Testing Spec
Project:
Cafeteria Management System:
Reslove

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

August 27, 2015

Contents

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 atten-

tion 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.

4.2 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
   - ..

5 Place Order

5.1 Manage Profile

var manageProfileObj = require('./manageProfile'); extends CMS
Constructors: manageProfile.exports=function(jsonObj) -usage: creates instances of this component
Methods:

- changePassword exports.changePassword =function(jsonObj) where jsonObj contains employeeID, password, newPassword
- changeEmail exports.changeEmail = function(jsonObj) where jsonObj contains employeeId, email
- setLimit exports.setLimit = function(jsonObj) where jsonObj contains employeeId,newLimit
- displayBill exports.displayBill = function(jsonObj) where jsonObj contains employeeId
- viewHistory exports.viewHistory = function(jsonObj)where jsonObj contains employeeId
- generateFavourites exports.generateFavourites = function(jsonObj) where jsonObj contains employeeId