Requirements Document

Version 0.2

Client: A software toolbox for small retail shops

Dan Plămădeală, S3436624
Abel Nissen, S3724786
Ruben Biskupec, S4235762
Florian de Jager, S3775038
Arjan Dekker, S3726169

university of groningen

Faculty of Science and Engineering

Lecturer: Dr. Mohamed Soliman
Teaching Assistant: Hichem Bouakaz
Last updated: Tuesday 19th May, 2020

Contents

Introduction

Several hundreds, if not thousands, of small supermarkets in The Netherlands are using archaic software systems to run their Points of Sales (cash desks) which are not very friendly in terms of extracting management information. Simple queries like "what is being sold in my shop at which time-of-day", "how many of item X are being sold per week", "Should I order more or less of item Y" are difficult to answer with hard data. Hence most shop owners rely on their "gut-feeling" which is not a smart thing to do.

The "Dorpswinkel" in Sauwerd (a village 10km north of Groningen) is a good example. The shop is owned and run by the people living in the village. As it is a "voluteers project" there are many people doing work in the shop and there is not a single "set of brains" in the shop that knows all. So relying on gut feeling is not an option - we need hard data on sales to be able to optimize the shop.

As a team our objective is to build a parallel MySQL database that will work as a better basis to retrieve management information. We also want to build a smartphone app that is able to scan a barcode and popup the recent sales of this product. The final goals are to make information on the stock and sales of goods accessible in and to create a generic database platform that allows queries to be performed alongside apps that contain these queries.

Actors

- manager: The manager of the shop. Keeps track of the financial situation of the shop and is responsible for it.
- employee: A standard employee of the shop that does various jobs in the shop. One example of such job is comparing and updating price labels.
- moderator: Is also an employee of the shop, but has extra privileges. The moderator is able to view purchasing information and issue queries.

User stories

This chapter focuses on actions users should be able to perform in the finished product.

Critical Functional Requirements

- ☐ As a manager, I want the system to store the sales data in the database, so that I can easily retrieve information.
  - The system automatically opens a new zip-file that is created by Casman every day, parses it and stores the data in the database.
- ☐ As an employee, I want there to be an easily extendable API, so that I can use and protect the sensitive customer sales data.
  - Backbone of storing customer data safely.
- ☐ As an employee, I want the API to handle both uploading and editing of sales data, so that I can run queries on that data.
  - Backbone of accessing the database.
- ☐ As a manager, I want to be able to query the database system via the web, and have the results returned as comma-separated or excel tables.
  - Number of sales of product x versus day of week, time-of-day.

Important Functional Requirements

- ☐ As a manager, I want to be able to access the query system off premise, so that I can keep track of sales and inventory while at home.
  - Database should be off premise.
- ☐ As an employee, I want to scan the bar-codes via mobile app, so that I can see the current price.
  - Should be able to print new price label if the 2 prices do not match.
- ☐ As a manager, I want to access the inventory via a smartphone by scanning a bar-code, so that I can see if anything has been stolen.
  - Convenient overview such that the person using it can use the data for ordering stock.

- ☐ As a manager, I want to access the status of every item via a smartphone by scanning a bar-code, so that I can see how well a certain product is selling.
  - The person using it can use the data to predict future sales.
- ☐ As a manager, I want to be able to query the database for "Koppelverkoop", i.e. which products are sold together in pairs with sales of product X, so that I have more insight in the combinations.
  - (For instance, is the pair of beer and potato chips often bought together?)
- ☐ As a moderator, I want to be able to request the total number of sales between a given interval, so that I can decide whether or not it is effective to keep the shop open.
  - Useful to see when it is profitable to keep the shop open.
- ☐ As a moderator, I want to be able to request the total number of sales of a certain product between a given interval, so that I can see when we should have stock of certain products.
  - Useful to see at what times you need certain products to be stocked.

Useful Functional Requirements

- ☐ As a moderator, I want to view the purchasing prices of the goods, so that I can compare them to selling prices and calculate the margins.
  - Useful for computing profit/loss.
- ☐ As an employee, I want to be able to scan newly updated price labels and compare them with our current price, so that I can see which labels apply to our products.
  - Quick way of finding out which items of all updated prices are actually in the shop and need a new price label.
- ☐ As a customer, I want to be able to have an app-accessible loyalty card, so that I can have the cashier scan that card to apply a discount.
  - Store a customer's card ID with a sale transaction.
- ☐ As a customer, I want to be able to access my loyalty card account in an app, so that I can see what I've bought on it.

- Store a customer's card ID with a sale transaction.

☐ As a manager, I want to be able to reach the owner of a loyalty card, so that I could send him the current list of bonus products.

- Retrieve customer data from loyalty card, either via the app or email.

Non-Functional Requirements

1. **Security**

   ☐ To secure communication, https should be used.

   ☐ The barcode scanning and web query should only be accessible when the employee provides a valid username and password.

2. **Performance**

   ☐ The parser script should be able to process a receipt in less than 10 milliseconds so as not to get a backlog of receipts in the system.

   ☐ Queries should not put a big strain on the server. At least 10 queries need to be able to be performed at the same time without noticing any delay.

   ☐ API calls should take less than a second. Again to prevent a backlog.

   ☐ Reading barcodes should be close to instantaneous.

3. **User friendliness**

   ☐ The language of the user-interface should be Dutch mainly.

4. **Usability**

   ☐ The barcode scanner and query should be easy to understand and use. A single 15 minute training session should be enough for a basic employee to know how to operate it.

   ☐ Experienced employees should be able to use all the system functions after a total of one hour training. After this training, the average number of errors made by experienced users should not exceed two per day.

5. **Portability**

   ☐ The barcode scanner and query should work properly on modern browsers including browsers on phones.

- ☐ The client app should work on both iOS and Android platforms.

6. **Maintainability**
   - ☐ The code should be clear and readable. [For potential new employees to be able to tweak and update it.]
   - ☐ There should be an English documentation of the system.

Won't Do

☐ -

Global Overview

A rough overview of the features we will be focussing on. For details see the design document.

- ☐ Store the shop's (sales) data in a new database.
- ☐ Allow for extensive queries to be run on that database.
  Think of queries such as:
  - How many products are being sold between 17:30 and 18:00
  - How do the sales of a product change during the year? (Show number of sales over 12 months)
- ☐ Access product's info in the database by scanning its barcode with a smartphone camera.
- ☐ View products frequently sold together in pairs.

Customer Meetings

When	What
26/02/2020	- Introduction meeting, discussed initial requirements. - Decided to prepare a requirements document for next time.
09/03/2020	- Confirmed requirements, discussed coding approach. - Decided to prepare a short demo presentation for our next meeting.
24/03/2020	- Short demo presentation. - General discussion about the project.
20/04/2020	- Demo presentation. - Gathered more updated requirements.
27/04/2020	- Showed finished modules. - Discussed implementation in the shop.
06/05/2020	- Discussed actual deployment in the shop. - Updated client on our progress.
12/05/2020	- Hands-on mobile phone demo. - Made preparations for testing.
19/05/2020	- Discussed bugs and improvements. - Gathered hands-on feedback.

Change log

Who	When	Which section	What	Time
Dan	27.02.20	The document	Created the document, added features.	20 min
Dan	27.02.20	The document	Added log table and user stories template. Made some changes to the layout.	10 min
Abel	28.02.20	The userstories document	Added 6 user stories.	1h
Arjan	02.03.20	The visualization document	Added a visualization for the application.	time
Abel	06.03.20	The userstories document	Added all provided requirements, rewrote and sorted them.	40min
Abel	06.03.20	The userstories document	Updated requirements (non and functional ones) with all available data so far.	30min
Florian	06.03.20	The userstories and users document	Extended the non-functional requirements and added basic info to user	1h
Ruben	08.03.20	Introduction	Added the Introduction	1h
Abel	10.03.20	The userstories document	Split complex requirements into multiple more clear ones. Added loyalty card user stories	1h
Abel	10.03.20	The customer meetings document	Updated document with meeting had so far and decisions made.	10min
Ruben	10.03.20	User stories	Split compound user stories and other changes	20min
Dan	13.03.20	Front Page	Improved the layout	5 min
Dan	13.03.20	non-functional requirements	improved the non-functional requirements	20 min
Abel	13.03.20	non-functional requirements	finished non-functional requirements	1h
Arjan	13.03.20	non-functional requirements	finished non-functional requirements	1h

Who	When	Which section	What	Time
Florian	13.03.20	non-functional requirements	finished non-functional requirements	1h
Abel	13.03.20	Introduction	edited the introduction	15 min
Ruben	16.04.20	Functional Requirements	Prioritized user stories	15 min
Abel	20.04.20	Functional Requirements	Updated due to new meeting with client	15 min
Abel	27.04.20	Global Overview	Added important functions in a quick overview	30 min
Ruben	11.05.20	User stories	Made user stories more specific	45 min
Abel	16.05.20	Introduction & Global Overview	Compacted the intro, detailed the overview	30 min
Abel	19.05.20	User meetings	Updated relevant information	20 min