Software Requirements Specification
Electronic Marking System

By:
Melany Barnes - 12030466
Maret -
Next person -

For:
Jan Kroeze
(Computer Science Department - University of Pretoria)

Version 1

February 27, 2014

Contents

1 Introduction

This document is formulated using LaTeX and UML class diagrams. Its purpose is to illustrate to the client the requirements of the project and reach an agreement between developers and client for the scope of the project. This document states clearly what the developed solution must be able to do and what function the solution cannot do. This document is regarded as an official contract between developers and client when the pre and post conditions stated are met. Due to the nature of software development processes, this document may still be finalised.

2 Vision

The aim of this project is to provide a mobile web-based marking IT solution to the client, Mr Jan Kroeze at the University of Pretoria, that allows users to upload student marks to a mark sheet and also be able to use these marks at a later stage. The solution is aimed to be easy to use and more efficient than the pen-and-paper marking system.

3 Background

4 Architecture requirements

4.1 Access channel requirements

Lecturers, tutors and students will access this system's services through a Mobile Android application. They will also be able to use this system's services through a web browser interface. A lecturer will be able to manage the marks and will be able to extract mark lists as csv files.

4.2 Quality requirements

Auditability:

(Source: Jan Kroeze, Priority: High, Requirement: NFRQ1)
Everything that happens on the system must be saved in a log and must be able to be queried. Especially alterations to marks will be logged. Any updates made as well as additions added will be logged.

Flexibility:

(Source: Jan Kroeze, Priority: Medium, Requirement: NFRQ2)
The application will work on Mobile phones with an Android operating system. The web interface will be able to be viewed from a mobile device or from a personal computer.

Reliability:

(Source: Jan Kroeze, Priority: High, Requirement: NFRQ3)
The application will always be on line and will be usable as long as users are connected to the internet, same applies to the web interface. If the user loses internet connection the system will make a local backup on the device being used and will commit this data to the system as soon as it gets internet connection again.

Scalability:

(Source: Jan Kroeze, Priority: High, Requirement: NFRQ4)

The system needs to react quickly on input or commands. Performance and reliability should not be dependent on the amount of users on the system.

Security:

(Source: Jan Kroeze, Priority: High, Requirement: NFRQ5)
Users will have to log in with their specific user name and password to access the system. Only valid users, with valid authentication and presence on the system may gain access to the system. Lecturers will be able to change closing times of mark sheets, change marks and view reports. Tutors will be able to input student marks which he/she is assigned to. Students will only be able to view their own marks.

Usability:

(Source: Jan Kroeze, Priority: High, Requirement: NFRQ6)
It must be a simple, easy to use system.

4.3 Integration requirements

The csv file that comes from the cs website, will import the students onto a mark sheet to have the list of students.

The integration channel that is being used is where the application or website is integrated with the mark sheet (database) so that they are in synchronous. When the application is started up it can get its data from the database and when a mark has been entered it will update after entering of the one mark that was entered. It will be so small of a file that when the update is made it will change almost instantly.

The protocol that is used, the system (the application or website is integrated with mark sheet) is responsible for transforming data across industry standard protocols. A JSON to SOAP conversion or a SOAP/HTTP to a SOAP/JMS or SOAP/Message Queue conversion would be responsibilities of this system.

Any quality requirements for the integration itself, is performance, audit. In performance the file must be in smallest format, in audit the system lock and when things are send it must be able to check the time the file was sent

The mark sheet can also be exported to a csv or a pdf for email purpose or a new mark sheet

4.4 Architecture constraints

The technologies that are being used are:

- Python, Django
- Android
- SQL
- HTML 5
- CSS
- https/http
- Csv
- Pdf
- XML
- POP3 and SMTP

Client-Queue-Client systems - passive queue architecture

This approach allows markers to send marks to a mark sheet. Markers can read data from and send data to a server that acts simply as a queue to store the data and let the lecturer edit and look at the marks and students retrieve marks to browse. This allows multiple users to distribute and synchronize files and information. The fact that this architecture has a queue, that is all why this will be used. This architecture will be used partially for the system.

SOAP interface is what this system is mainly going to focus on

It is a protocol to send markup(structured) information of web services, the system will work on xml and csv. It dependent on message frameworks; which is the smallest over internet for communication. For when the marker retrieve of the mark sheet and giving marks the system will synchronous. The sync will be between application/website and mark sheet(database).

5 Functional requirements

5.1 Introduction

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ01)
Any registered user should be able to log in and out of the system using a unique username and password. After a user has loged in, he is redirected to his home page.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ02)
A student must be able to view their marks on the application as well as on the web interface. The student can then view each module's marks separately via links to each module.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ03)
A marker, has additional links to the modules he marks. When a marker click on these links, the active marksheet of the module is displayed. The marker can then click on the marksheet to open the marksheet and enter marks. When the marker open the marksheet he has to enter the students initials, surname, full name, student number or username. The system has to use autocomplete search for this function. The marker then may view, delete add or modify the marks. Afterwards he has to save and close the marksheet.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ04)
On a lecturer's home page, a list of all his existing marksheets is

displayed. On the home page, he may either create a new marksheet, or view an existing marksheet. An existing mark sheet can be viewed by searching or by selecting one out of the list.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ05)
To create a new marksheet, the lecturer of a module must have a [New Marksheet] button. For each new marksheet, the lecturer should be able to enter a description, question amount with a maximum amount of marks per question, a rubric, marker's privileges, expiry date and release date. The marking privileges refer to who may enter or change marks, the default should be the lecturer only. The expiry date refer to whan the marksheet is locked, meaning no more marks may be entered. The release date refer to when the student may view their marks, the default is the expiry date. After all the details are entered, there must be a form of confirmation like a [confirm] button.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ06)
When viewing a marksheet, the lecturer may choose to unlock the marksheet. The lecturer may then modify the marks on the sheet. The lecturer must save the mark sheet and lock it.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ07)
When viewing a marksheet, the lecturer may export the marksheet in either csv or pdf file. he may also choose to delete the mark sheet

(Source: Jan Kroeze, Priority: Important, Requirement: FRQ08)
The lecturer should be able to view statistics of a marksheet. This report must be exportable as a pdf. This includes the percentages that passed and has extinctions, as well as the average and standard deviation. The report also include a distribution graph, pie chart and bar graph.

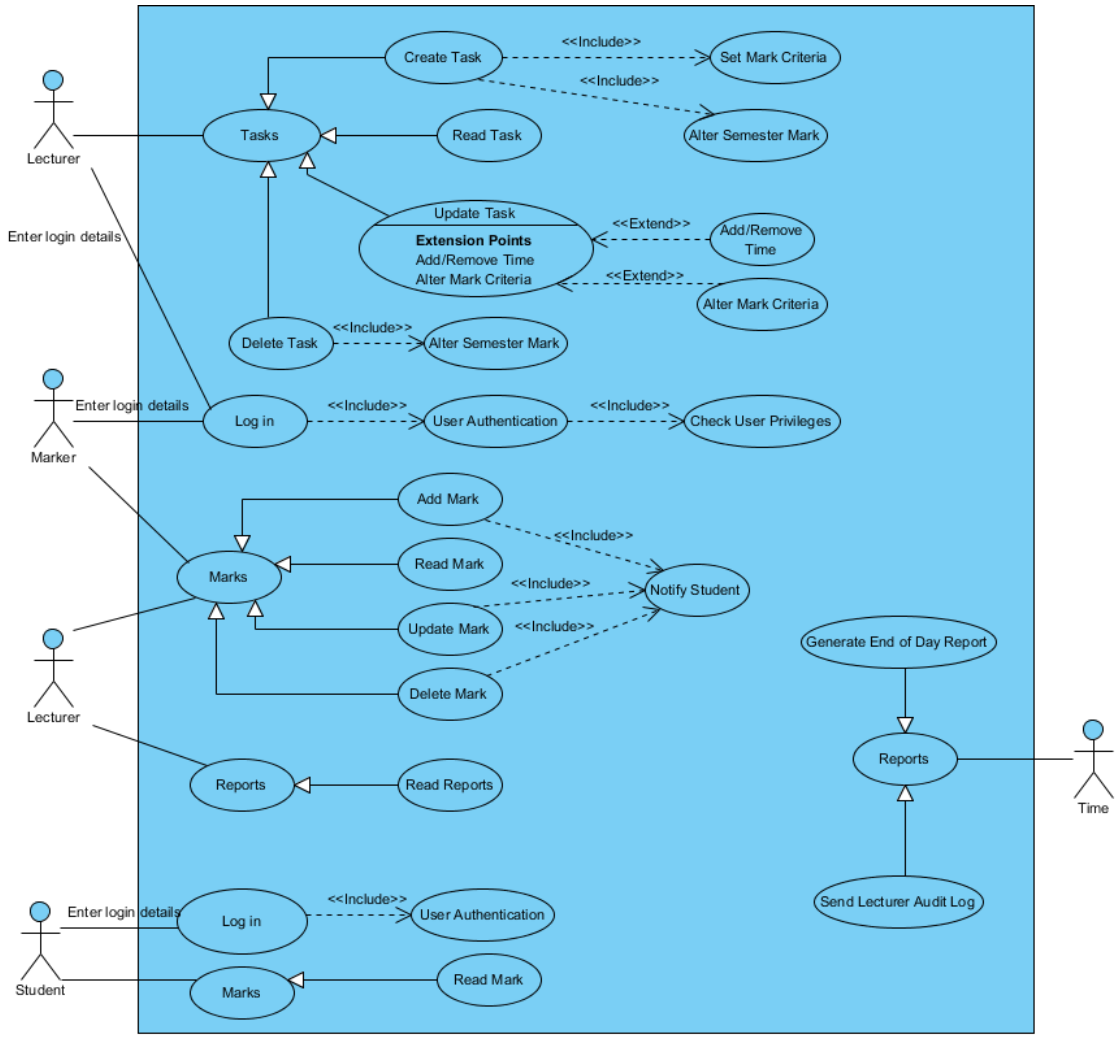
(Source: Jan Kroeze, Priority: Important, Requirement: FRQ09)
The system must be able to calculate compositional marks.

(Source: Jan Kroeze, Priority: Critical, Requirement: FRQ10)
The system must log all changes made to the marksheet. This must include a timestamp, the modification and the user involved. Only the lecturer should be able to view the log.

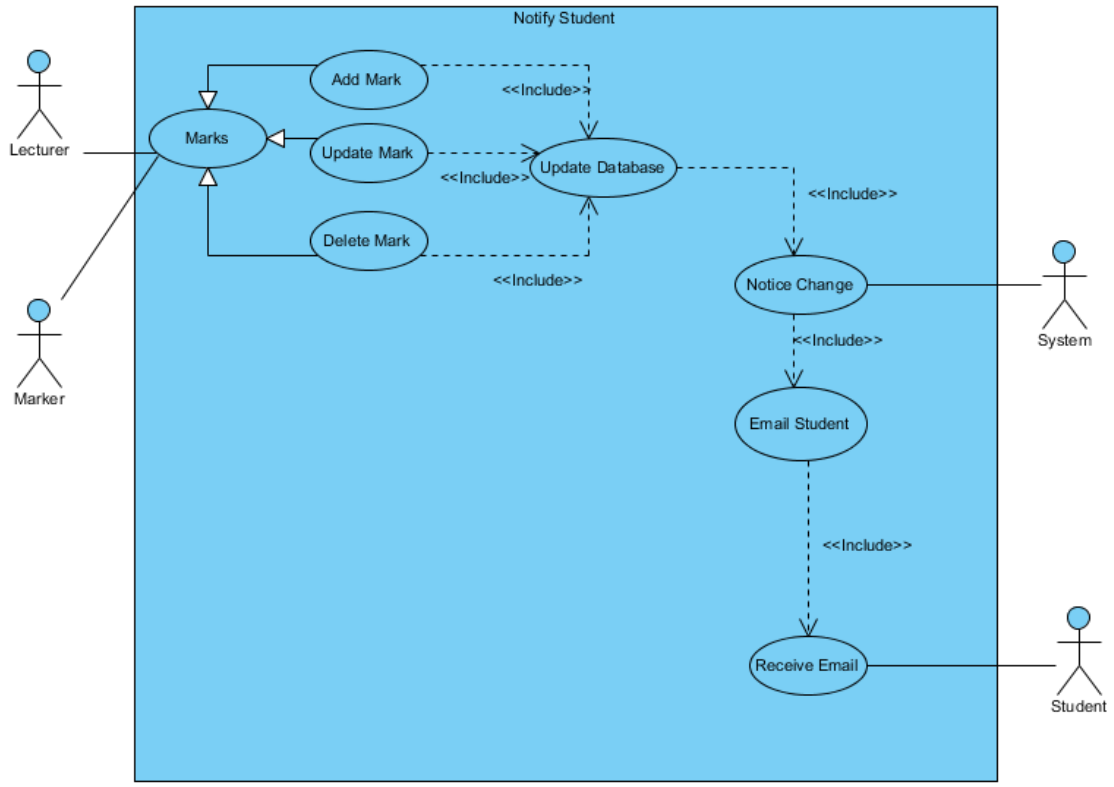
(Source: Jan Kroeze, Priority: Nice-To-Have, Requirement: FRQ11)
The system must notify a student when his marks is modified.

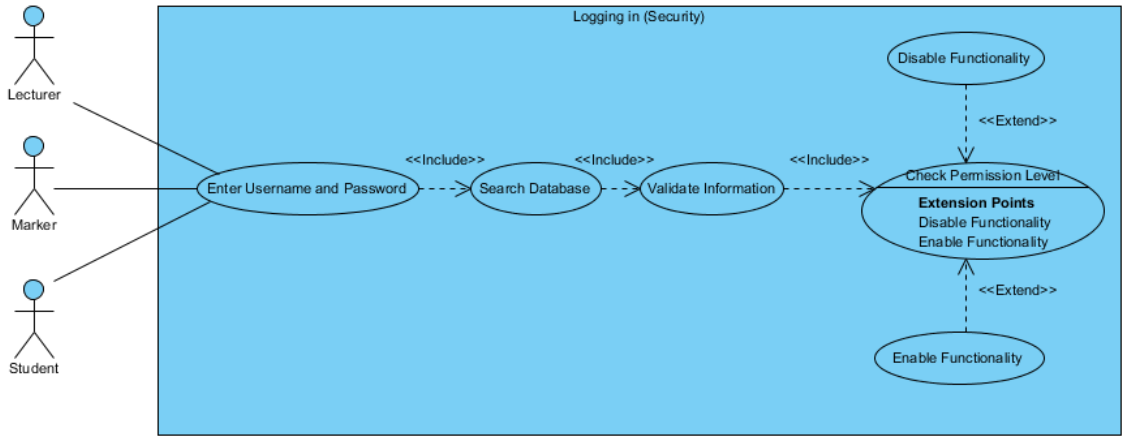
(Source: Jan Kroeze, Priority: Important, Requirement: FRQ11)
The system must automatically log out after no activity in 30 minutes.

5.2 Scope and Limitations/Exclusions



5.3 Required functionality





5.4 Use case prioritization

Critical Use Cases are the main cases that the system is made up of namely, Logging In, Create Tasks, Create Marks and Generating Reports. Without these cases the system will have limited to no functionality which will lead to a system that is not required by anyone.

Important Use Cases are the cases that improves the critical use cases and introduces a wider variety of functionality. These cases are Read, Update and Delete of Tasks, Read Update and Delete of Marks.

Nice-To-Have Use Cases make the system more user friendly and provides a better user experience. This case is the Notify Student use case, this is where the system automatically, emails the student of any changes made, either to their marks, namely adding marks, updating marks and deleting marks. This functionality can also be extended to the creation, update and deleting of tasks. This will help the student to stay up to date with what tasks needs to be done in the module.

5.5 Use case/Services contracts

5.6 Process specifications

5.7 Domain Objects

6 Open Issues

- Theft/Loss of mobile device
- Cannot be held resonsible if the user does not log off.

7 Glossary

- CSV - Comma-separated values

- UML - Unified Modeling Language