Software Requirements Specification
Electronic Marking System

By:
Melany Barnes - 12030466
Christo Brits - 11080923
Po-Han Chiu - 11063612
Maret Stoffberg- 11071762
Next person -

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

Version 1

February 27, 2014

Contents

1 Introduction 3

2 Vision 3

3 Background 3

4 Architecture requirements 4

4.1 Access channel requirements 4
4.2 Quality requirements 4
4.3 Integration requirements 5
4.4 Architecture constraints 6

5 Functional requirements 7

5.1 Introduction 7
5.2 Scope and Limitations/Exclusions 10
5.3 Required functionality 11
5.4 Use case prioritization 12
5.5 Use case/Services contracts 13
5.6 Process specifications 13
5.7 Domain Objects 16

6 Open Issues 16

7 Glossary 17

1 Introduction

The purpose of this document is to illustrate to the client the requirements of the project and reach an agreement between the developers and the client for the scope of the project. This document states clearly what the developed system must be able to do and what functions the system cannot do. This document is regarded as an official contract between the developers and the client when the pre and post conditions stated are met. This document is formulated using LaTeX, Use Case Diagrams, Activity Diagrams and UML Class Diagrams.

2 Vision

The aim of this project is to provide a mobile web-based marking IT solution to the client, Mr Jan Kroeze from the Computer Science Department at the University of Pretoria, that allows users to upload student marks to a mark sheet and also be able to use/view 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

Currently the course module, COS 222, offered by the Computer Science Department, University of Pretoria, uses a pen-and-paper sytem to mark their student practicals. During a practical marking session, markers/tutors gives the student a piece of paper to write their own names and student number. The marker then writes the mark obtained by that student on that piece of paper. Only after all students have their practicals marked, will all the marks written on different pieces of paper be digitalized and submitted for use by the lecturer. From this system, the client sees the opportunity to use an IT solution for the markers to directly submit the marks electronically to a database, that the lectuerer stores the marks, when the markers are marking the practicals. This reduces the potential risk of student marks being lost before it is recorded electronically. This also reduces the use of paper during

each practical marking seesion, saving trees. The idea of using the Android platform is because it is assumed that tutors are most likely in possession of a Android OS smartphone, so there will be no hardware issues. It is also assumed that there are wifi-hotspots in the Informatorium, University of Pretoria, where the practical marking sessions are assumed to take place, so there will be no internet connection issues.

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 upload/extract mark lists as csv files.

4.2 Quality requirements

Auditability:

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

4.3 Integration requirements

All the data will be imported into the system via a CSV file.

The database will be updated in real time. Meaning when a mark sheet is opened, the relevant data will be uploaded to the application or web interface from the database and thus as a mark is entered by a marker, the system will be updated.

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.

Since speed is critical, the files used to update the database will be as small as possible. This also increases scalability

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 mark sheet 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 mark sheets 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 Mark Sheet] button. For each new mark sheet, 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 mark sheet, the lecturer may choose to unlock the mark sheet. 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 mark sheet 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 mark sheet. 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.

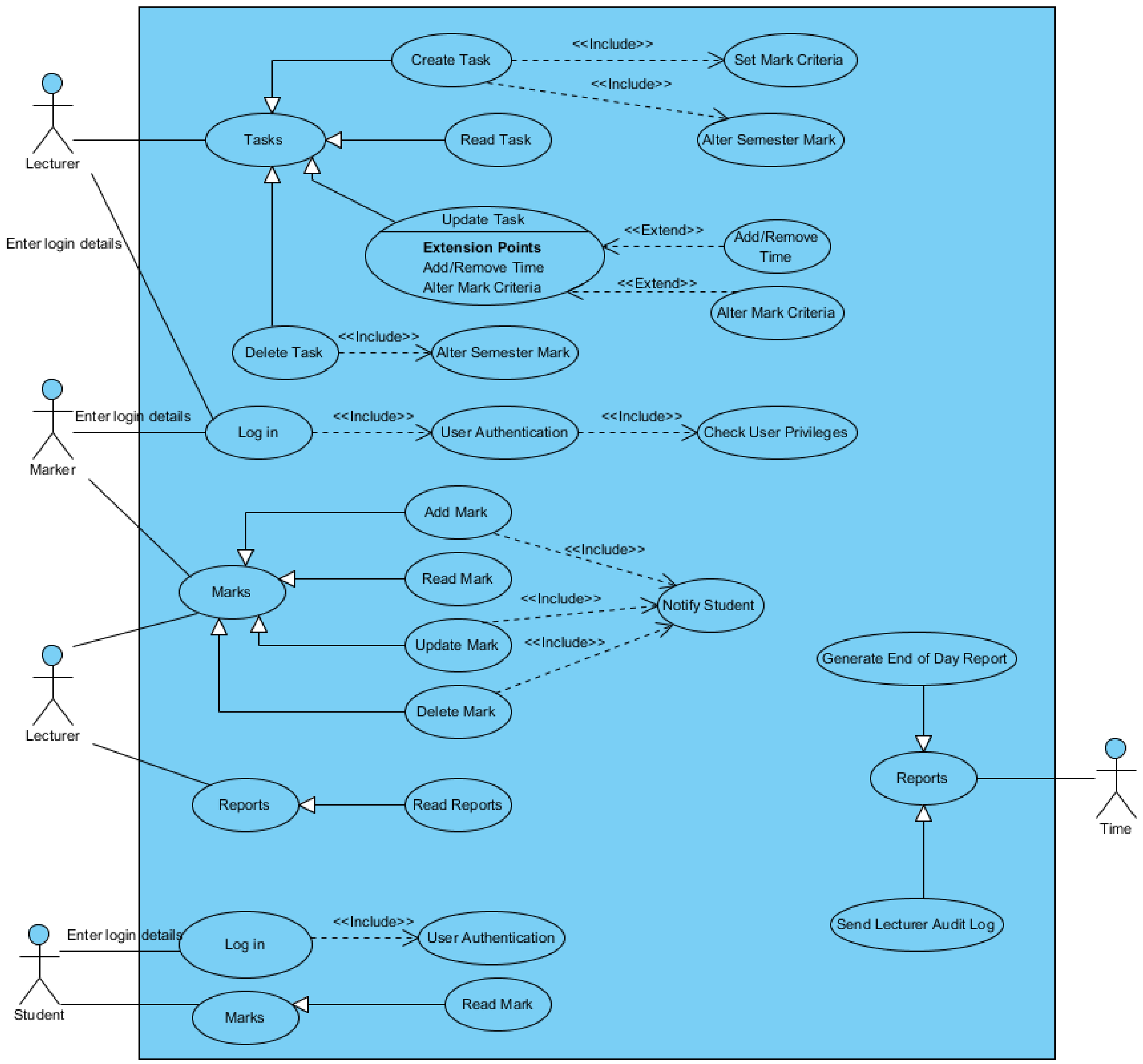
(Source: Jan Kroeze, Priority: Important, Requirement: FRQ09)
The system must be able to calculate compositional marks.

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

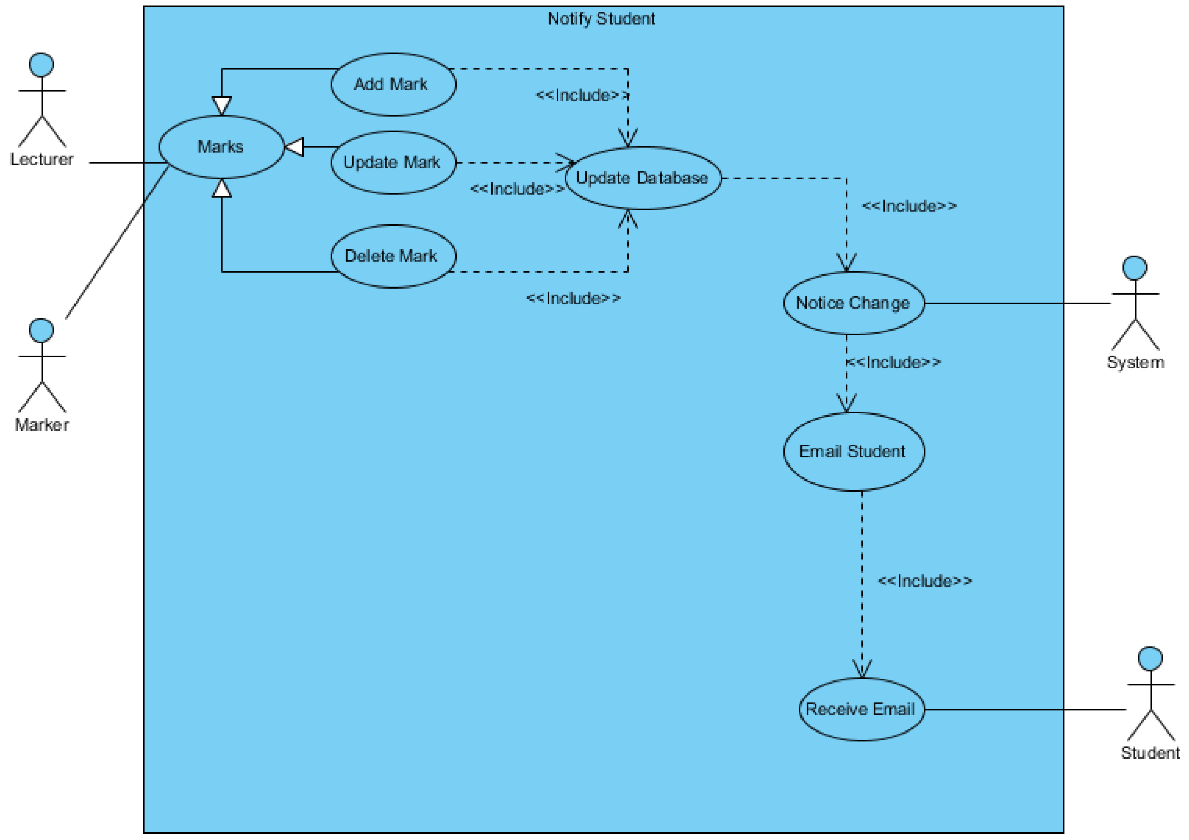
**(Source: Jan Kroeze, Priority: Nice-To-Have, Requirement: FRQ11)
The system must notify a student when his marks is modified.**

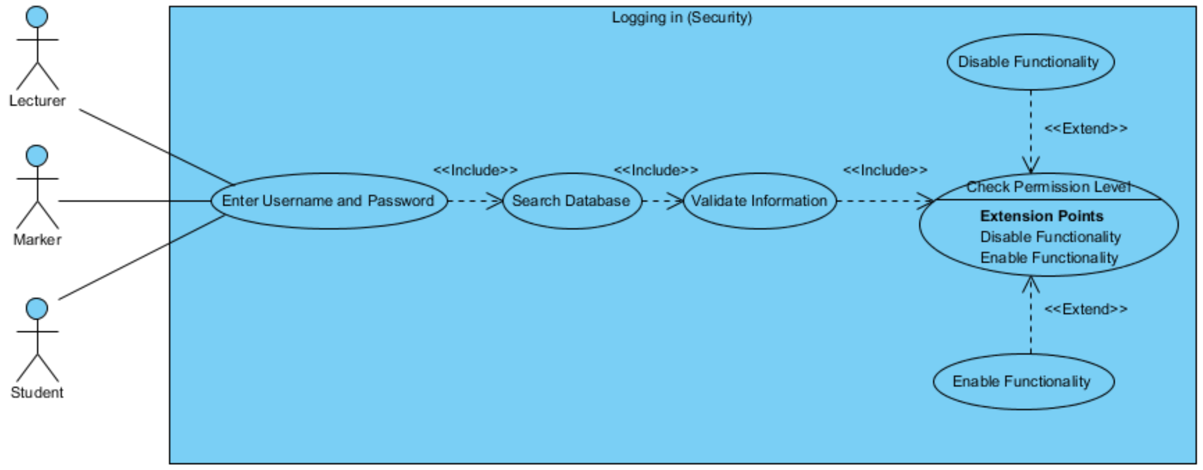
**(Source: Jan Kroeze, Priority: Important, Requirement: FRQ11)
The system must automatically log out after no activity in 60 minutes.**

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.

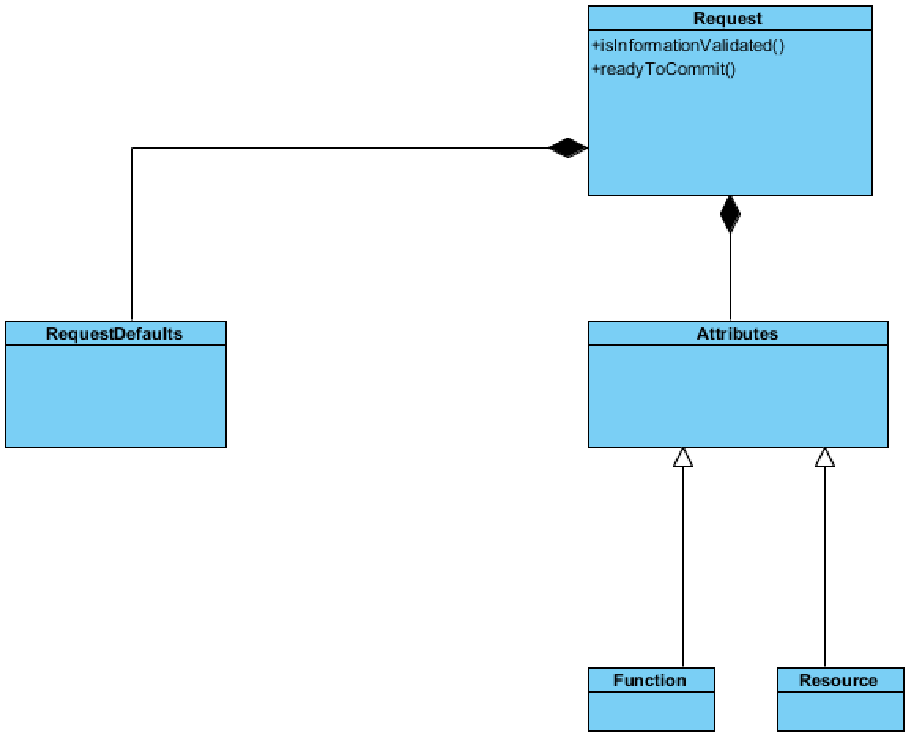
Nice-To-Have Use Cases make the system more user friendly and provides a better user experience. This case is the Notify Student use case, this is where the system automatically, emails the student of any changes made, either to their marks, namely adding marks, updating marks and deleting marks. This 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.

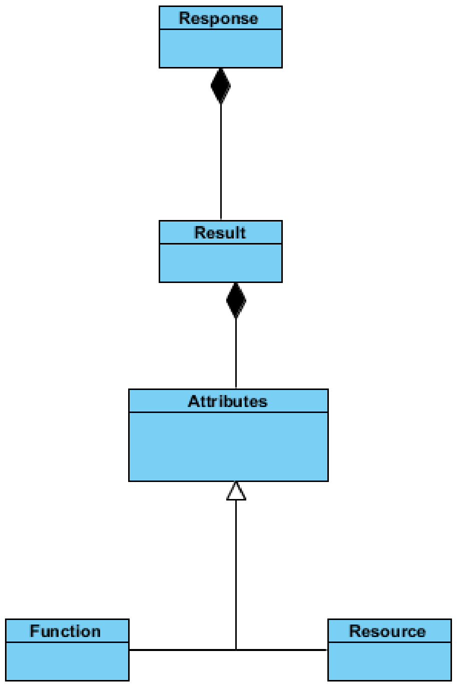
5.5 Use case/Services contracts

Pre-Conditions:
Login - User information is stored in the database. Create Task - Lecturer is assigned to the module and the user has the correct permission level. Read Task - A task is already created and the user has the correct permission level. Update Task - A task is already created and the user has the correct permission level. Delete Task - A task is already created and the user has the correct permission level. Add Mark - Lecturer or Marker is assigned to the module. Task timeslot is open. A Student is assigned to the corresponding module and task. Read Mark - Lecturer, Marker or Student is assigned to the module. Update Mark - Lecturer is assigned to the module and the user has the correct permission level. Delete Mark - Lecturer is assigned to the module and the user has the

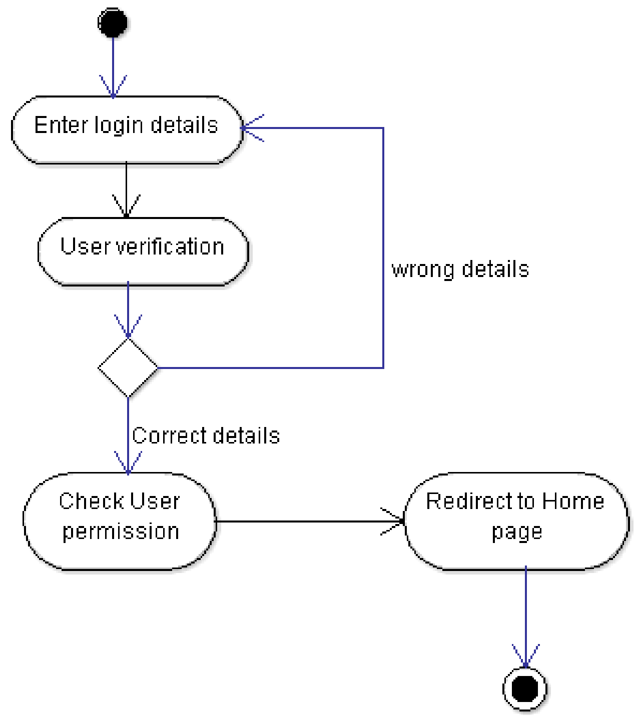
correct permission level. Generate Report - Lecturer is assigned to the module and the user has the correct permission level.

Post-Conditions:
Login - User information has been validated. Create Task - New task exists and Lecturers and Markers assigned to the module with the correct permission level is able to work with it. Students assigned to the module is notified. Read Task - Knowledge is obtained. Update Task - Task is now correct and Lecturers and Markers assigned to the module with the correct permission level is able to work with it. Students assigned to the module are notified. Delete Task - Task is removed from the system. Students assigned to the module are notified. Add Mark - Mark is added to the student's semester mark. Student who received the mark is notified. Read Mark - Knowledge is obtained. Update Mark - Student's semester mark is updated accordingly. Student who's mark has been updated is notified. Delete Mark - Student's semester mark is updated accordingly. Student who's mark has been updated is notified. Generate Report - Knowledge is obtained.


Request
+isInformationValidated()
+readyToCommit()
RequestDefaults
Attributes
Function
Resource

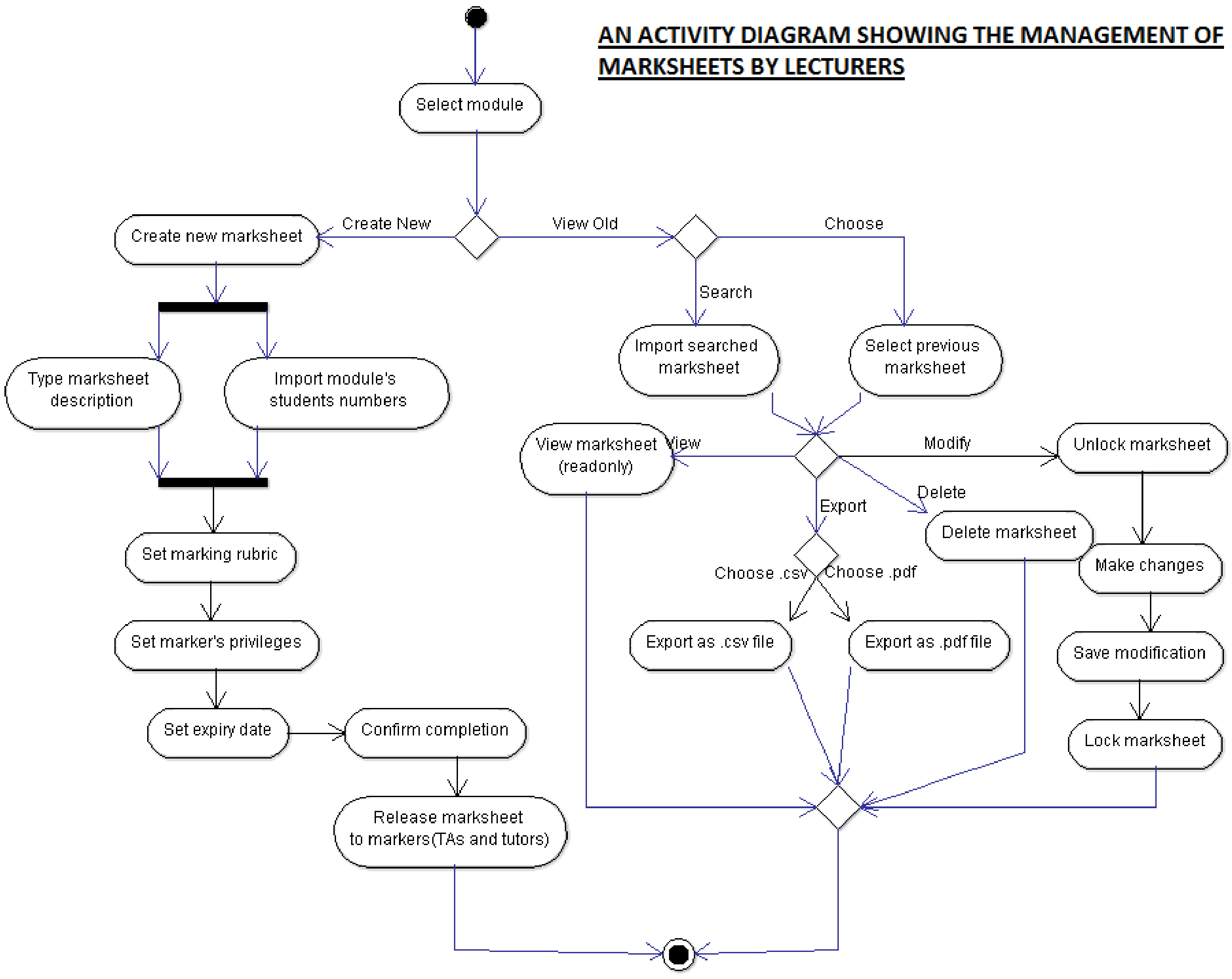

Response
Result
Attributes
Function
Resource

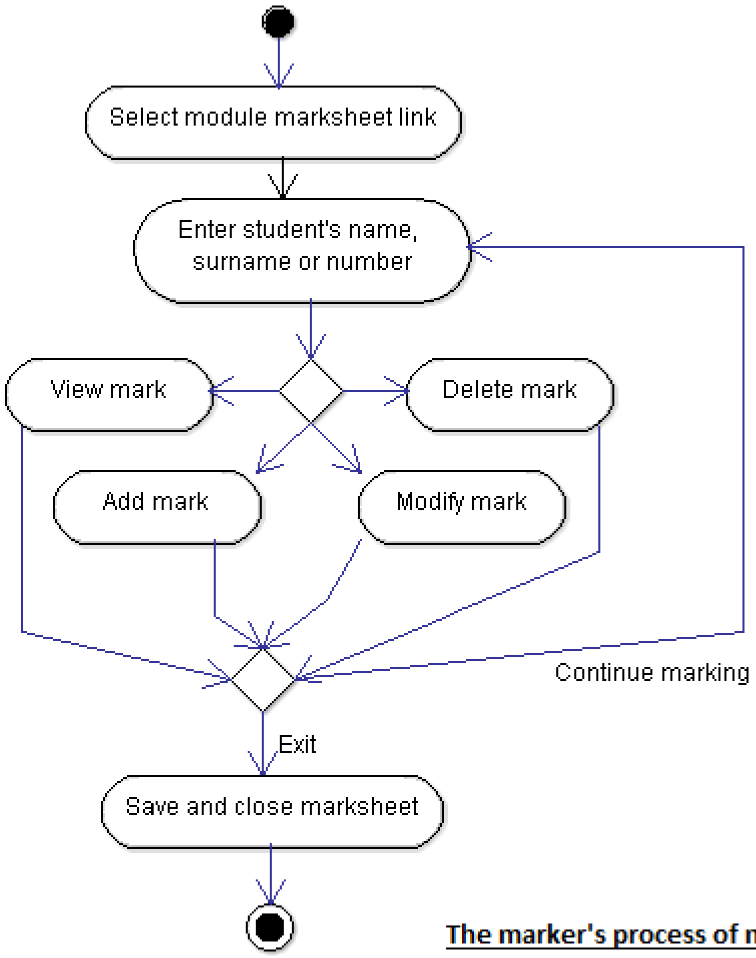
5.6 Process specifications



<u>THE ACTIVITY DIAGRAM OF THE APPLICATION'S LOG IN PROCESS</u>

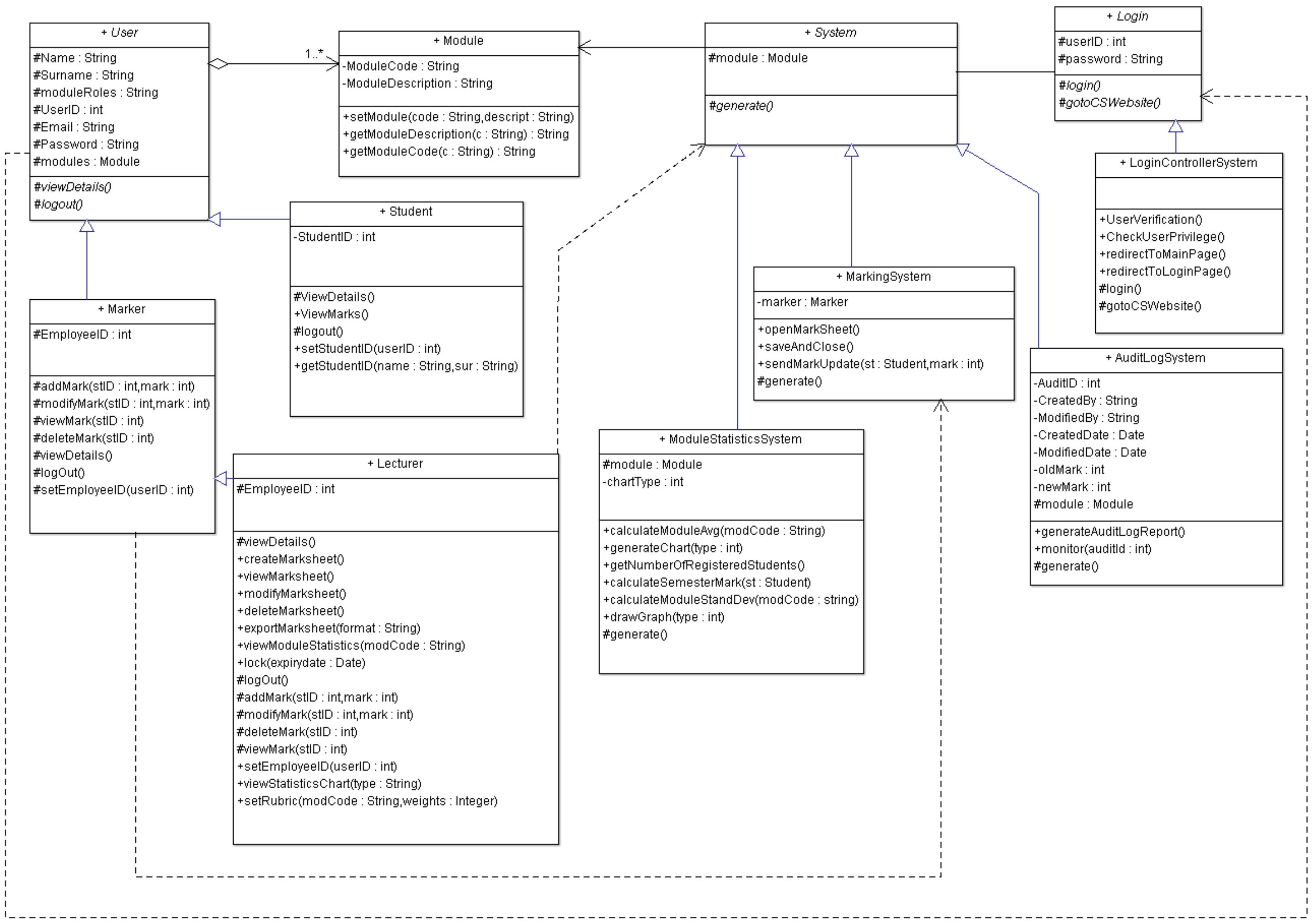
AN ACTIVITY DIAGRAM SHOWING THE MANAGEMENT OF MARKSHEETS BY LECTURERS





The marker's process of marking

5.7 Domain Objects



<u>**UML CLASS DIAGRAM FOR THE MARKING APPLICATION**</u>

6 Open Issues

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

7 Glossary

- CSS - Cascading Style Sheets
- CSV - Comma-separated values
- HTML - Hperttext Markup Language
- HTTP - Hypertext Transfer Protocol
- JMS - Java Message Service
- JSON - JavaScript Object Notation
- POP3 - Post Office Protocol
- SMTP - Simple Mail Transfer Protocol
- SOAP - Simple Object Access Protocol
- SQL - Structured Query Language
- UML - Unified Modeling Language
- XML - Extensible Markup Language