Functional Requirements Document Spesification
Project:
Cafeteria Management System: Reslove

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

May 28, 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 architectural requirements , functional requirements, application design and testing to provide a clear view of the system as a whole. An agile approach is being followed, hence the main use cases that this document will be focussing on are placing orders and managing a user profile. These will be explored in quite some detail. Screenshots of a few of the unit tests are included in the document, however, the actual javascript files with all the unit tests for the above mentioned use cases can be located on the github repository via the url provided above. The user manual required can be found in the Read Me file situated in the github repository.

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 Functiona Requirements and Aplication Design

In this section we will discuss the application functionality required by users and other stakeholders.
T-RISE have decided t use the Agile management approach in designing the Cafeteria Management System. The method involves an interactive and incremental method of managing and designing the system. In the agile method we will submit deliverables in stages, as they are completed, thus we will complete small portions of the deliverables in each delivery cycle. For this reason we will create the needed daigrams and planning for each stage, add these to our documentation and then implemet the respective modules. After implimentation we will do thourough unit testing as descussed under the 'testing' heading. If all the tests passed for the respective components it will be added to our working system, thus the working system will be fully functional at all times.

4.1 Use Cases

Below is a list of all the use cases we have identified:

- Login
- Register
- Manage Profile
- Place Order
- Notify
- Manage Inventory
- Report

4.2 Use Case Prioritization

Below the use cases mentioned above will be catagorized as critical, important or nice to have.

- Login - Critical
  This is a critical use case since you can not send through any order that is placed if you are not logged in you will only be able to view the menue if you are not logged in.
- Register - Critical
  This is a critical use case since you can not log into the system if you are not registered and if you are not logged in you can not place orders.
- Manage Profile - Important
  This use case will be considered as important, the user needs to be able to see his/her balance edit contact information and so forth, but if a user can not manage the profile it will not cause the system to crash for example , although it is still crutual in the system that users will be able to view certain information it will be classified as important.
- Place Order - Critical
  This will be classified as critical since the whole system revolves around the ability of placing orders at the cafeteria and viewing balances.

- Notify - Important
  this will be classified as important, although some of the notifications such as notifieng a user when their order is ready for colection is a nice to have functionality, other notifications such as notifying the casher or cafeteria manager that they are low on certain inventory is crutial to the working of the system, we will classify notification as important

- Manage Inventory - Important
  This use case will be classified as important since the amount of stock willdetermine what may and may not be ordered.

- Reporting - Important
  This use case will be considered as important, users can print a billing report and administrative users will be able to send billing reports to Pay-Roll, thus this functionality can be considered as important.

4.3 Use Case/Service Contracts

4.4 Required Functionality

4.5 Process Specification

4.6 Domain Model

5 Open Issues