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 26, 2015

Contents

1 Introduction **3**

2 Vision **3**

3 Background **3**

4 Architecture Requirementss **4**
4.1 Access Channel Requirements 4
4.1.1 Human Access Channels 4
4.1.2 System Access Channels 5
4.1.3 Integration Access channels 6
4.2 Quality Requirements 6
4.3 Integrationl Requirements 8
4.4 Architecture Constraints 8

5 Functiona Requirements and Aplication Design **9**
5.1 Use Cases 10
5.2 Use Case Prioritization 10
5.3 Use Case/Service Contracts 11
5.4 Required Functionality 11
5.5 Process Spesification 11
5.6 Testing 11
5.7 Domain Model 11

6 Open Issues **11**

1 Introduction

This document contains the functional requirements specification 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 architecture requirements , functional requirements and application design to provide a clear view of the system as a whole.

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 then assist in the collection of payments for the cafeteria, manage inventory/stock, facilitate payments for access cards (or the use of unique access card numbers), and facilitate ordering from the cafeteria. The system will allow the cafeteria to use the system that they are currently using as it is with combination of a user friendly application and online facility to place orders and check stock and make predictions of needed stock for the following week.

3 Background

As specified in the tender proposal document from Resolve - the cafeteria is currently cash only and does not accept bank cards or electronic payments. This makes it difficult for employees as they have to carry cash if they want to purchase anything from the cafeteria. In this case the employee might as well go to an outside food provider and pay with their preferred method of payment. This problem wastes fuel for the employees, time for the company, and does not bring in the maximum amount of income to the cafeteria, hindering its growth and improvement.

Resolve is looking for a way to accept payments from employees for the canteen using their employee access cards 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. At our first meeting with Resolve they have also brought under our attention that at times the cafeteria does not have enough stock to make some of the menu items, thus the reporting of inventory or stock will also be part of the system. The system will predict what inventory/stock needs to be bought for the next week.

4 Architecture Requirementss

The software architecture requirements include the access and integration requirements, quality requirements, and architectural constraints. These points will be thouroughly discussed below under.

4.1 Access Channel Requirements

In this section we will discuss the requirements for the different channels through which the system can be accessed by people (users - client side) and systems (server-side), we will also discuss the integration channels which will be suported by the system.

4.1.1 Human Access Channels

The Cafeteria Management System would be accessed by different users via the online web page (thus it willbe a web interface) or through the mobile application that will be suited for different platforms. The web interface will be accesable through all the standard-complient web browsers for example Morzilla Firefox, Google Chrome, Safari and Microsoft Internet Explorer. The mobile application will be available to access on multiple platforms including the standard iSO/Anroid platforms. Different services will be available to different users. There are five types of users: Super User, Cafeteria Manager, Casher, Normal User, and Resolve Admin:

Super User

The super user will be the only administrative user that will have global access to all the functionality of the Cafeteria Management System, in particular the Super User will have access to the branding of the Cafeteria

Management system (changing the logo and so forth) . The super user will also have access to all the functionality of all the other users listed below.

Cafeteria Manager

The cafeteria manager will have the ability to view his/her own profile, edit his/her profile, and place orders as normal users can. This user will also be able to add and edit menu items, view the orders placed and the inventory or stock, and add or remove inventory or stock.

Cashier

The cashier will be able to view his/her profile, edit his/her profile, view the orders placed, and tick off those orders that are done and collected. The cashier will also be able to make a purchase and check inventory or stock and add or remove inventory or stock, since some stock could have gotten old or rotten and needs to be removed from the available stock list.

Normal User

The normal user will typically be a resolve employee registered on the Cafeteria Management System. A normal user will only be able to view his/her profile, edit his/her profile, place orders, check if their order is ready, and print their balance reports.

Resolve Admin

The resolve admin user will only be able to view all the registered users and their total balance outstanding, this is for administrative and financial usage purposes requested by the resolve team.

4.1.2 System Access Channels

The differant thecnologies we will be using will use appropriate access channels that will be supported by the thecnologies we will be using. For example we will be using NodeJs running on an Express server and the server needs to be connected to the database we will be using getting informations and transfering it from the server to the respective node modules and so forth. The integration channels will also be accessable by the mobile applications, since Phone Gap which is the program we will be using to help us convert our web interface into a mobile application will give the user the illusion that

they are working on an application but it will run in the background as a web interface.

4.1.3 Integration Access channels

- The system will have to integrate with the database getting information of the employees - such as contact information to notify the user that an order is ready, get inventory or stock and so forth.
- The system will also have to integrate with the server to pass information to and from the database.

4.2 Quality Requirements

Preformance

The Performance of a software system will be measured in the run time efficiency. In the Cafeteria Management System we will be implementing technologies such as AngularJS wich will ensure for fast interactive services on the client side, giving the user a fast and effective way to order their meals from the Cafeteria without wasting voluble work time of the company. Although the performance is also influenced by the architectural design we will ensure that processes on the server side are also fast and efficient to work smoothly with the client side. Ideally Reporting queries should not take longer that +/- 6 seconds and non reporting queries should take less than +/- 0.4 seconds.

Reliability

When creating a software system it is not possible at the first run to create a system that is completely 'bug free', but a certain level of debugging and reliability of a system is needed to have it fully functional. Thus in the case of reliability unit testing is of upmost importance, if all pieces of code that gets added into the working system if fully tested for every possible scenario your system is more likely to have a very higher reliability than systems where only a few unit tests were conducted. When creating a system for a client it is important to make it as reliable as possible to promote good and satisfactory services as promised.

Scalability

Scalability refers to a software system's ability to handle increased workloads. The Cafeteria Management System will be scalable if it can handle more than the currently registered employees, or even twice or three times as many users.

Security

Security is considered as an important quality requirement in any online software system. For the Cafeteria Management System no user will be able to log into the system without being registered to the system. On registration all personal details, such as the employees e-mail address, will be verified to ensure all registered users can contacted if needed.

Flexability

In the creation of the Cafeteria Management System it is important to keep the software as technology neutral as possible, this is why in the creation of an application of our online system the application will be able to work on multiple platforms facilitating a wide variety of users and the online facility will be able to open on all standard browsers.

Maintainability

Maintainability refers to the design of the system that needs to allow for the addition of new requirements without the risk of introducing new errors. In the process of implementing the Cafeteria Management System it is important to remember that the owner of the system might want to add some functionality to the existing software at a later stage. It is thus important to implement coding standards to keep the software neat and readable since future developers should be able to easily understand the system. It will ensure that when the software is altered there will not be any trouble reading it or discovering bug. Maintainability thus also refers to the testability of a software system - it is important to ensure that the software system is adequately tested at all levels.
The other important factor for maintainability is to use thechnologies that will be around for a long time.

Testability

Testing the software system is of upmost importance and thus unit test-

ing will take place, unit testing using mock objects isolating components , integration testing will also take place - where the components will now be tested in an actual enviroment not wilt mock objects - thus all pre-conditions needs to be met and all post conditions needs to be true indicating that the services stated by the service contracts has been provided. Testing should also include scalibility testing, usability testing and preformance testing.

Integrability

Integrability refers to the testing of separately developed components to ensure that they work together. As we make our different modules using angular we will make sure each module when needed to interact with another model can do so efficiently. This will be achieved through unit testing and integration testing of the different modules as we build our system. This also means that if a module is removed from the system, the system will be able to run smoothly, it will not disrupt the whole system. If we thus modify one of the modules it won't disrupt the rest of the system. Therefore the system should also be able to address future integration requirements by providing access to its services using widly adopted public standards.

Usability

Usability can be considered as a core quality requirement .
Usability involves measuring the user's performance with regard to the software system. It is important to have a usable and pluggable system that the staff members of Resolve can use with ease; this implies that the site does not break down every time you click a link for example.
Usability of the Cafeteria Management System will be ensured by unit testing, all aspects of each module of our system will be thoroughly tested before it will get passed on to be implemented in the working system - thus firstly the system will be fully functional.
Secondly the system needs to be simple and easy to use without any guidance or ducumentation thus it needs to be user friendly.

4.3 Integrationl Requirements

4.4 Architecture Constraints

Technologies we will be using in the creation of the Cafeteria Management System includes the following:

- HTML - The Software system will be mainly web-based.
- JavaScript together with AngularJS and NodeJS - this will enable us to add extra functionality to our web page and modulirise the system thus also helping us to implement dependancy injection. For creating reports we can also import ReportingJS to create visually pleasing reports that is logically structured
- CSS together with BootStrap - which will allow us to style our page and also make it interactive.
- Mongo DB for our database which goes extreemly well with NodeJS.
- Express server will be set up as our server that will host the system.
- Phone gap will be used to convert our web page into a usable application which will then look like the online webpage that will run like a web interface in the background but will seem like a mobile application to the user that will be acceessable from multiple platforms.

The above mentioned thecnologies will be our basis we will create our system on, but as we are busy building the Cafeteria Management System we will add other thechnologies as needed.

5 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.

5.1 Use Cases

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

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

5.2 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.

5.3 Use Case/Service Contracts

5.4 Required Functionality

5.5 Process Spesification

5.6 Testing

5.7 Domain Model

6 Open Issues