# Design Requirements
# Project:
# Cafeteria Management System:
# Reslove

T-RISE
Rendani Dau (13381467)
Elana Kuun (12029522)
Semaka Malapane (13081129)
Antonia Michael (13014171)
Isabel Nel (13070305)

August 28, 2015

# Contents

| | |
|---|---|
| **Document Title** | Design Requirements Document |
| **Document Identification** | Document 0.0.2 |
| **Author** | Rendani Dau, Elana Kuun, Semaka Malapane, Antonia Michael, Isabel Nel |
| **Version** | 0.0.2 |
| **Document Status** | Second Version - added the function templates for the client controllers |

| Version | Date | Summary | Authors |
|---|---|---|---|
| 0.0.1 | 29 May 2015 | First draft contains first two use cases template methods and declarations | Rendani Dau, Elana Kuun, Semaka Malapane, Antonia Michael Isabel Nel, |
| 0.0.2 | 27 August 2015 | Second draft adding all the client and controllers for the updated system | Rendani Dau, Elana Kuun, Semaka Malapane, Antonia Michael Isabel Nel |

# 1 Introduction

This document contains the functional requirements specification, architecture requirements and testing for the Resolve Cafeteria Management System that will be created for Software Engineering (COS 301) at the University of Pretoria 2015, by the group T-RISE. In this document we will thoroughly discuss and layout the project's design requirements to provide a clear view of the system as a whole. An agile method is being followed so the following document focusses on the PlaceOrder and ManageProfile modules.

# 2 Vision

The vision of this project is to implement a flexible, pluggable, fully functional software application that will be maintainable, with detailed supporting documentation and an instruction manual for the Cafeteria Management System. This system will assist in managing the cafeteria's inventory/stock, executing orders from the cafeteria, generating bills and sending these to the appropriate parties and facilitating payments for access cards (or the use of unique access card numbers).

# 3 Background

As specified in the project proposal document from Resolve - the cafeteria is currently cash only and does not accept bank cards or electronic payments. This makes it inconvenient for employees as they have to carry around cash if they want to purchase anything from the cafeteria. Hence, this is equivalent to purchasing from an external food outlet where they can also pay with their preferred method of payment. The employees have to hence use up fuel and time and lastly this does not bring in the maximum amount of income to the cafeteria, hindering its growth and improvement.

Resolve is therefore looking for a means to accept payments from employees for the canteen using their employee access cards or access card numbers, with an amount being deducted from their salary at the end of the month.

Resolve proposed the Cafeteria Management System to assist with this problem. After our first meeting with the client, they brought to our attention that at times the cafeteria does not even have enough stock to provide some of the menu items, thus the managing of inventory or stock will also be part of the system. The system will also predict what inventory/stock

needs to be bought for the next week in order to avoid such a problem. At the end of each month, the bill for the month will be sent to either payroll or to the employee. This option is configurable from the user's profile. The employee can also set a spending limit for each month for control purposes. The system will have its own maximum, such that users cannot set a limit that exeeds this.

# 4 Standards and conventions

## 4.1 Design standards

The diagrams are designed and created using UML. The main use case of the system is decomposed into components.

# 5 Authentication

## 5.1 authentication.client.controller

The authentication controller allows existing users to log in and new users to sign up.

1. Declaration:
   angular.module('users').controller('AuthenticationController', ['$scope', '$http', '$location', '$cookies', 'Authentication', function($scope, $http, $location, $cookies, Authentication)

2. Methods:

   - Sign up
     $scope.signup = function()

     -Usage
     Method that allows a new user to sign up.
   - Sign in
     $scope.signin = function()

     -Usage
     Method that allows an existing user to sign in.

## 5.2 users.authentication.server.controller

The authentication controller allows existing users to log in and new users to sign up. It also creates a new super user or admin if eitehr one does not exist, and it sets the system wide limit to 5000.

1. Methods:

   - Sign up
     exports.signUp = function(JsonOBject req, JsonOBject res)

     -Usage
     This method ensures that all the fields neccessary to sign up were completed correctly. It returns an error message if one or more of the fields were entered incorrectly, otherwhise it adds the user deatils to the database.
   - Sign in
     exports.signin = function(JsonOBject req, JsonOBject res, function next)

     -Usage
     This method allows an existing user to sign it if the user entered his/her details correctly. It performs a check to see if the user id and the passoword is correct. If the details are incorrect an error message is displayed.
   - Sign out
     exports.signout = function(JsonOBject req, JsonOBject res)

     -Usage
     This method lets a user sign out of the system.
   - Check super user
     exports.checkSuperUser = function()

     -Usage
     This method performs a check to confirm whether there is an existing super user or not. If there is no super user, it creates a super user and admin user, and sets the system wide limit to 5000. If these actions were not completed successfully the method displays an error message. This method also checks whether there is an existing admin user and creates one if there is not currently and admin user. An error message is displayed if this action was not performed successfully.

# 6 Manage Profile

## 6.1 superuser.client.controller.js

1. Declaration:
   angular.module('users').controller('superuserController', ['$scope', '$http', '$location', '$window', 'Users', 'Authentication', function($scope, $http, $location, $window, Users, Authentication) { }

2. Methods:

   - Assign roles
     $scope.assignRoles = function(Boolean isValid) {}

     Usage: Superuser can assign cashier, cafeteria manager, finance manager and admin roles
   - Assign roles admin role
     $scope.assignRolesAdminRole = function(Boolean isValid) { }

     Usage: Admin user also has access to the assign roles functionality and serves as a back up superuser
   - Change employee ID
     $scope.changeEmployeeID = function(Boolean isValid) {}

     Usage: The superuser can change the employee ID of the users if the user signed up with the incorrect ID or if the company changes the IDs.
   - Remove employee
     $scope.removeEmployee = function(Boolean isValid) {}

     Usage: The superuser is also able to remove users from the system, due to resignation or dismissal for example
   - Search employee
     $scope.searchEmployee = function(Boolean isValid) { }

     Usage: This function is used to retrieve employees from the system to be able to change their employe ID or remove them from the system
   - Search employee ID
     $scope.searchEmployeeID = function(int row) {}

Usage: This function is used to retrieve employee IDs

- Set system wide limit
  $scope.setSystemWideLimit = function(Boolean isValid){ }

  Usage: This is used by the superuser to set the maximum monthly spending limit for all the users of the system.

- Set canteen name
  $scope.setCanteenName = function(Boolean isValid){ }

  Usage: The canteen name is configurable from the superuser's branding settings page.

- Check user
  $scope.checkUser = function(){}

  Usage: This is a security function that checks to make sure that the authorized superuser is accessing the page and if this is not the case, the user will be redirected to the home page

- Load employees
  $scope.loadEmployees = function(){}

  Usage: This function is used to dynamically populate the drop down menu for the change employee ID functionality

## 6.2 cashier.client.controller.js

1. Declaration:
   angular.module('users').controller('cashierController', ['$scope', '$http', '$stateParams', '$location', 'Authentication', function($scope, $http, $stateParams, $location, Authentication)

2. Methods:

   - Get orders
     $scope.getOrders = function()

     Usage: Function to obtain the list of orders that an order placed from the menu

   - Mark as ready
     $scope.markAsReady = function(String username, String order-

Number)

Usage: Function that the cashier uses to send a notification to the user notifying the user that their order is ready. The function also changes the status in the model from open to ready.

- Mark as collected
$scope.markAsCollected = function(String username, String itemName, String orderNumber)

Usage: On the click of the 'Order collected' button on the cashier page, this function will be called. It changes the status of the order in the model from ready/open to closed.

- Mark as paid
$scope.markAsPaid = function(String username, String itemName, String orderNumber)

Usage: On the click of the 'Paid and collected' button on the cashier page, this function will be called. It changes the status of the order in the model from ready/open to closed.

- Check user
$scope.checkUser = function()

Usage: This is a security function that checks to make sure that the authorized cashier is accessing the page and if this is not the case, the user will be redirected to the home page

## 6.3 cashier.client.controller.js

1. Declaration: angular.module('users').controller('FinanceController', ['$scope', '$http', '$location', '$stateParams', 'Authentication', function($scope, $http, $location, $stateParams, Authentication)

2. Methods:

   - Get user orders
   $scope.getUserOrders = function()

   Usage: Function to obtain the orders placed by the users via adding to plate from the menu page and proceeding to place that order. The finance manager will enter the employee ID of the user

in the textbox displayed on the page and the orders placed by that user will be displayed on that page.

- Check user
  $scope.checkUser = function()

  Usage: This is a security function that checks to make sure that the authorized cashier is accessing the page and if this is not the case, the user will be redirected to the home page

## 6.4 users.profile.server.controller

The profile controller provides fuctionality to search for an employee, retrieve the system limit, and for a user to update his/her password.

1. Methods:

   - Update
     exports.update = function(JsonObject req, JsonObject res)

     -Usage
     This method allows a user to update his/her profile and displays an error message if the profile could not be updated.
   - Get system limit
     exports.getSystemLimit = function(JsonObject req, JsonObject res)

     -Usage
     This method retrieves the system limit, it is used to prevent an employee from setting their limit too high.
   - Search employee
     exports.searchEmployee=function(JsonObject req, JsonObject res)

     -Usage
     This method allows an employee to be searched. It returns the emplyee details if that employee is found, if the employee is not found it displays an error message.

## 6.5 settings.client.controller

The settings controller allows the user to view and edit profile information. The controller also perfomrs security checks to make sure that unauthorised

users cannot access certain functions of the system.

1. Declaration:
   angular.module('users').controller('SettingsController', ['$scope', '$http', '$location', 'Users', 'Authentication', function($scope, $http, $location, Users, Authentication)

2. Methods:

   - Update user profile
     $scope.updateUserProfile = function(Boolean isValid)

     -Usage
     This method allows the user to update profile information. It displays an error if any of the information entered by the user is not in the correct format, or if the limit set by the user exceeds the system limit.
   - Change user password
     $scope.chageUserPassword = function()

     -Usage
     This method lets the user change his/her password.
   - Get system limit
     $scope.getSystemLimit = function()

     -Usage
     This method retrieves the system wide limit.
   - Search employee
     $scope.searchEmployee = function(Boolean isValid)

     -Usage
     This method allows the superuser to search for an employee, for example when a new role needs to be assigned to the employee. This method is also used in finance where an employee is searched to display his/her bill.
   - Check user role
     $scope.checkuser = function()

     -Usage

This method performs an authentication check to ensure that unauthorised users cannot access certain features. For example, a normal user will not be able to navigate to the manage cafeteria page.

- Check if the user has a financial role
  $scope.checkFinUser = function()

  -Usage
  The checkFinUser method performs a check to establish wheter the user has a finance role or not. It prevents unauthorised users from performing actions that can only be done by users with a finance role.

## 6.6 password.client.controller

The password controller allows a user to reset his/her passoword if they forgot what their password is.

1. Declaration:
   angular.module('users').controller('PasswordController', ['$scope', '$stateParams', '$http', '$location', 'Authentication', function($scope, $stateParams, $http, $location, Authentication)

2. Methods:

   - Forgot password
     exports.forgot = function(JsonObject req, JsonObject res, function next)

     -Usage
     This method sends a link to the user who forgot his/her password that they can follow to reset the password. This link is sent to the user's email account. It displays an error message if the user could not be found, if the user id field is blank, or if an email could not be sent.

   - Validate reset token
     exports.validateResetToken = function(JsonObject req, JsonObject res)

     -Usage
     The method validates that the user can reset his/her password

with the link they followed. If the link is invalid it displays an error message.

- Reset
  exports.reset = function(JsonObject req, JsonObject res, function next)

  -Usage
  This method resets the user password if the link followed to reset his/her password is valid. It displays an error message if the user did not enter the same password twice or if the password reset token is invalid. If the password has been changed an email is sent to the user to notify him/her of the change.
- Change password
  exports.changePassword = function(JsonObject req, JsonObject res)

  -Usage
  This method allows the user to change his/her password. It displays an error message if the user could not be found or is not logged in, if the user did not enter the current password correctly, if the new password is not long enough, or if the user did not enter the same new password twice in order to confirm the new password.

# 7 Manage System

## 7.1 superuser.client.controller.js

1. Declaration:
   angular.module('users').controller('superuserController', ['$scope', '$http', '$location', '$window', 'Users', 'Authentication', function($scope, $http, $location, $window, Users, Authentication)
2. Methods:

   - Assign roles
     $scope.assignRoles = function(isValid)

     Usage: Superuser can assign cashier, cafeteria manager, finance manager and admin roles

- Assign roles admin role
  $scope.assignRolesAdminRole = function(isValid)

  Usage: Admin user also has access to the assign roles functionality and serves as a back up superuser
- Change employee ID
  $scope.changeEmployeeID = function(Boolean isValid)

  Usage: The superuser can change the employee ID of the users if the user signed up with the incorrect ID or if the company changes the IDs.
- Remove employee
  $scope.removeEmployee = function(Boolean isValid)

  Usage: The superuser is also able to remove users from the system, due to resignation or dismissal for example
- Search employee
  $scope.searchEmployee = function(Boolean isValid)

  Usage: This function is used to retrieve employees from the system to be able to change their employe ID or remove them from the system
- Search employee ID
  $scope.searchEmployeeID = function(row)

  Usage: This function is used to retrieve employee IDs
- Set system wide limit
  $scope.setSystemWideLimit = function(Boolean isValid)

  Usage: This is used by the superuser to set the maximum monthly spending limit for all the users of the system.
- Set canteen name
  $scope.setCanteenName = function(Boolean isValid)

  Usage: The canteen name is configurable from the superuser's branding settings page.
- Check user
  $scope.checkUser = function()

Usage: This is a security function that checks to make sure that the authorized superuser is accessing the page and if this is not the case, the user will be redirected to the home page

- Load employees
  $scope.loadEmployees = function()

  Usage: This function is used to dynamically populate the drop down menu for the change employee ID functionality

## 7.2 users.superuser.server.controller.js

1. Methods:

   - Assign roles
     exports.assignRoles = function(req, res)

     Usage: Superuser can assign cashier, cafeteria manager, finance manager and admin roles. It also changes the role of the users in the database/model.
   - Assign roles admin role
     exports.assignRolesAdminRole = function(req, res)

     Usage: Admin user also has access to the assign roles functionality and serves as a back up superuser. It also changes the role of the users in the database/model.
   - Change employee ID
     exports.changeEmployeeID = function(req, res)

     Usage: The superuser can change the employee ID of the users if the user signed up with the incorrect ID or if the company changes the IDs. This changes the employee ID of the user and saves the nnew one in the database.
   - Remove Employee
     exports.removeEmployee = function(req, res)

     Usage: The superuser is also able to remove users from the system, due to resignation or dismissal for example. The employee is then also removed from the database.
   - Send email
     function sendEmail(newLimit)

Usage: This is a helper function to mail all users of the system. It takes the new limit as a parameter and tells the users what the new limit is in the email.

- Set system wide limit
  exports.setSystemWideLimit = function(req, res)

  Usage: The superuser is able to change the monthly spending limit of the system here and it will save this value to the database.

# 8 Manage Cafeteria

## 8.1 menuitems.client.controller

The menuitems client controller handles all the functionality concerning the menu, such as adding and updating menu items.

1. Declaration:
   angular.module('menuItems').controller('MenuItemsController', ['$scope', '$rootScope','$http', '$stateParams', '$location', '$cookies', 'Authentication', 'MenuItems', function($scope, $rootScope, $http, $stateParams, $location, $cookies, Authentication, MenuItems)

2. Methods:

   - Toggle Collapsible Menu
     $scope.toggleCollapsibleMenu = function()
     -Usage
     This method collapses and opens the Menu Bar. The function is invoked on the click of a button in the HTML file.
   - Add Form Field
     $scope.addFormField = function()
     -Usage
     This method adds new text-boxes to assist with adding ingredients for menu items. The method is invoked when the button addIngridients is clicked in the HTML.
   - Load Ingridients
     $scope.loadIngredients = function()
     -Usage
     This method transfers ingredients from the request result array to

the working scope array. The method is invoked by other methods in the controller.

- Add More Ingridients Update
  $scope.addMoreIngredientsUpdate = function()
  -Usage
  This method adds new text-boxes to assist with updating ingredients for existing menu items. The method is invoked when the button addIngridients is clicked in the update ingredients section in the HTML.
- Remove Ingridient
  $scope.removeIngredient = function(int index)
  -Usage
  This method removes the ingredient at the specified index for the menu item being updated.
- Undo Remove Ingredient
  $scope.undoRemoveIngredient = function(index)
  -Usage
  This method undo's a remove ingredient operation performed by the above function.
- Update Menu Itemp
  $scope.updateMenuItem = function()
  -Usage
  This method sends the update request to the server to update the menu items. It will perform all error checks and display any errors returned from the server in the HTML.
- Search Menu
  $scope.searchMenu = function(Boolean isValid)
  -Usage
  This method searches for a menu item in the database. The search query is obtained from the scope variable.
- Create menu item
  $scope.createMenuItem = function(Boolean isValid)
  -Usage
  This method creates a new menu item in the database. The method will perform all error checks and validation and send the request to the server. The server response will then be displayed on the page.
- Create Menu Category
  $scope.createMenuCatagory = function(Boolean isValid)

-Usage
This method adds a new Menu item category to the database. The method will perform error checks and send the request to the server. The server response will be displayed on the page.

- Remove Ingredient
  $scope.removeIngredientOption = function(int index) -Usage
  Removes ingredients from the new menu item being added. Method is invoked on button click.
- Load Menu Items
  $scope.loadMenuItems = function() -Usage
  This method gets all the menu items from the server and populates the local array. If the server responds with an error, the error will be displayed on the page.
- Load Menu Categories
  $scope.loadMenuCategories = function()
  -Usage
  This method gets all the menu item categories from the server. If the server responds with an error, the error wil be displayed on the page.
- Check Stock
  $scope.checkStock = function(String menuItemName)
  -Usage
  This method checks if all the ingredients required to make a menu item are in stock and the quantities are enough. The item will then be marked appropriately as in-stock or out-of-stock.

# 9 Manage Inventory

## 9.1 inventory.client.controller

The inventory controller handles all the main functionality concerning inventory, such as adding an anventory item.

1. Declaration:
   angular.module('inventory').controller('InventoryController', ['$scope', '$http', '$stateParams', '$location', 'Authentication', 'Inventory', function($scope, $http, $stateParams, $location, Authentication, Inventory)
2. Methods:

- Add form field inventory
  $scope.addFormFieldInventory = function()

  -Usage
  This method dynamically adds fields to the inventory page that are used to update the quantity of an inventory item. Each inventory item is displayed with the option to edit the quantity.
- Update inventory quantity
  $scope.updateInventoryQuantity = function()

  -Usage
  This method updates the quantity of an inventory item.
- Delete inventory item
  $scope.deleteInventoryItem = function()

  -Usage
  This method removes an inventory item from the database. It displays an error message when a user wants to remove an inventory item that is used by a menu item, in this case it displays an error message and does not remove the inventory item from the database.
- Load inventory items
  $scope.loadInventoryItems = function()

  -Usage
  The load inventory items searches for and returns all the inventory items in the database. It displays an error message when it could not load all the inventory items.
- Create inventory item
  $scope.create = function(Boolean isValid)

  -Usage
  This method creates a new inventory item and stores it in the database. It displays an error message if all the neccessary fields are not entered correctly.
- Search inventory
  $scope.searchInventory = function(Boolean isValid)

  -Usage

This method searches for and returns a specific inventory item. It displays an error message if the inventory item could not be found.

- Update inventory
  $scope.updateInventory = function(Boolean isValid)

  -Usage
  This method updates a specified inventory item. It displays an error message if all the neccessary fields were not entered correctly or if the item could not be updated.

- Check if cafeteria manager
  $scope.checkCMUser = function()

  -Usage
  This method performs a check to te confirm that the user has the cafeteria manager role. It prevents unauthorised users to perform actions that only the cafeteria manager is allowed to perform.

## 9.2 inventory.server.controller

The inventory controller handles all the main functionality concerning inventory, such as adding an anventory item.

1. Methods:

   - Load inventory items
     exports.loadInventoryItems = function(JsonOBject req, JsonOBject res)

     -Usage
     This method retrieves and returns all the inventory items in the databse. It is used to display all the inventory items where neccessary.

   - Search inventory
     exports.searchInventory=function(JsonOBject req, JsonOBject res)

     -Usage
     This method searches and returns a specific inventory item. It returns an error message if the specified item could not be found, or if the inventory item's name is empty.

- Decrease inventory exports.decreaseInventory = function(JsonOBject req, JsonOBject res)

  -Usage
  This method decreases a specific inventory item with a certain amount. It returns an error message if the inventory item could not be found or if the inventory item could not be decreased.

- Update inventory
  exports.updateInventory = function(JsonOBject req,JsonOBject res)

  -Usage
  This method updates a specific inventory item. It returns an error message if the item to be updated was not found, or if the item could not be updated.

- Update inventory quantity exports.updateInventoryQuantity=function(JsonOBject req, JsonOBject res)

  -Usage
  This method updates the quantity of a specific inventory item. If the inventory item could not be found or updated it returns an error message.

- Delte inventory item
  exports.deleteInventoryItem = function(JsonOBject req, JsonOBject res)

  -Usage
  This method deletes a specific inventory item. It returns an error messag if the item could not be found or deleted.

# 10 Place Order

## 10.1 orders.client.controller.js

1. Declaration: angular.module('orders').controller('OrdersController', ['$scope', '$rootScope','$http', '$stateParams', '$location', '$cookies', 'Authentication', 'Orders', function($scope, $rootScope, $http, $stateParams, $location, $cookies, Authentication, Orders)

2. Methods:

   - Change item quantity
     $scope.quantityChange = function()

     Usage: Function to increase or decrease the number of items ordered. It is used when a user wants to order multiple of the same item. It then also increases or decreases the total price accordingly.

   - Change or add preferences
     $scope.prefChange = function()

     Usage: Function to add preferences. The user can specify how they prefer their order to be prepared.

   - Change the total
     $scope.subTotal = function()

     Usage: Function to calculate the total. This function calculates the total that will be displayed for the user at the bottom - so they can know how much their order is going to cost.

   - Place order
     $scope.placeOrder = function()

     Usage: Function to place the order. If the user is not signed in, the function redirects them to the signin page. If the user is signed in, the function checks the order total against the user's available balance. If the user has enough funds in their account the order is placed. Otherwise a pop-up message tells the user they have insufficient funds in their account and that if they proceed with placing the order they will have to use cash - if the user agrees to use cash the order is placed otherwise nothing is done.

   - Remove item from plate
     $scope.removeFromPlate = function()

     Usage: Function to remove an item from the plate. If the user decides they don't want an item anymore, they can remove it from the plate (before the order is placed).

## 10.2 orders.server.controller.js

1. Methods:

   - Place order
     exports.placeOrder = function(req, res)

     Usage: Function to place the order. It adds the order to the orders table if the user is successful in placing the order, otherwise it generates the appropriate error message.
   - Mark the order as ready
     exports.markAsReady = function(req, res)

     Usage: Function to mark the order as ready. The cashier can mark the order as ready when the items are done being prepared and ready to be collected. This changes the status of the order to ready and sends the client a notification to come fetch their order.
   - Send Email
     function sendEmail(uname, orderNum)

     Usage: Function to send the user an email. This function is called in markAsReady - it sends the user an email letting them know their order (with the given orderNumber) is ready to be collected.
   - Mark order as paid
     exports.markAsPaid = function(req, res)

     Usage: Function to mark the order as paid and collected. This function marks the order as closed in the orders table and it removes it from the process orders page.
   - Mark order as collected
     exports.markAsCollected = function(req, res)

     Usage: Function to mark the order as paid and collected. This function marks the order as closed in the orders table and it removes it from the process orders page. It also subtracts the total of the order from the user's limit because the client only collected the order and didn't pay. This can only be used when the user has sufficient funds in their account.
   - Get a list of orders
     exports.getOrderList = function(req, res)

Usage: Function to get a list of all the orders that are open and ready. This is used to display the orders that are still being processed and have not yet been collected on the process orders page (accessed by the cashier).