Software Requirements Specification
Computer Science Marks System

Version: 1.1

https://github.com/erosslee/COS301_MiniProject

By group 10:
Zenadia Groenewald (12265676)
Thulasizwe Mavuso (29236259)
Uteshlen Nadesan (28163304)
Zühnja Riekert (12040593)
Estian Rosslee (12223426)
Moeletji Semenya (12349136)

For:
Mr. Jan Kroeze (University of Pretoria)

February 26, 2014

Contents

1 Introduction

The purpose of this document is to specify the requirements for this project. Developers will find this useful as the content describes the vision, scope and architectural requirements as specified by the client, Mr. Jan Kroeze, from the University of Pretoria's Department of Computer Science. Also, it will help all parties that are involved to understand the implications and what the final product should be capable of. This document serves as an official contract and agreement between the developers and the client.

2 Vision

The main purpose of this project is to:

- provide the Department of Computer Science with a sustainable and convenient system by which marks may be kept track of, changed and stored in a secure and efficient manner.
- allow markers to conveniently access the system in question with their mobile devices.
- allow students to view their marks.

This project scope serves as a basis for the project to expand for various similar applications with minimal changes necessary.

3 Background

Currently students view their marks for COS modules once they are posted on the CS website. It is usually listed according to their student numbers; thus students are able to view each other's marks.

During COS practical evaluations, markers evaluate each student's work and write their marks down on paper. That then has to be passed to someone who has the time to enter the marks into the University's system. In the process of this marking system, the marks can easily get lost or be tampered with.

The aim of this project is to eliminate these threats. Marks won't have the opportunity to get lost, since they will be added to the system immediately. Students will be able to access their marks once it is added to the system and they won't be able to view any marks that are not their own.

This project is the result of the Department of Computer Science need for their own improved marking system to potentially better the administration

of students and their marks, and their decision to utilise the COS 301 courses learning opportunity for the development of the required marking system; however it is also to prepare the developers for the larger undertaking of a similar task for external benefactors and stakeholders. These, among the aforementioned, have led to the development of this project:

1. Potential business opportunities.
2. The potential for work experience and skills development.
3. The request of the clients to simplify/solve problems they are currently facing.
4. Potential experience in how project development and business procedures work.
5. The opportunity to learn how software development integrates with the business environment.
6. Eliminating a common problems experienced by lecturers and students alike with regards to mark management and administration.

4 Stakeholders

The stakeholders invloved:

- Mr. Jan Kroeze and the Department of Computer Science (client).
- Students and lecturers at the Department of Computer Science (end-users).
- COS 301 students of 2014 (developers).
- University of Pretoria -(organization).

5 Architectural requirements

5.1 Access channel requirements

The system's services are to be accessed by Android application clients and Browser clients through its web interface. These are the only two platforms that are to be supported by the system.

5.2 Quality requirements

5.2.1 Performance

- System must not take more than 1.5 seconds to return a searched for student.
- A request to view marks must be granted within 1.2 seconds.
- System must have auto complete mechanism for search services.

5.2.2 Audit-ability

- Every action on the system must be recorded on a log that can later be viewed and queried.
- Actions that are to be recorded
  - The current user performing the action.
  - New mark capturing.
  - Deletion of marks.
  - Edition of marks.
  - Reason for editing the marks.
  - The date/time on which an action was performed.

5.2.3 Scalabity

- The system must be able to be accessed by up to the current number of students registered for the course at any point.
- Additions and updates must be available to all the TA's when a mark sheet is unlocked.

5.2.4 Authorization

- All actions are granted to a user based on the privileges they have.

5.2.5 Authentication

- The system must have an access control mechanism.
- Access control must be handled at log in/out points.

5.3 Integration requirements

5.3.1 MVC component communication

- The system will be built according Model-View-Controller(MVC) architectural design pattern.
  - Model: The database with information about the system.
  - View: The web interface and Android application.
  - Controller: The SOAP interface.
- These components will need to communicate with each other and share information(using XML).
- Protocols that could be used:
  - SOAP(for structred information exchange in a network)
  - HTTPS(SOAP is going to rely on this protocol)
- Quality Requirements
  - Performance: Communication should happen in real-time.
  - Reliability: Messages sent between components should be translated and structured correctly.
  - Security: Messages should go from one component to another in the safest and simplest manner. item Auditability: All communication between components should be recorded on an audit-log.

5.3.2 The Department of Computer Science website

- Students and lectueres would only need to sign-in onto the CS website and they will automatically be signed-in onto the systems web-interface in order use the functions in 6.2.1.
- Protocols that could be used:
  - LDAP(To help provide feature mentioned above)
- Quality Requirements
  - Security: Students or lecturers who cannot sign-in onto the CS website will be denied the services offered by the web-interface.

5.3.3 Fitchfork auto-marking system

- The marking system would need to have an interface for communicating with Fitchfork in order to get marks.
- Protocols that could be used:
  - TCP/IP(To specify how the two systems will communicate with each other)
  - TSL/SSL(To provide security over the network when the two systems communicate)
- Quality Requirements
  - Performance: The speed at which the two systems communicate should be realistic so when a practical auto-locks the marks are available for students to view.
  - Security: Information communicated should not be intercepted or sniffed by user in the system.
  - Reliability: The connection between systems must maintained when communicating.
  - Auditability: Information on both the systems must correlate and be recorded.

5.4 Architectural constraints

5.4.1 Technologies to be used

- For the application platform
  - Android SDK
  - Java
- For the web interface
  - JavaScript
  - HTML 4.0/HTML 5
- Must be secure therefore must run over HTTPS
- Server side
  - Django framework
  - Python
- Web service

  - SOAP

- MySQL for database
- LDAP for distributed directory information service

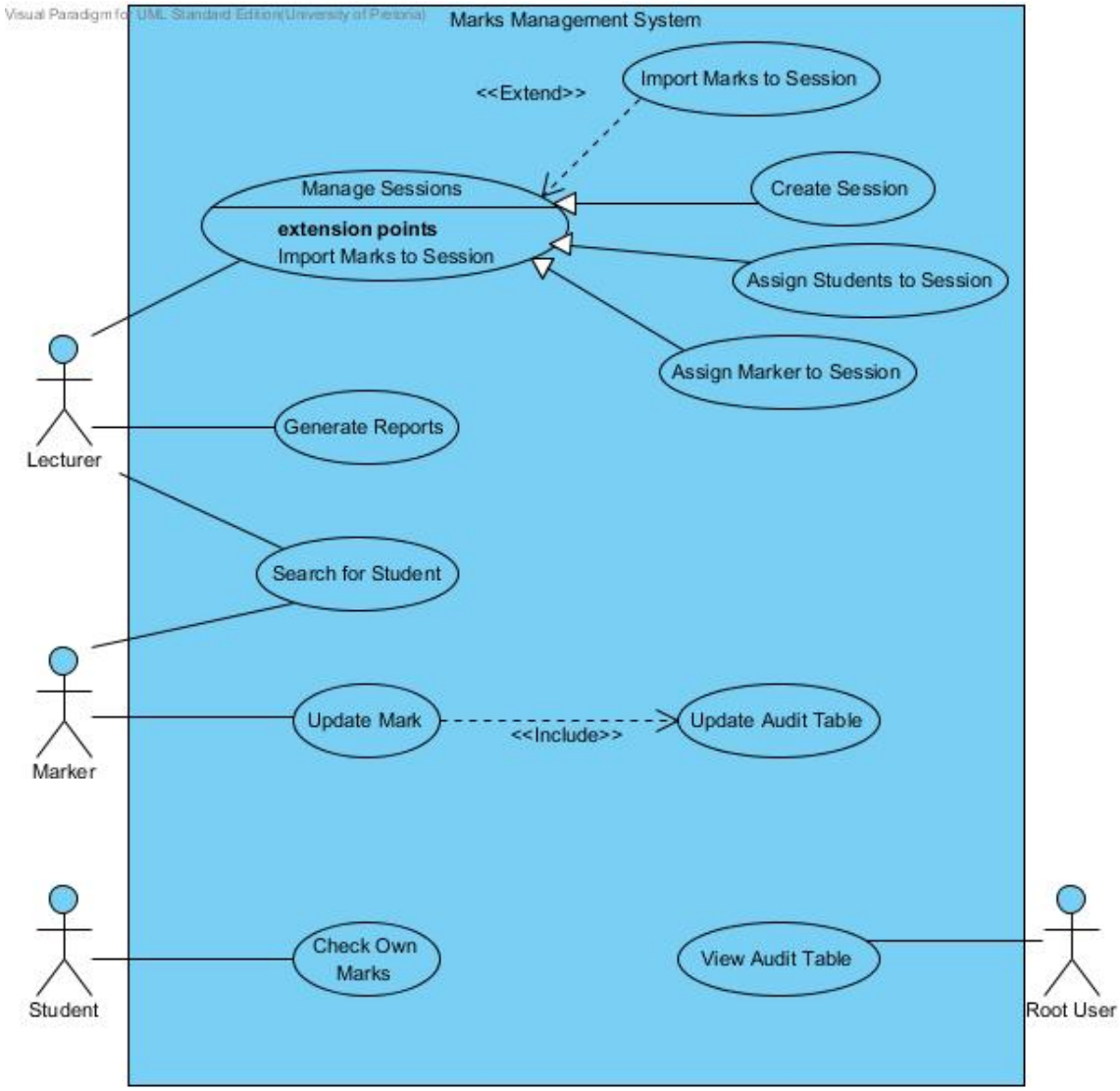
6 Functional requirements

6.1 Introduction

The function of this project is to design an application interface that would enable a marker to upload the mark of a student, in a practical session, with his/her phone.

6.2 Scope and Limitations/Exclusions

High Level Use Case Diagram:



6.2.1 Scope

Phone interface for the marker

- Markers should only be able to view students who have registered for the specific practical period the marker is assigned to.
- On specific markers with specific credentials should be able to log on to the application.

- Markers can only log onto the application for their specified time slot and the application should run a check to provide this.
- The marker should be able to look up the student being marked on his/her phone with either a student number or a first, last or both the first and last names of the student.
- The interface should be simplistic and show the rubric and mark layout and have input boxes for the different mark fields.
- For a change of a mark, the marker must provide a reason in the textbox that is saved for future reference.

A web interface is also needed for lecturers and students

Lecturers:

- Can only view records for courses they are in charge of.
- Lock and unlock the database from being edited.
- Need to be able to view records of marks as well as specific marks.
- PDF documents for Lecturers should be generated detailing marks and descriptions.
- Graphs of students marks should also be generated in the form of normal distributions, bar graphs and pie charts.
- Counts of students should be available, as well as students registered standard deviations, averages and medians.

Students:

- Students should only be allowed to view their own marks and check their progress in the course thus far as well as overall progress with the course.
- The student should not be able to edit marks.

6.2.2 Limitations/Exclusions

- No student should be able to view any mark besides their own.
- The application should only be programmed for android smart phones.

6.3 Required functionality

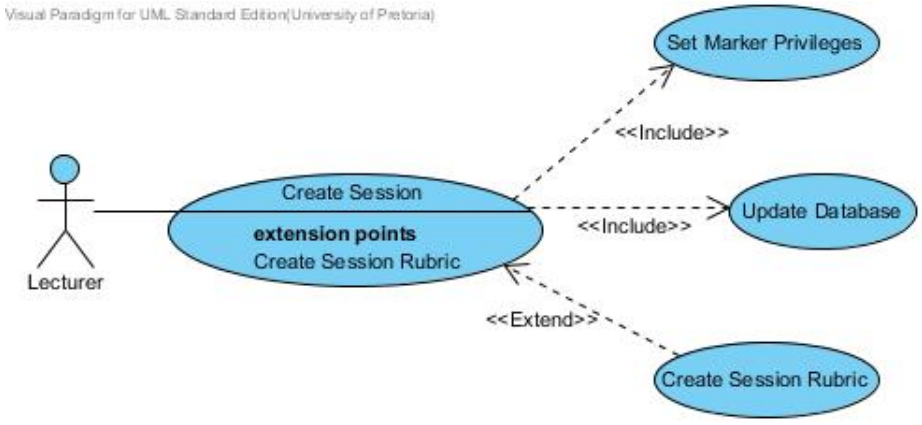
This application should be:

- Accessible on android phones.
- Scalable and viewable on devices.
- Secure and information sent and received should be encrypted.
- Auto log out after a specified time.

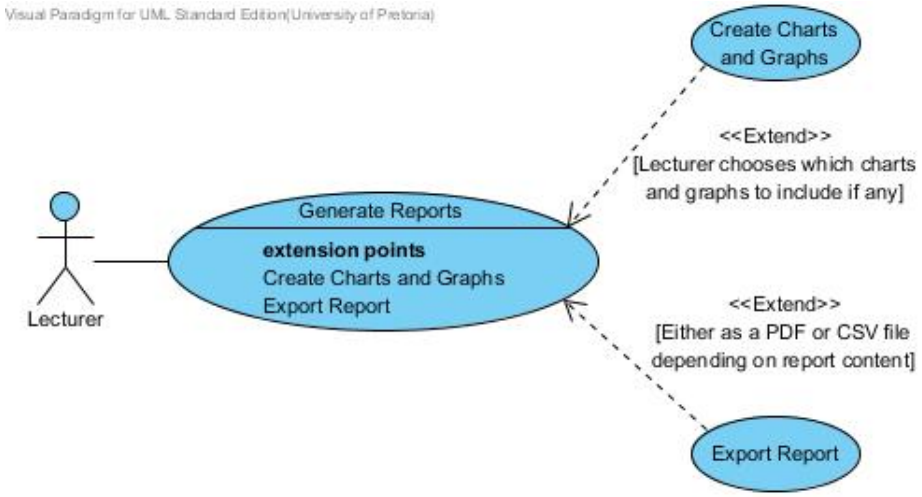
The application:

- Should have a database containing all students, lecturers and administrators as well as markers and a root user.
- The root user should have access to all fields of the database.
- Should integrate with the automatic marker Fitchfork to receive specific student marks and be able to add these marks to a students mark database.
- Should auto-complete names and student numbers being entered by markers.
- Generate CSV files for viewing and be able to accept CSV files of marks for input into the database of students marks.

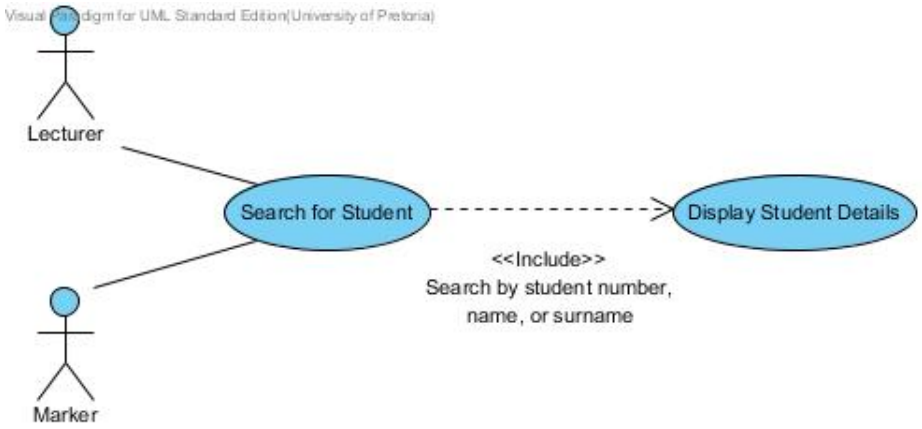
Create session Use Case diagram:



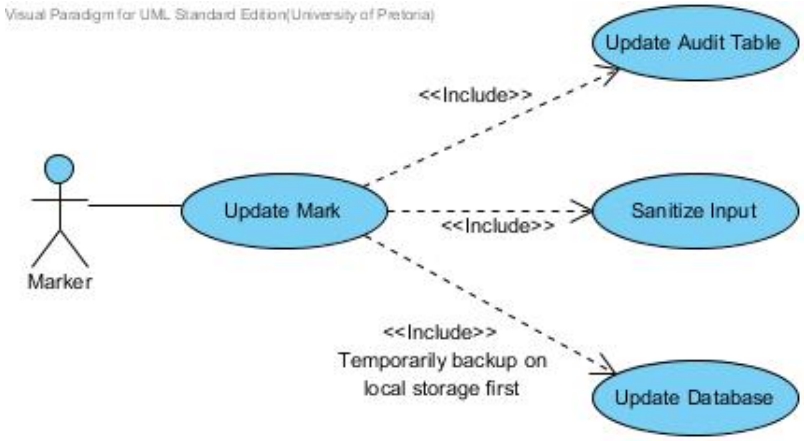
Generate reports Use Case diagram:



Search for student Use Case diagram:



Update mark Use Case diagram:



6.4 Use case prioritization

6.4.1 Critical:

- The application needs to be functioning on the Android Platform and should be available on all Android phones for Android Version 2.0 upwards.
- Marks must be able to be entered and uploaded to the server.
- Have a web interface for viewing marks for students.
- Have a web interface specific for lecturers.
- Fitchfork integration.
- Auto-lock the marks database from being edited 30 minutes to an hour after the practical session.

6.4.2 Important:

- Good to exceptional performance speeds.
- Encryption of communications.
- Have a root user with unlimited access to the database and application. (The root user cannot edit audit-able documents)
- Generated PDF documents with detailed reports.

6.4.3 Nice-To-Have

- Auto-Complete.
- Auto log off.
- Student search via first name and/or surname.

6.5 Use case/Services contracts

Pre-conditions and Post-conditions

6.5.1 Search for student

Pre-conditions

- The marker or lecturer doing the search must be logged in.
- Student must be in the correct marking session.
- Valid student number should be entered or
- Student name and/or surname should be entered correctly.

Post-conditions

- The student is successfully found.
- The student's details are displayed.
- Option for a marker to update a mark is displayed.

6.5.2 Update student mark

Pre-conditions

- The student should be successfully found.
- A valid reason for mark update is provided.
- Valid mark entered.

Post-conditions

- Mark successfully updated to marks database.
- Audit log successfully updated to reflect the updated mark, the time of the update and the personnel number of the marker.

6.5.3 View marks

Pre-conditions

- Student has to be logged in.

Post-conditions

- Student can successfully only view own marks.
- Mark sheet being viewed remains locked to the student so that the student can not edit any marks.

6.5.4 Login

Pre-conditions

- User credentials must be valid

Post-conditions

- User has access to the functionality of the application as determined by their role (Student, Marker, Lecturer, Root User).

6.5.5 Logout

Pre-conditions

- User has to be logged in
- (Automatic logout condition) No action has been taken by the user in the last 5 minutes

Post-conditions

- User successfully logged out.

6.5.6 Edit mark/assessment session

Pre-conditions

- The lecturer should be logged in.

Post-conditions

- Successfully made changes to the mark/assessment session.

6.5.7 Create mark/assessment session

Pre-conditions

- The lecturer should be logged in.

Post-conditions

- Successfully created mark/assessment session in marks database.
- Mark/assessment session available for managing by the lecturer.

6.5.8 Create report

Pre-conditions

- The lecturer should be logged in.

Pre-conditions

- Report successfully generated for lecturer viewing.
- Option given to export the report as either a PDF or CSV file.

6.5.9 Export marks

Pre-conditions

- The lecturer should be logged in.
- Mark sheets from which marks are to be selected are locked.
- The lecturer selects which marks to export.

Post-conditions

- Marks successfully exported as a CSV file.

6.5.10 Import marks

Pre-conditions

- The lecturer should be logged in.
- Mark/assessment session for which the marks are intended must already exist.
- File to be imported must be in CSV format.

Post-conditions

- Marks successfully updated to the mark sheet of the relevant mark/assessment session in the marks database.

6.5.11 View audit log

Pre-conditions

- Root user should be logged in.

Post-conditions

- Root user is able to view the audit log.

6.5.12 Assign marker to mark/assessment session

Pre-conditions

- Lecturer has to be logged in.
- Session should already exist.

Post-conditions

- Marker is successfully added to the mark/assessment session.

6.6 Process specifications

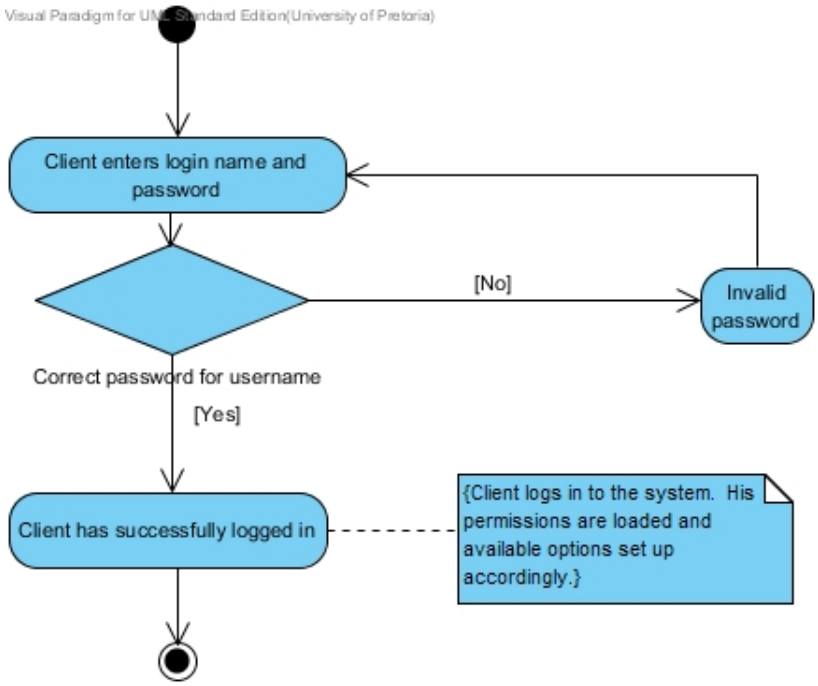


Figure 1: Login:

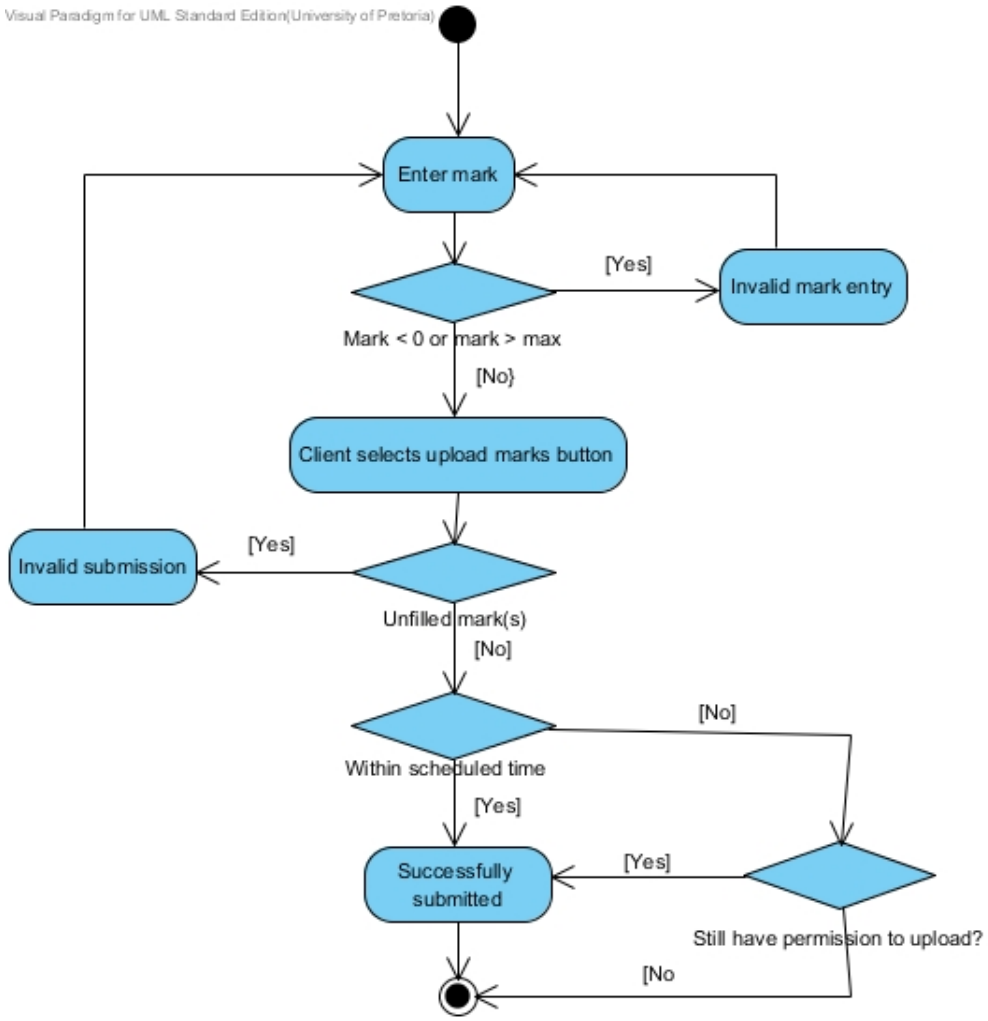


Figure 2: The marking process:

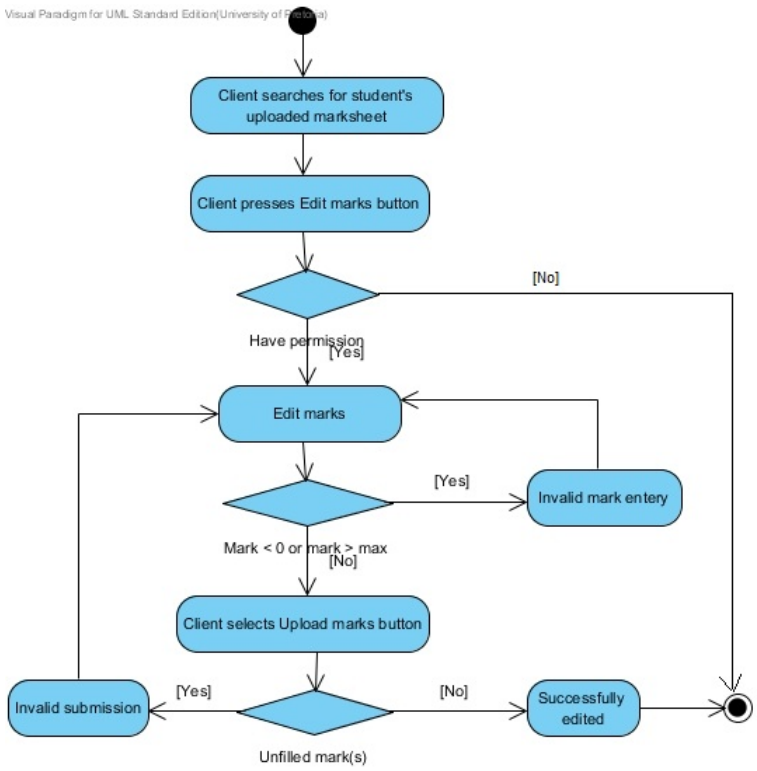
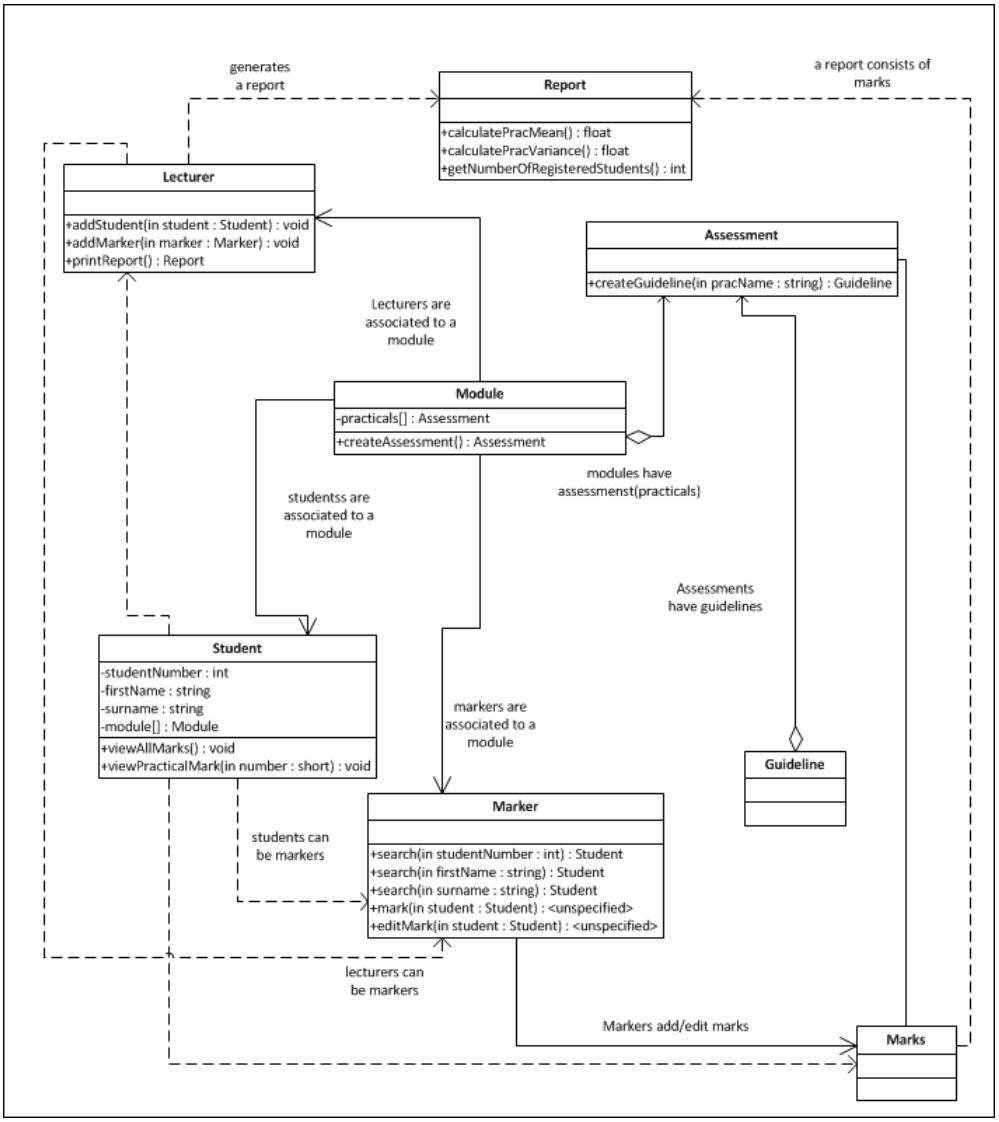


Figure 3: Edit an uploaded mark:

6.7 Domain Objects



Above is a class digram of the system with classes that contain attributes and methods in order to make the relatioships easier to comprehend.

7 Open Issues

- A marker may not own any mobile devices.

8 Glossary

- Android - Smart phone operating system to be programmed for
- Authentication - to establish user as genuine

- Authorization - permission or power granted by an authority
- Client - a workstation on a network that gains access to central data files, programs, and peripheral devices through a server
- COS - Computer Science (Courses)
- Database - Web framework
- Developers - The programmers of the application
- End-users - Clients and/or the actual users of the interface
- Evaluators - The markers in the practical session
- Fitchfork - Auto marking system used by the Department of Computer Science
- HTTPS - HyperText Transmission Protocol, Secure
- Interface - The part of the application that the end users will be using the application through
- Java - high-level, object-oriented computer programming language used especially to create interactive applications running over the Internet
- JavaScript - an embedded language run in web browsers
- LDAP - Lightweight Directory Access Protocol (Database of use for the Department of Computer Science)
- MySQL - The most popular open source relational database management system
- PDF - a file format that makes it possible to display text and graphics in the same fixed layout on any computer screen
- Scalability - The ability of something, especially a computer system, to adapt to increased demands
- SOAP - originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks