplier = Supplier.find(params[:id])

  respond_to do |format|
    if @supplier.update_attributes(params[:supplier])
      format.html { redirect_to @supplier, notice: 'Supplier was successfully updated.' }
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
      format.json { render json: @supplier.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /suppliers/1
# DELETE /suppliers/1.json
def destroy
  @supplier = Supplier.find(params[:id])
  @supplier.destroy

  respond_to do |format|
    format.html { redirect_to suppliers_url }
```

```
      format.json { head :no_content }
    end
  end
end
```

11.1.24 TransactionsController

```
class TransactionsController < ApplicationController
  # GET /transactions
  # GET /transactions.json
  def index
    @transactions = Transaction.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @transactions }
    end
  end

  # GET /transactions/1
  # GET /transactions/1.json
  def show
    @transaction = Transaction.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @transaction }
    end
  end

  # GET /transactions/new
  # GET /transactions/new.json
  def new
    @transaction = Transaction.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @transaction }
    end
  end

  # GET /transactions/1/edit
  def edit
    @transaction = Transaction.find(params[:id])
  end

  # POST /transactions
  # POST /transactions.json
  def create
    params[:transaction][:sale] = Sale.find params[:transaction][:sale]
    @transaction = Transaction.new(params[:transaction])
```

```
    respond_to do |format|
      if @transaction.save
        format.html { redirect_to edit_sale_path(@transaction.sale), notice: 'Transaction was
        format.json { render json: @transaction, status: :created, location: @transaction }
      else
        @sale = @transaction.sale
        format.html { render :action => "../sales/checking_out" }
        format.json { render json: @transaction.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /transactions/1
  # PUT /transactions/1.json
  def update
    @transaction = Transaction.find(params[:id])

    respond_to do |format|
      if @transaction.update_attributes(params[:transaction])
        format.html { redirect_to @transaction, notice: 'Transaction was successfully updated
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @transaction.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /transactions/1
  # DELETE /transactions/1.json
  def destroy
    @transaction = Transaction.find(params[:id])
    @transaction.destroy

    respond_to do |format|
      format.html { redirect_to transactions_url }
      format.json { head :no_content }
    end
  end
end
```

11.1.25 UsersController

```
class UsersController < ApplicationController
  def members_index
    @users = User.all
    respond_to do |format|
      format.html
      format.json { render json: @users }
    end
```

```
  end

  def staff_index
    @users = User.where(:role  => ['Owner', 'Manager', 'Stock Control', 'Cashier'])

    respond_to do |format|
      format.html
      format.json { render json: @users }
    end
  end

  def edit
    @user = User.find(params[:id])
    respond_to do |format|
      format.html
      format.json { render json: @user }
    end
  end

  def update
    @user = User.find(params[:id])
    @user.role = params[:role]
    @user.discount = params[:discount]
    @user.membership = params[:membership]

    respond_to do |format|
      if @user.save
        format.html {
          flash[:notice] = 'User was successfully updated.'
          render :edit
        }
        format.json { render json: @user }
      else
        format.html { render action: "edit" }
      end
    end
  end
end
```

11.1.26

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>POS - <%= yield(:title) %></title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="description" content="">
   <meta name="author" content="">

   <!-- Le styles -->
```

```
    <%= stylesheet_link_tag "bootstrap.min" %>
    <%= stylesheet_link_tag "bootstrap-responsive.min" %>
    <%= stylesheet_link_tag "main" %>


    <%= csrf_meta_tags %>

    <!-- Le HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->

    <!-- Le fav and touch icons -->
    <link rel="shortcut icon" href="/assets/favicon.ico">
    <link href="/assets/icon.png" rel="icon" type="image/png"/>
</head>

<body>

    <div class="navbar navbar-fixed-top">
        <div class="navbar-inner">
            <div class="container">
                <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </a>
                <a class="brand" href="/">POS</a>


                <% if user_signed_in? %>
                <div class="btn-group pull-right">
                    <a class="btn dropdown-toggle" data-toggle="dropdown" href="#">
                        <i class="icon-user"></i> <%= current_user.email %>
                        <i class="caret"></i>
                    </a>
                    <ul class="dropdown-menu no-collapse" style="margin-top:8px;">
                        <li><a href="/users/edit">My Account</a></li>
                        <li><a href="#">My Purchase History</a></li>
                        <li class="divider"></li>
                        <li><a href="/logout">Sign Out</a></li>
                    </ul>
                </div>
                <% else %>
                <div class="btn-group pull-right">
                    <a class="btn dropdown-toggle" data-toggle="dropdown" href="#">
                        Sign In
                        <i class="caret"></i>
                    </a>
                    <div class="dropdown-menu no-collapse" style="padding: 15px;padding-bottom:0px
                        <form action="/login" method="post" accept-charset="UTF-8">
                            <input id="username" style="margin-bottom: 15px;" type="text" name="user
```

```
            <input id="password" style="margin-bottom: 15px;" type="password" name="
            <div class="btn-group">
               <input class="btn btn-primary" style="width: 100%; height: 32px; font
            </div>
         </form>
      </div>
   </div>
   <% end %>


   <div class="nav-collapse">
      <ul class="nav">
         <li>
            <a href="/">Home</a>
         </li>
         <% if user_signed_in? %>
         <% if current_user.can_checkout %>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown">Sales<b class
            <ul class="dropdown-menu">
               <li><%= link_to 'New Sale', new_sale_path %></li>
               <li><%= link_to 'Previous Sales', sales_path %></li>
               <li><%= link_to 'Refunds', refunds_path %></li>
            </ul>
         </li>
         <% end %>
         <% if current_user.can_manage_stock %>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown">Stock Control
            <ul class="dropdown-menu">

               <li><%= link_to "Stock Levels", stock_levels_path %></li>
               <li><%= link_to "Products", products_path %></li>
               <li><%= link_to "Suppliers", suppliers_path %></li>
               <li><%= link_to "Stock Locations", stock_locations_path %></li>
               <li><%= link_to "Stock Transfers", stock_transfers_path %></li>
               <li><%= link_to "Supplier Stock Orders", supplier_stock_orders_path %
            </ul>
         </li>
         <% end %>
         <% if current_user.can_report %>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown">Reports<b cla
            <ul class="dropdown-menu">
               <li><a href="/reports">Index</a></li>
               <li><a href="/reports/sale">Sales</a></li>
               <li><a href="/reports/supplier">Suppliers</a></li>
               <li><a href="/reports/staff">Staff</a></li>
               <li><a href="/reports/stock">Stock</a></li>
               <li><a href="/reports/financial">Financial</a></li>
               <li><a href="/reports/customer">Customer</a></li>
            </ul>
```

```
                    </li>
                    <% end %>
                    <% if current_user.can_checkout %>
                    <li class="dropdown">
                       <a href="#" class="dropdown-toggle" data-toggle="dropdown">User Manageme
                       <ul class="dropdown-menu">
                          <li><a href="/members">Customer Memberships</a></li>
                          <li><a href="/staff">Staff Management</a></li>
                       </ul>
                    </li>
                    <% end %>
                    <% end %>
                    <li>
                       <a href="/help">Help</a>
                    </li>
                 </ul>
              </div><!--/.nav-collapse -->
           </div>
        </div>
     </div>

     <div class="wrapper">
        <div class="container">
           <center>
              <div class="global-flash">
                 <% if notice %>
                 <div class="alert alert-info"><h1>Notice</h1><p><%= notice %></p></div>
                 <% end %>
                 <% if alert %>
                 <div class="alert alert-error"><h1>Error</h1><p><%= alert %></p></div>
                 <% end %>
              </div>
           </center>
           <%= yield %>
        </div>
        <div class="push"><!--//--></div>
     </div><!-- /container -->

     <!-- Le javascript
     ================================================== -->
     <!-- Placed at the end of the document so the pages load faster -->
     <%= javascript_include_tag "application" %>


  </body>
  </html>
```