

Invenire Stock Inventory Manager

Web based ERP software for managing stock inventory
with advanced POS system

Developer manual

copyright 2015 syncrypts. All rights reserved.

Index of Contents

- Installation
- Source code structure
- Source code description

Installation Manual

- Upload the downloaded zip file to your server in the public_html directory.
- Extract the zip file
- Create a new database from your server mysql.
- Create user to the database and link the database to the user.
- Open the file database.php from the directory /application/config/database.php
- Fill up these informations with your database hostname, database username, database password, database name respectively which you have created in the previous step.

```
'hostname' => '',  
'username' => '',  
'password' => '',  
'database' => '',
```

- Now from server phpmyadmin go to your database. Select import and choose the file install.sql located in uploads/install.sql
- And you are ready to go now to browse the application.
- Default admin credentials

Email: admin@example.com

Password: 1234

THE WHOLE SYSTEM HAS BEEN REDESIGNED. YOU WILL NEED TO RESTORE THE SYSTEM

Source code structure

The application is developed on Codeigniter framework and completely follows MVC. The details of this framework can be found in <http://www.codeigniter.com/>

The root directory contains

- application

contains the core files of the application

- assets

- contains all css styles and javascript files
- system
 - contains all the configurations and library files of the framework
- uploads
 - admin_image
 - contains the profile image uploaded by admin himself
 - customer_image

```

55 function customer($param1 = '', $param2 = '', $param3 = '')
56 {
57     if ($this->session->userdata('admin_login') != 1)
58         redirect(base_url() . 'index.php?login');
59
60     if ($param1 == 'create')
61     {
62         $data['customer_code'] = $this->input->post('customer_code');
63         $data['name'] = $this->input->post('name');
64         $data['email'] = $this->input->post('email');
65         $data['password'] = sha1($this->input->post('password'));
66         $data['phone'] = $this->input->post('phone');
67         $data['address'] = $this->input->post('address');
68         $data['gender'] = $this->input->post('gender');
69         $data['age'] = $this->input->post('age');
70
71         $this->db->insert('customer' , $data);
72
73         // mail sending configuration
74         $password_unhashed = $this->input->post('password');
75         $email_to = $data['email'];
76
77         // upload a image for customer
78         $customer_id = mysql_insert_id();
79         move_uploaded_file($_FILES['userfile']['tmp_name'], 'uploads/customer_image/' . $customer_id . '.jpg');
80         $this->session->set_flashdata('flash_message' , 'New customer added successfully');
81         // send the mail to customer
82         $this->email_model->account_opening_email('customer' , $email_to , $password_unhashed);
83         redirect(base_url() . 'index.php?admin/customer' , 'refresh');
84     }

```

- contains the uploaded images of customers
- product_image
 - contains the images of products added
- product_category_image
 - contains the images of product category
- product_sub_category_image
 - contains the images of product sub-category
- supplier_image
 - contains of the images of the supplier
- index.php
 - this file on loading calls all the codeigniter functions

application

- config
 - contains the configuration files of the application
- controllers
 - Admin.php
 - Customer.php
 - Employee.php
 - Login.php

- Modal.php
- Multilanguage.php
- helpers
- libraries
- models
 - Barcode_model.php
 - Crud_model.php
 - Email_model.php
- views
 - admin
 - contains all the view files of admin panel
 - customer
 - contains all the view files of customer panel
 - employee
 - contains all the view files of employee panel
 - footer.php
 - forgot_password.php
 - header.php
 - includes_bottom.php
 - contains links to javascript files
 - includes_top.php
 - contains links to css files
 - index.html
 - index.php
 - login.php
 - the login page
 - modal.php

Source code description

```

21
22 // ADMIN DASHBOARD
23 function dashboard()
24 {
25     if ($this->session->userdata('admin_login') != 1)
26         redirect(base_url() . 'index.php?login');
27
28     $page_data['page_name']    = 'dashboard';
29     $page_data['page_title']  = 'Dashboard';
30     $this->load->view('index' , $page_data);
31 }
32

```

Admin Controller(/application/controllers/Admin.php)

Admin dashboard:

dashboard function loads the view file named dashboard.php and redirects to login page if the admin login is not true.

Manage customers

customer function contains the logical expressions of creating a new customer, editing the informations about an existing customer and deleting customer informations from the database. The parameters define the form action in which the data is to be created or edited. Email is sent to the new customer created which is defined in /application/models/email_model.php

```
// new account opening email
function account_opening_email($account_type , $email_to , $password_unhashed) {
    $system_url = base_url();

    $email_sub      = "New Account Opening";
    $email_to       = $email_to;
    $email_msg      = "Congratulations! Your account has been created.<br \>";
    $email_msg      .= "Your password for new account is ".$password_unhashed. "<br \>";
    $email_msg      .= "Login here " . $system_url . " to continue. <br \>";
    $email_msg      .= "You can change your password after logging in from your profile settings.";

    $this->do_email($email_msg , $email_sub , $email_to);
}
```

Manage suppliers

function supplier holds similar code logical expressions for adding, editing and deleting suppliers. Suppliers will have no login and so no email is sent to the supplier on creating his/her account.

Manage product category

the function product_category also holds the same logical expressions for creating new product category, editing an existing one and deleting some category from the database.

Manage product sub category

the function product_sub_category also holds the same logical expressions for creating new product sub-category, editing an existing one and deleting some sub-category from the database. Each sub-category is linked to a specific category.

Manage products

```
266
267 function product($param1 = '' , $param2 = '' , $param3 = '')
268 {
269     if ($this->session->userdata('admin_login') != 1)
270         redirect(base_url() . 'index.php?login');
271
272     if ($param1 == 'create')
273     {
274         $data['serial_number'] = $this->input->post('serial_number');
275         $data['category_id'] = $this->input->post('category_id');
276         $data['sub_category_id'] = $this->input->post('sub_category_id');
277         $data['name'] = $this->input->post('name');
278         $data['purchase_price'] = $this->input->post('purchase_price');
279         $data['selling_price'] = $this->input->post('selling_price');
280         $data['note'] = $this->input->post('note');
281
282         $this->db->insert('product' , $data);
283         // upload image
284         $product_id = mysql_insert_id();
285         move_uploaded_file($_FILES['userfile']['tmp_name'], 'uploads/product_image/' . $product_id . '.jpg');
286         $this->session->set_flashdata('flash_message' , 'New product added successfully');
287         redirect(base_url() . 'index.php?admin/product' , 'refresh');
288     }
289 }
```

function product holds all the logics for creating a new product with estimated purchase price and selling price. The purchase price is later updated in another function on creation of a purchase. Product image is saved to /uploads/product_image and they are named according to product_id. Parameters are used for defining the form of addition on edition.

Product barcode

```
327
328 function product_barcode($param1 = '' , $serial_number = '')
329 {
330     if ($this->session->userdata('admin_login') != 1)
331         redirect(base_url() . 'index.php?login');
332
333     if ($param1 == 'create_barcode')
334     {
335         $this->barcode_model->create_barcode($serial_number);
336     }
337
338     $page_data['page_name']      = 'product_barcode';
339     $page_data['page_title']     = 'Product Barcodes';
340     $page_data['products']      = $this->db->get('product')->result_array();
341     $this->load->view('index' , $page_data);
342 }
343
```

Each and every product is differentiated by software generated barcode and the barcode is generated according to a random serial number of products. Barcode model is used for creating the barcodes. The create_barcode function of barcode_model generates barcode and from controller the \$serial_number is passed to the model.

```
25 // Don't forget to sanitize user inputs
26 $text = $serial_number;
27
28 // The arguments are R, G, B for color.
29 $color_black = new BCGColor(0, 0, 0);
30 $color_white = new BCGColor(255, 255, 255);
31
32 $drawException = null;
33 try {
34     $code = new BCGcode39();
35     $code->setScale(2); // Resolution
36     $code->setThickness(30); // Thickness
37     $code->setForegroundColor($color_black); // Color of bars
38     $code->setBackgroundColor($color_white); // Color of spaces
39     $code->setFont($font); // Font (or 0)
40     $code->parse($text); // Text
41 } catch(Exception $exception) {
42     $drawException = $exception;
43 }
44
45 /* Here is the list of the arguments
46 1 - Filename (empty : display on screen)
47 2 - Background color */
48 $drawing = new BCGDrawing('', $color_white);
49 if($drawException) {
50     $drawing->drawException($drawException);
51 } else {
52     $drawing->setBarcode($code);
53     $drawing->draw();
54 }
55
56 // Header that says it is an image (remove it if you save the barcode to a file)
57 header('Content-Type: image/png');
58 header('Content-Disposition: inline; filename="barcode.png"');
59
60 // Draw (or save) the image into PNG format.
61 $drawing->finish(BCGDrawing::IMG_FORMAT_PNG);
62 }
63
```


Manage damaged products

function `damaged_products` contains the logical expressions to add a product to damaged product list. Also it declares the codes for editing and deleting damaged product from database.

Manage orders

```
function order($param1 = '' , $param2 = '' , $param3 = '')
{
    if ($this->session->userdata('admin_login') != 1)
        redirect(base_url() . 'index.php?login');

    if ($param1 == 'create')
    {
        $data['order_number']      = $this->input->post('order_number');
        $data['customer_id']      = $this->input->post('customer_id');
        $data['product_id']      = $this->input->post('product_id');
        $data['quantity']        = $this->input->post('quantity');
        $data['shipping_address'] = $this->input->post('shipping_address');
        $data['order_status']     = $this->input->post('order_status');
        $data['payment_status']   = $this->input->post('payment_status');
        $data['note']             = $this->input->post('note');
        $data['timestamp']        = strtotime($this->input->post('timestamp'));

        $status = $this->input->post('order_status');
        $this->db->insert('order' , $data);
        // mail sending configuration
        $order_number      = $data['order_number'];
        $email_to          = $this->db->get_where('customer' , array('customer_id' => $data['customer_id']))->row()->email;
        $product_ordered   = $this->db->get_where('product' , array('product_id' => $data['product_id']))->row()->name;
        $product_quantity  = $data['quantity'];
```

function `order` creates an order under a customer name and sends mail to the customer about the order created for him/her.

multiple products with any quantity can be added to an order. Admin can set order and payment status for that specific order. Admin can also take payment for that order. Payment option is now with recurring payment under same order invoice so admin can take a particular payment in multiple time until the due is zero.

Manage Purchases

```
function purchase_add($param1 = ' ' , $param2 = ' ')  
{  
    if ($this->session->userdata('admin_login') != 1)  
        redirect(base_url() . 'index.php?login');  
  
    if ($param1 == 'create')  
    {  
        $this->crud_model->new_purchase();  
        redirect(base_url() . 'index.php?admin/purchase_add');  
    }  
  
    $page_data['page_name']      = 'purchase_add';  
    $page_data['page_title']     = 'New Purchase';  
    $this->load->view('index' , $page_data);  
}
```

The function `purchase_add` can create new purchase and the purchase process starts when the `create` parameter is true and it calls `new_purchase` function from `/application/model/Crud_model.php`

```
// DECLARATION: CREATE A NEW PURCHASE  
function new_purchase()  
{  
    $data['purchase_code'] = $this->input->post('purchase_code');  
    $data['supplier_id']   = $this->input->post('supplier_id');  
    $data['timestamp']     = strtotime($this->input->post('timestamp'));  
    $purchase_entries     = array();  
    $product_ids          = $this->input->post('product_id');  
    $quantities            = $this->input->post('quantity');  
    $purchase_prices       = $this->input->post('purchase_price');  
    $number_of_entries    = sizeof($product_ids);  
    for ($i = 0; $i < $number_of_entries; $i++) {  
        if ($product_ids[$i] != "" && $quantities[$i] != "" && $purchase_prices[$i] != "") {  
            $new_purchase_entry = array(  
                'product_id' => $product_ids[$i],  
                'quantity'   => $quantities[$i],  
                'purchase_price' => $purchase_prices[$i]  
            );  
            array_push($purchase_entries , $new_purchase_entry);  
            // INCREASES THE PRODUCT QUANTITY IN STOCK  
            $this->db->where('product_id', $product_ids[$i]);  
            $this->db->set('stock_quantity', 'stock_quantity + ' . $quantities[$i], FALSE);  
            $this->db->update('product');  
        }  
    }  
    $data['purchase_entries'] = json_encode($purchase_entries);  
    $this->db->insert('purchase' , $data);  
    $purchase_id = $this->db->insert_id();  
    // CREATE PAYMENT ENTRY  
    $data2['purchase_id'] = $purchase_id;  
    $data2['amount']      = $this->input->post('amount');  
    $data2['method']      = $this->input->post('method');  
    $data2['type']        = 'expense';  
    $data2['timestamp']   = strtotime($this->input->post('timestamp'));  
    $data2['supplier_id'] = $this->input->post('supplier_id');  
    $this->db->insert('payment' , $data2);  
    return $purchase_id;  
}
```

the new_purchase function enters data to purchase informations. It enters data to payment informations and the purchase data is saved as expense for the company. It also updates the stock quantity of the products and also purchase price.

Manage Sales

```
// SALE PRODUCT
function sale_add($param1 = '')
{
    if ($this->session->userdata('admin_login') != 1)
        redirect(base_url() . 'index.php?login');

    if ($param1 == 'do_add')
    {
        $this->crud_model->add_new_sale();
        $this->session->set_flashdata('flash_message' , 'New sale added successfully');
        redirect(base_url() . 'index.php?admin/sale_add');
    }

    $page_data['page_name']      = 'sale_add';
    $page_data['page_title']     = 'Create a new sale';
    $this->load->view('index' , $page_data);
}
```

function sale_add creates new sales associating with add_new_sale function of /application/models/crud_model.php

```
function add_new_sale()
{
    // CREATE SALE ENTRY

    $data['invoice_code']      = $this->input->post('invoice_code');
    $data['discount_percentage'] = $this->input->post('discount_percentage');
    $data['vat_percentage']    = $this->input->post('vat_percentage');
    $data['customer_id']       = $this->input->post('customer_id');
    $data['timestamp']         = strtotime(date("Y-m-d H:i:s"));

    $invoice_entries          = array();
    $product_ids              = $this->input->post('product_id');
    $total_numbers             = $this->input->post('total_number');
    $selling_prices            = $this->input->post('selling_price');

    $number_of_entries = sizeof($product_ids);
    for ($i = 0; $i < $number_of_entries; $i++) {
        if ($product_ids[$i] != "" && $total_numbers[$i] != "" && $selling_prices[$i] != "") {
            $new_entry = array('product_id' => $product_ids[$i], 'total_number' => $total_numbers[$i], 'selling_price' => $selling_prices[$i]);
            array_push($invoice_entries, $new_entry);
        }
    }
    $data['invoice_entries'] = json_encode($invoice_entries);

    $this->db->insert('invoice' , $data);
    $invoice_id = $this->db->insert_id();

    // CREATE PAYMENT ENTRY
    $data2['invoice_id']      = $invoice_id;
    $data2['amount']          = $this->input->post('amount');
    $data2['method']          = $this->input->post('method');
    $data2['type']            = 'income';
    $data2['timestamp']       = strtotime(date("Y-m-d H:i:s"));
    $data2['customer_id']     = $this->input->post('customer_id');

    $this->db->insert('payment' , $data2);
}
```

the sale entry is taken as json data and enters informations to sale invoices and payment. Each sale is counted as income.

function get_selected_product holds the logical expressions for sending a product to sale invoice during a sale.

```
function get_selected_product($input_type , $input_id , $total_number)
{
    if ($input_type == 'mouse')
    {
        $product_info = $this->db->get_where('product' , array('product_id' => $input_id))->row();
    }
    else if ($input_type == 'barcode')
    {
        $product_info = $this->db->get_where('product' , array('serial_number' => $input_id))->row();
    }
    if (sizeof($product_info) == 0)echo '';

    echo '
    <tr id="entry_row_'. $total_number .'>
        <td id="serial_number_'. $total_number .'>'. $total_number .'</td>
        <td>'. $product_info->serial_number .'</td>
        <td>
            '. $product_info->name .'
            <input type="hidden" name="product_id[]" value="'. $product_info->product_id ." size="3" style="width:50px;"
            id="single_entry_product_id_'. $total_number .'>
        </td>
        <td>
            <input type="number" name="total_number[]" value="1" size="3" style="width:50px;"
            id="single_entry_quantity_'. $total_number ." max="4"
            onkeyup="calculate_single_entry_total(' . $total_number .')>
        </td>
        <td>
            <input type="number" name="selling_price[]" value="'. $product_info->selling_price ."
            id="single_entry_selling_price_'. $total_number ."
            onkeyup="calculate_single_entry_total(' . $total_number .')>
        </td>
        <td id="single_entry_total_'. $total_number .'>'. $product_info->selling_price .'</td>
        <td>
            <i class="fa fa-trash-o" onclick="remove_row(' . $total_number .')"
            id="delete_button_'. $total_number ." style="cursor:pointer;"></i>
        </td>
    </tr>';
}
```

Manage Employee

function employee can create, edit and delete employees. A email is sent to the employee when admin creates the employee account.

Employees are categorised into two classes. One is sales staff and another is purchase staff.

```
function employee($param1 = '' , $param2 = '' , $param3 = '')
{
    if ($this->session->userdata('admin_login') != 1)
        redirect(base_url() . 'index.php?login');

    if ($param1 == 'create')
    {
        $data['name']      = $this->input->post('name');
        $data['email']     = $this->input->post('email');
        $data['password']  = sha1($this->input->post('password'));
        $data['type']      = $this->input->post('type');

        $this->db->insert('employee' , $data);
        // mail sending configuration
        $email_to          = $data['email'];
        $password_unhashed = $this->input->post('password');

        // upload an image
        $employee_id = mysql_insert_id();
        move_uploaded_file($_FILES['userfile']['tmp_name'], 'uploads/employee_image/' . $employee_id . '.jpg');
        $this->session->set_flashdata('flash_message' , 'New employee added successfully');
        // send mail to employee
        $this->email_model->account_opening_email('employee' , $email_to , $password_unhashed);
        redirect(base_url() . 'index.php?admin/employee' , 'refresh');
    }
}
```

Private Messaging

```
function message_new($param1 = '' , $param2 = '')
{
    if ($this->session->userdata('admin_login') != 1)
        redirect(base_url() . 'index.php?login');

    if ($param1 == 'send_new_message') {

        $new_message_thread_code = $this->crud_model->send_new_message();
        $this->session->set_flashdata('flash_message' , 'Message Sent');
        // mail sending configurations
        $get_receiver = $this->db->get_where('message_thread' , array('message_thread_code' => $new_message_thread_code))->row()->receiver;
        $receiver      = explode('-', $get_receiver);
        $user_to_email_type = $receiver[0];
        $user_to_email_id = $receiver[1];
        $email_to         = $this->db->get_where($user_to_email_type , array($user_to_email_type.'_id' => $user_to_email_id))->row()->email;
        // send the mail to the receiver
        $this->email_model->message_notification_email_sender_admin($email_to);
        redirect(base_url() . 'index.php?admin/message_read/' . $new_message_thread_code , 'refresh');
    }

    $page_data['page_name']      = 'message_new';
    $page_data['page_title']     = 'Messaging';
    $page_data['customers']     = $this->db->get('customer')->result_array();
    $page_data['sales_staff']    = $this->db->get_where('employee' , array('type' => 1))->result_array();
    $page_data['purchase_staff'] = $this->db->get_where('employee' , array('type' => 2))->result_array();
    $page_data['message_thread_code'] = $param2;
    $this->load->view('index' , $page_data);
}
```

function message_new holds the logical expressions for sending the message to any user by the admin. An email notification is send to the receiver every time when admin sends message or make a reply to their message. Messages are tracked by message thread code which is defined in

the send_new_message function located in /application/models/

```
function send_new_message() {  
  
    $message_body      = $this->input->post('message_body');  
    $receiver          = $this->input->post('receiver');  
    $sender            = $this->session->userdata('login_type') . '-' . $this->session->userdata('user_id');  
  
    $query1 = $this->db->get_where('message_thread' , array('sender' => $sender , 'receiver' => $receiver))>num_rows();  
    $query2 = $this->db->get_where('message_thread' , array('sender' => $receiver , 'receiver' => $sender))>num_rows();  
  
    if ($query1 == 0 && $query2 == 0) {  
  
        $message_thread_code = substr(md5(rand(1000000000, 20000000000)), 0, 15);  
  
        $data_message_thread['message_thread_code'] = $message_thread_code;  
        $data_message_thread['sender']             = $sender;  
        $data_message_thread['receiver']           = $receiver;  
        $this->db->insert('message_thread' , $data_message_thread);  
    }  
  
    if ($query1 > 0)  
        $message_thread_code = $this->db->get_where('message_thread', array('sender' => $sender, 'receiver' => $receiver))>row()->message_thread_code;  
    if ($query2 > 0)  
        $message_thread_code = $this->db->get_where('message_thread', array('sender' => $receiver, 'receiver' => $sender))>row()->message_thread_code;  
  
    $timestamp = strtotime(date("Y-m-d H:i:s"));  
  
    $data_message['message_thread_code'] = $message_thread_code;  
    $data_message['message_body']       = $message_body;  
    $data_message['sender']              = $sender;  
    $data_message['timestamp']           = $timestamp;  
    $this->db->insert('message' , $data_message);  
  
    return $message_thread_code;  
}
```

crud_model.php.

Profile Settings

function profile_settings holds the logical expressions for

- updating current profile informations.
- uploading a profile picture.
- updating or changing login password.

System settings

function settings holds the logical expressions for

- updating system informations like company name or email and other system informations.
- uploading a company logo.

Customer Controller(/application/controllers/customer.php)

Manage orders

function order holds the logical expressions for

- creating a new order by logged in customer
- the order is then sent to the admin for approval
- mail is sent to the admin about the order

Private Messaging

function message_new holds the logical expressions for

- sending a new message to admin
- a mail is sent to the admin when customers send admin any message or replies.

Profile settings

function profile_settings holds the logical expressions for

- updating profile informations
- uploading profile picture
- changing or updating the login password.

Employee Controller(/application/controllers/employee.php)

Manage Sales (only when a sales staff logs in)

function sale_add holds the logical expressions for creating new sale.

Manage Purchase (only when a purchase staff logs in)

function purchase_add holds the logical expressions for creating new purchase.

Private Messaging

function message_new holds the logical expressions for

- sending a new message to admin
- a mail is sent to the admin when employees send admin any message or replies.

Profile settings

function profile_settings holds the logical expressions for

- updating profile informations
- uploading profile picture
- changing or updating the login password.

Login Controller(/application/controllers/login.php)

function do_login holds the logical expressions for

- matching email and password for different user.
- setting the login type as admin or customer or employee.
- setting session data which are used for other views

function reset_password

- matches email address to saved email address
- generates a new password
- sends email to that address with the newly generated password

function logout

- unsets the session and destroys all the session data saved for that session.

The Model Files (/application/models)

models

- barcode_model.php
contains the logical expressions for generating barcodes for products.
- crud_model.php
contains the basic functions for create, retrieve, update and delete which works in association of controllers or views.
- email_model.php
contains the functions that send email on different events.

The View Files (/application/views)

views

- admin
 - contains all the view files of admin panel
- customer
 - contains all the view files of customer panel
- employee
 - contains all the view files of employee panel
- footer.php
 - contains the footer informations of the application
- forgot_password.php
 - page for resetting password by email
- header.php
 - contains the header view of the application
- includes_bottom.php
 - contains links to javascript files
- includes_top.php
 - contains links to css files
- index.html

- index.php
 - includes the footer, header, javascript, css, modals at the time of page loading.
- login.php
 - the login page
- modal.php
 - contains the modal views which are used in the view of different panels.