Software Requirements Specification
Squirrel Marking System

Version: 1.0

Organization:
University of Pretoria: Group 3
GitHub:

https://github.com/Roach-301/CS301_Group3

Authors:
Johan Esterhuyse (10043283)
Tokologo Machaba (12078027)
Heelin Mistry (10299344)
Pieter le Roux (1045486)
Rudiger Roach (11004322)
Thulasizwe Mavuso (29236259)

March 13, 2014

Contents

1 Software architecture design

1.1 Choices of technologies

The existing web server runs on Apache, therefore our system will continue the use thereof.

Django web server, running on Apache, will be our interface to run on.

Djangos Object Relational Mapper Will be used to persist information to the database.

Django unittest module will be used to perform unit tests with. MySQL server allready runs on the CS server and this existing database will be used to pull information from and store information to.

The android application will be built using JAVA.

1.2 Chosen frameworks

Django application server will eventually be the framework onto which the system will be deployed and Djangos bundled Object-Relational mapper will be used to access the database.

1.3 Chosen protocols

To send data between objects, it will first be encoded into JSON strings. JSON is an easy to use standard that can be parsed by most programming languages. This will enhance our ability to keep layers seperate from each other and just pass JSON strings between layers.

1.4 Chosen libraries

PDF creation will be done using iTextPDF from itextpdf.com. iTextPDF is open source with implementations on multiple different platforms. It is also licensed to enable re-use by developers.

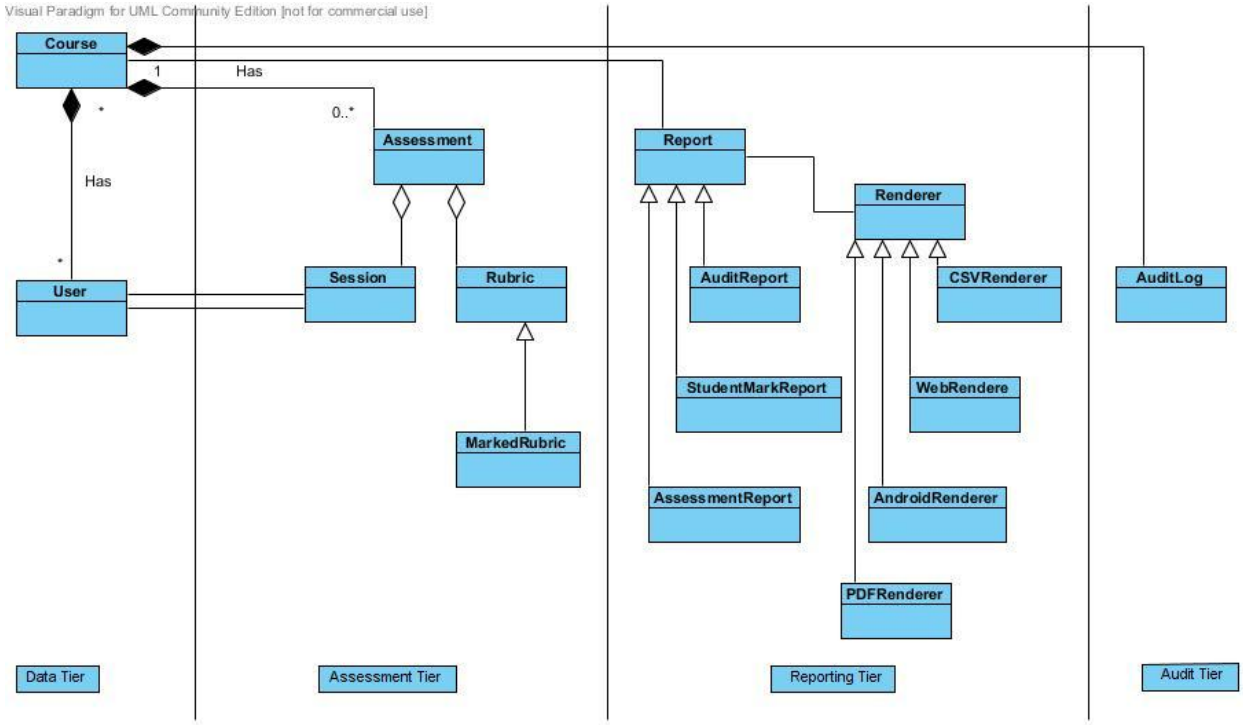
PDF rendering will be done by using the PDFrenderer library from https://github.com/katjas/PDFrenderer. This library is LGPL-2.1 licensed and therefore fine for use by us. The library is written in JAVA and uses JAVA2D to render PDF documents.

To marshall and de-marshall JSON, Apache camel will be used from http://camel.apache.org/. This library is fully compatible with java and independent from the transport used.

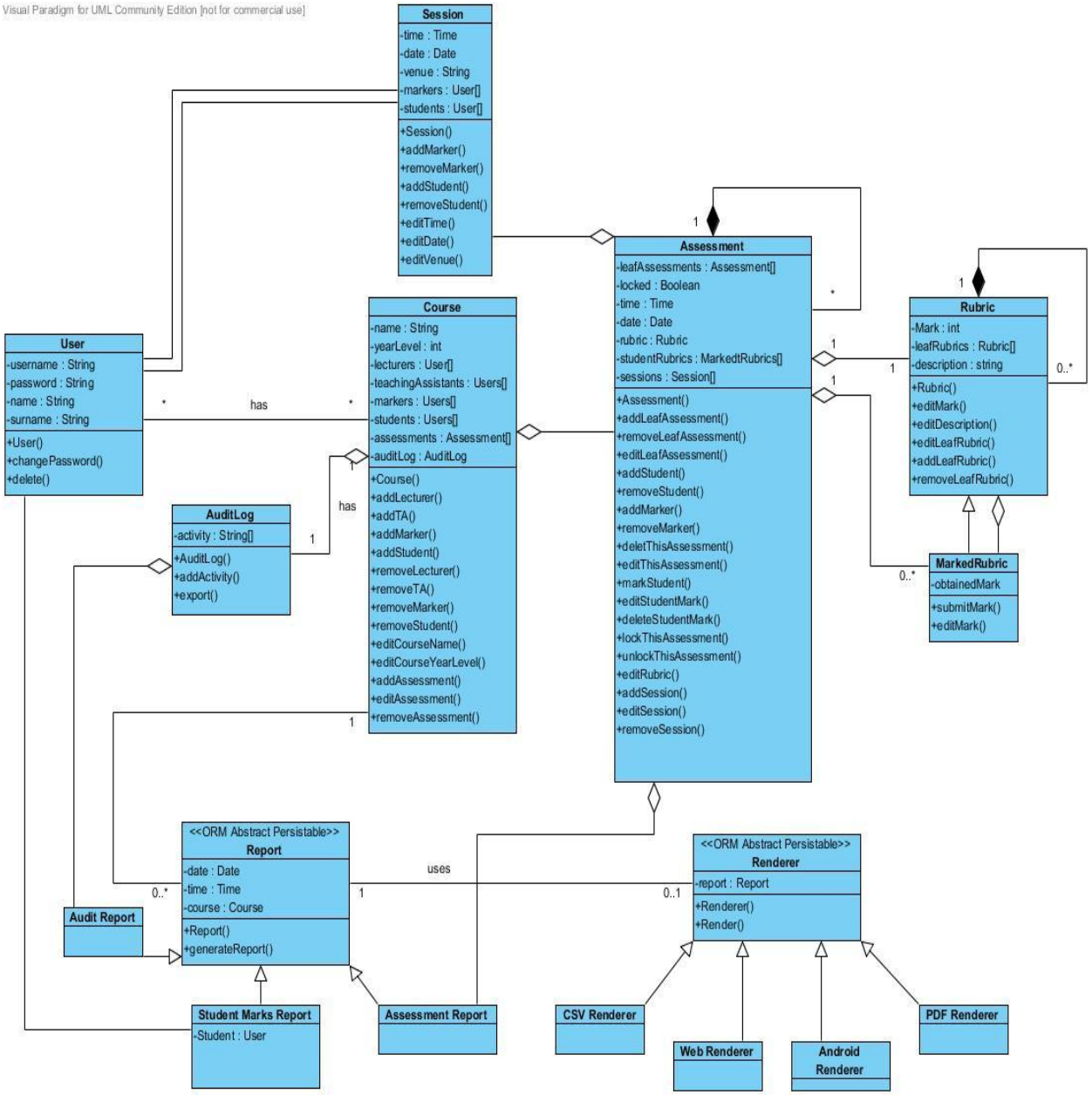
2 Application design

2.1 Back-end

API:



Class Diagram:



2.2 Web Application

- When the user directs to the system through one of the widely used browsers, a login screen is displayed.
- The login screen will be developed with HTML 5 and CSS3
- It will consist of textboxes for the username and password and a button to send the information to the server.

Authentication Level

LDAP Authentication:

- The system accesses the CS LDAP to validate and grant privilege to the logged in user.
- When the user has been authenticated, pages are sent over to client browser over HTTPS

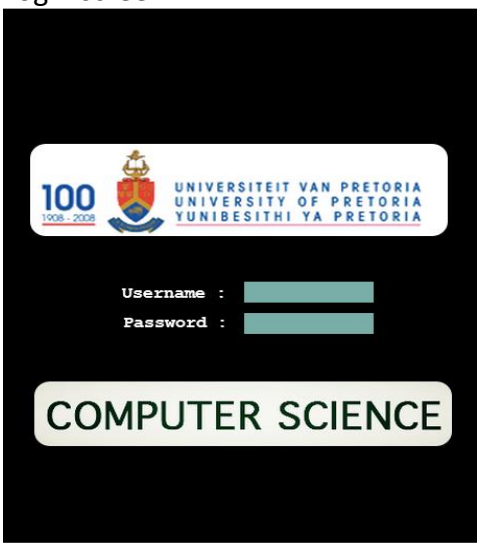
Home Page Level

Possible actions according to privilege:

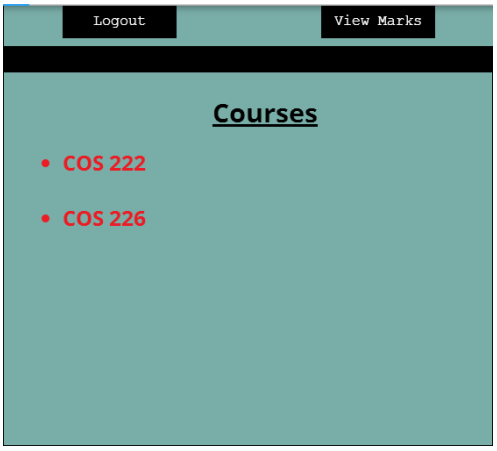
- A student will be directed to a page where he/she will only be able to view leaf and aggregate marks.
- Markers will be presented with options to either:
    - Create an assessment
    - Edit marks for a specific student through a search interface
    - Delete marks and a text-area for the reason for doing so.
    - Changes committed will be displayed immediately through the use of JavaScript and JSON.
- Lectures will be able to:
    - Manage information about the course through an easy to use interface that will make use of JSON to send data back and forth.
    - Manage assessments:
        - The lecturer will determine how marks are aggregated.
        - Lecture will decide which assessments are the leaf assessments.
        - He will specify how much each leaf assessment contributes to the aggregated mark.
    - Generate report:
        - The report can be generated into either a PDF or CSV at any level of granularity.

2.3 User Interface Design

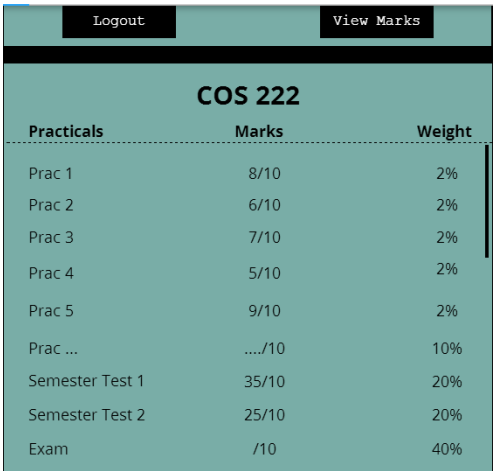
Login screen:



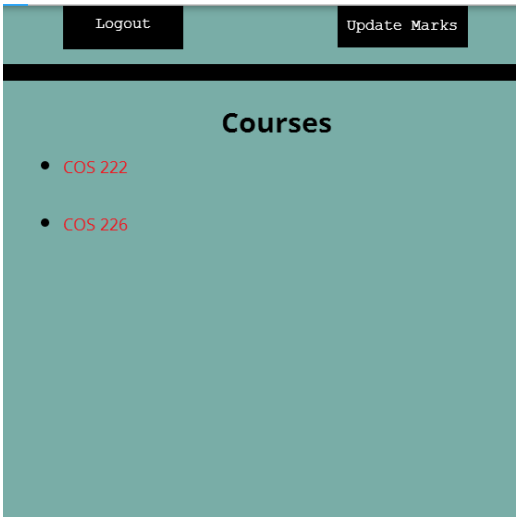
Students Menu:



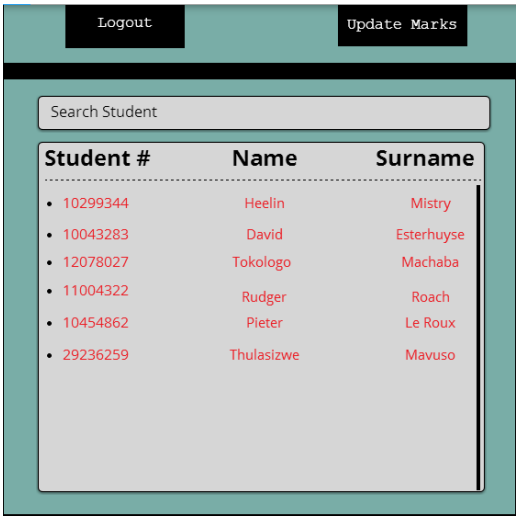
Students Marks:



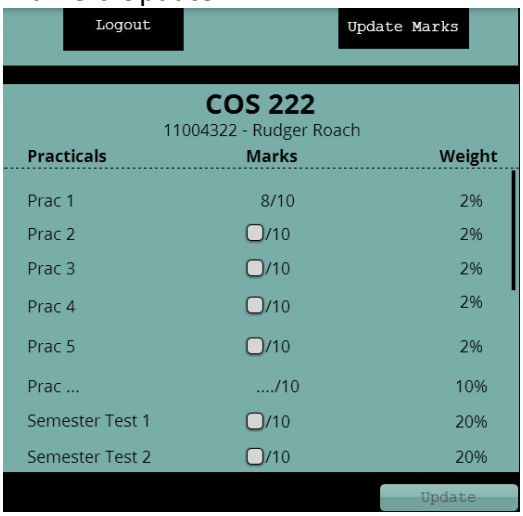
Markers Menu:



Markers Select:

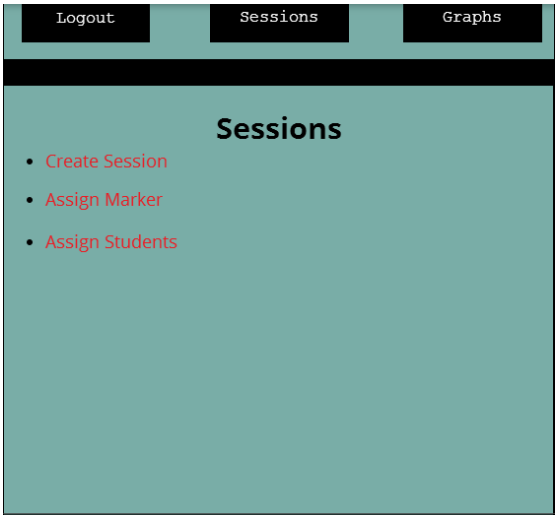


Markers Update:

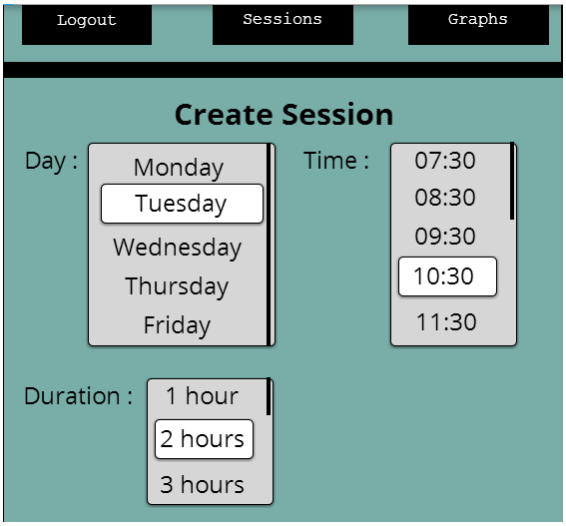


Lecturer Menu:

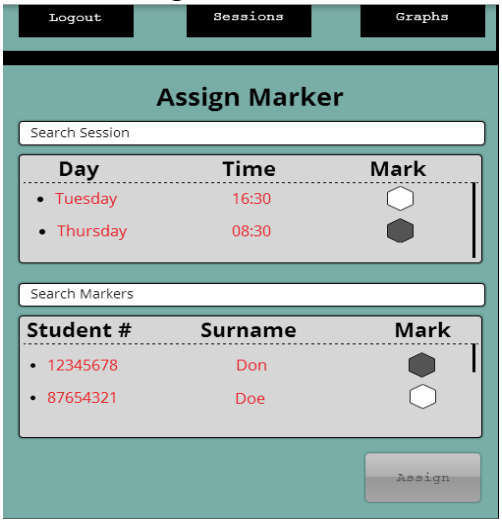
Lecturer Session:



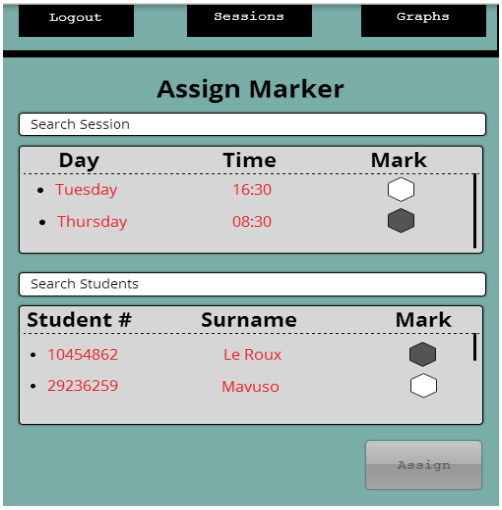
Lecturer Create Session:



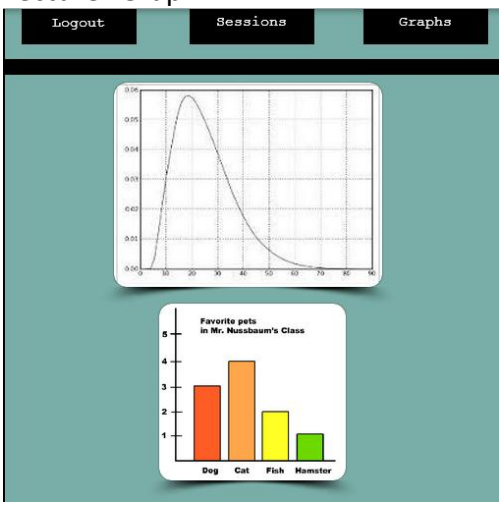
Lecturer Assign Marker:



Lecturer Assign Student:



Lecturer Graph:



Database Design The Database will be centralised to one simple design which will be used by all aspects of the system wherever Database access is required.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
code	varchar(20)	YES		NULL	
name	varchar(255)	NO			
lecturer	varchar(255)	NO		0	
description	text	YES		NULL	
semester	smallint(6)	NO		0	
has_webct	tinyint(4)	YES		NULL	
year_group	int(11)	YES		NULL	
hidden	tinyint(3) unsigned	NO		0	
last_updated	datetime	NO		0000-00-00 00:00:00	
discussion_board	tinyint(4)	YES		NULL	
tutors_allowed	tinyint(2)	YES		NULL	

3 Application layer Level

3.1 Protocols to be used

- SOAP interface will be used through XML unmarshalling when accessed by other systems.
- The application will run over, thus SOAP will run over HTTPS for a secure connection.

3.2 Android Application

The specifications of lower levels of granularity.

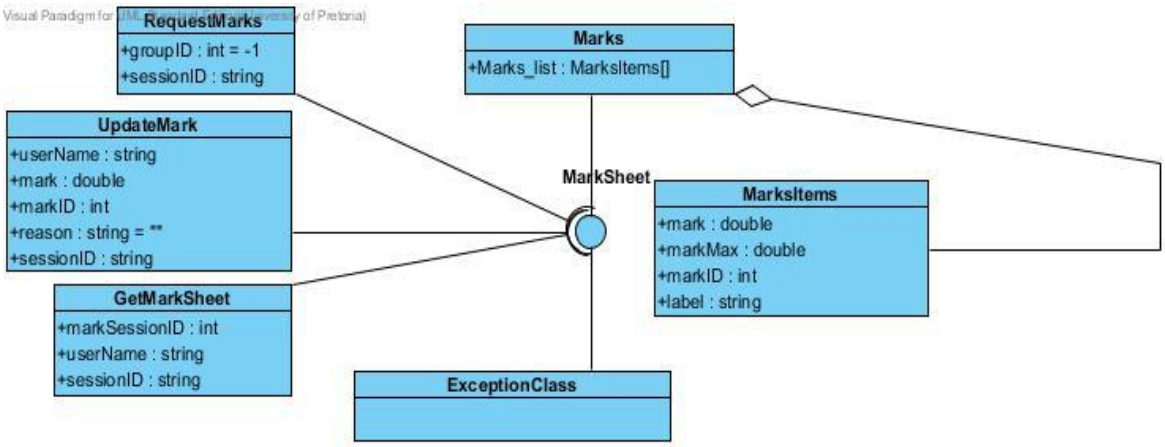
Users will also have to be able to login to the application using their CS details. Users that should be able to use the application are markers, lecturers and students (who have android smart phones).

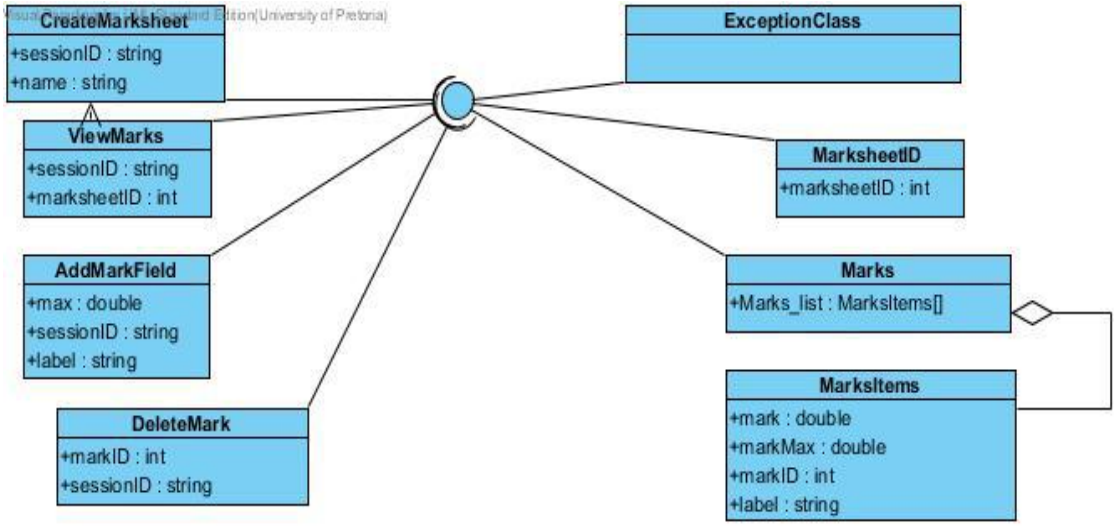
- Students:
    - Students should be able to check their marks with the android application.
- Markers:
    - After logging in markers will be able to insert the marks of students that have been assigned to them, for the subject they are marking. They should only be able to insert marks during the practical session (unless otherwise stated).
    - Markers can edit student marks but only for a reasonable circumstance.
    - Markers can search for students by student number, name and surname.
    - Marks will be saved locally between updates to the server, so if anything happens to the application or the phone information will be saved, therefore no need to reinsert data already done
- Lecturers:
    - Lecturers should be able to login and choose between a menu of creating a session or viewing marks.
    - When creating a session, a day, time and duration of the session should be assigned
    - When assigning students and markers to session, they can be searched for by using student numbers, names and surnames. Multiple students can be assigned to a session by marking the students with a radio button.
    - After assigning students and markers information should be saved locally between updates to the server.

- The application:
    - The system should log out after no more than 10 minutes of not being used.
    - The mobile uses MySQL database which it interacts with via the server.
    - Updates between the application and the server should be saved locally on the mobile memory.
    - When a mark is sent to the system it should be checked to see if it was actually inserted, if not it should try again, if repeated failures are encountered an exception should be thrown telling the user to either retry or try to upload the marks later.
    - Mark sheets should lock after the practical session has ended, so that no more marks can be inserted (unless otherwise stated).

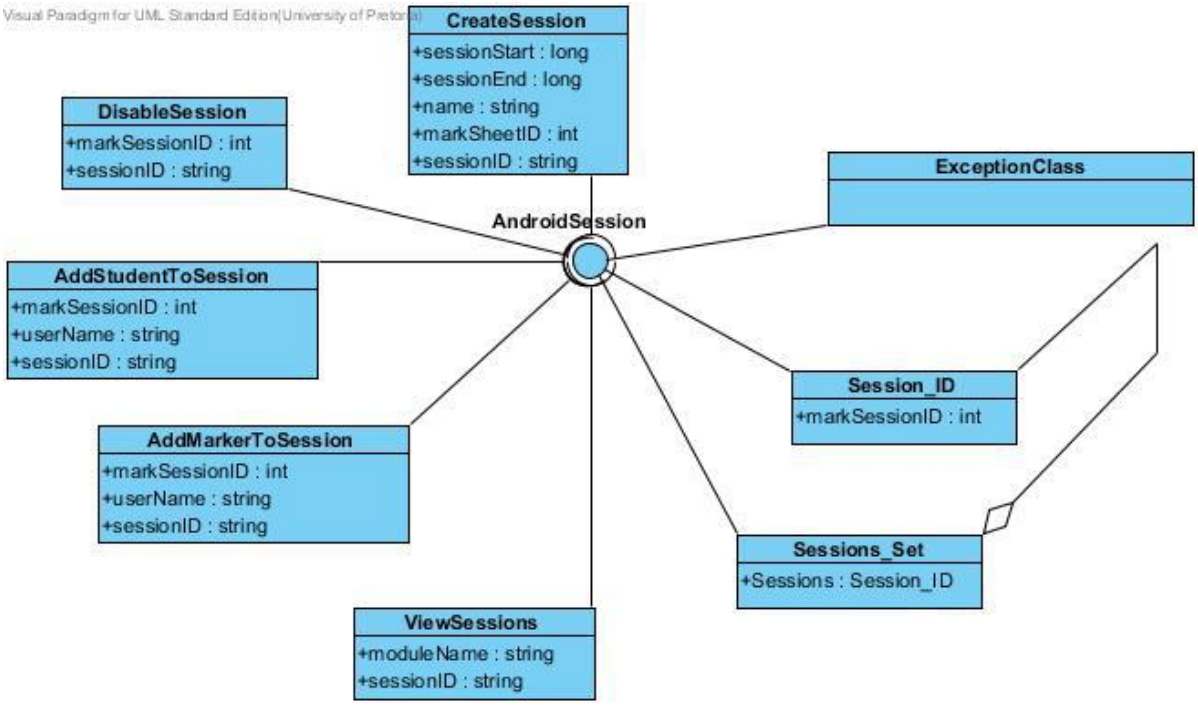
API Specifications

In the Android application the markers will be handling the marks as shown below:

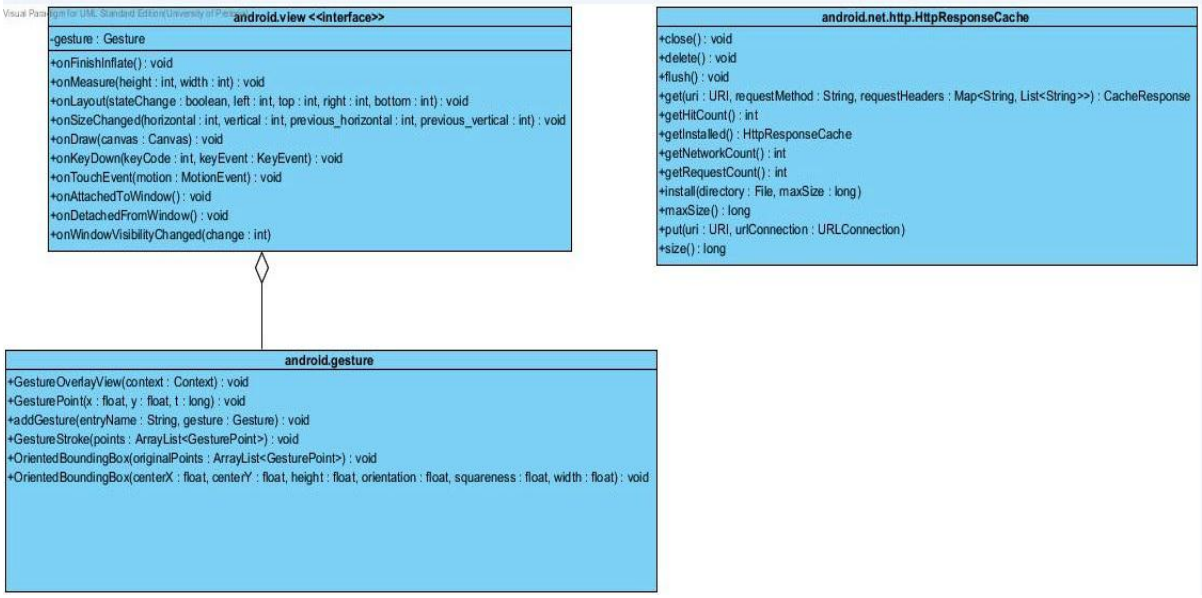




The android application will work according to sessions and whether or not they are open.



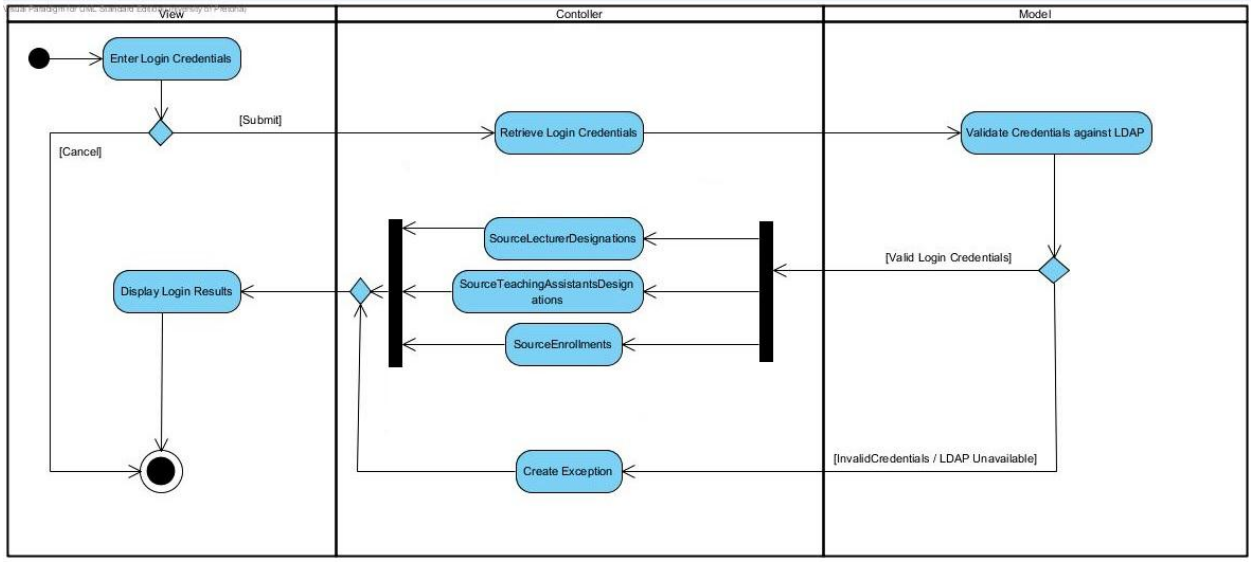
Class Diagrams The Android Application will have the following structure:



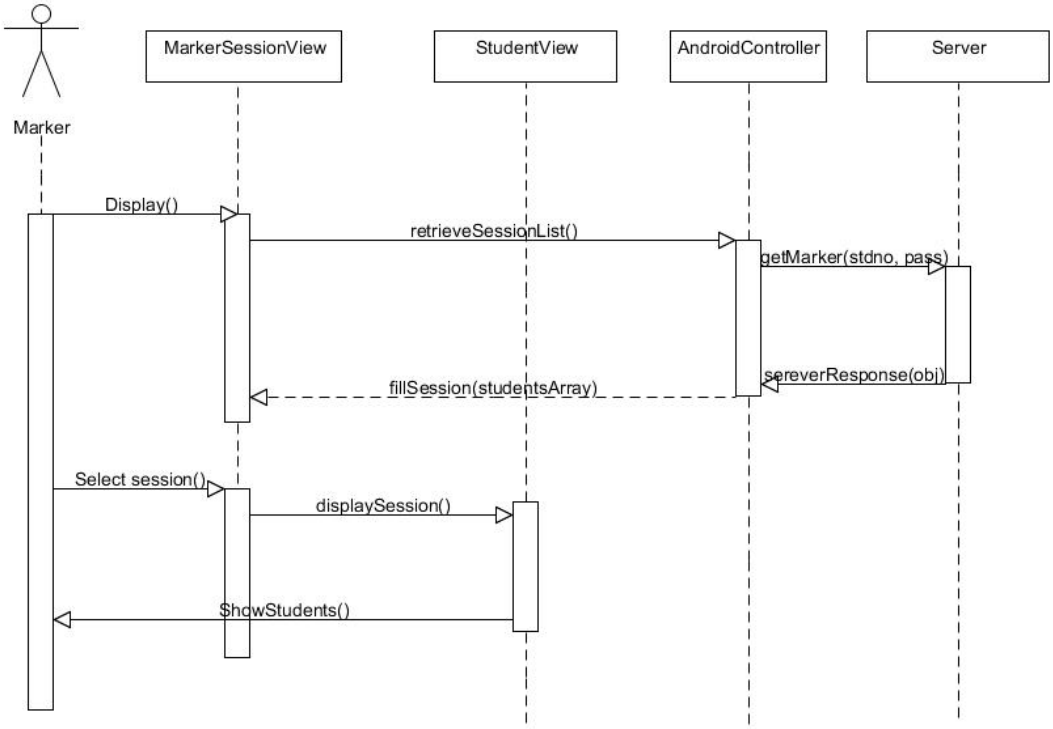
Source: http://developer.android.com/

System Process Specifications

Login activity Diagram:

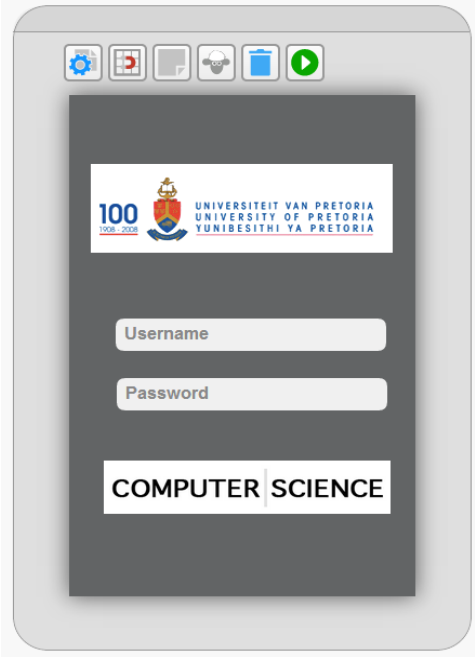


Marker's View Point:

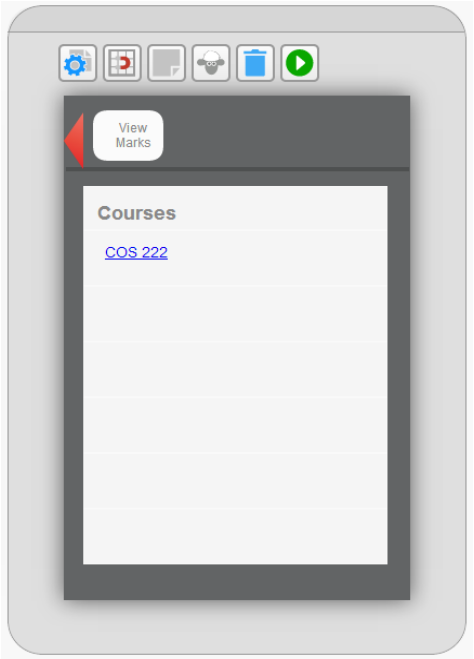


User Interface Design:

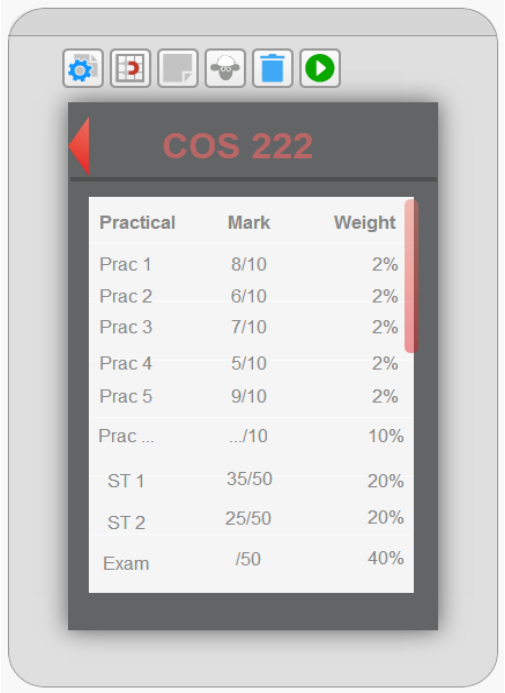
Login screen:



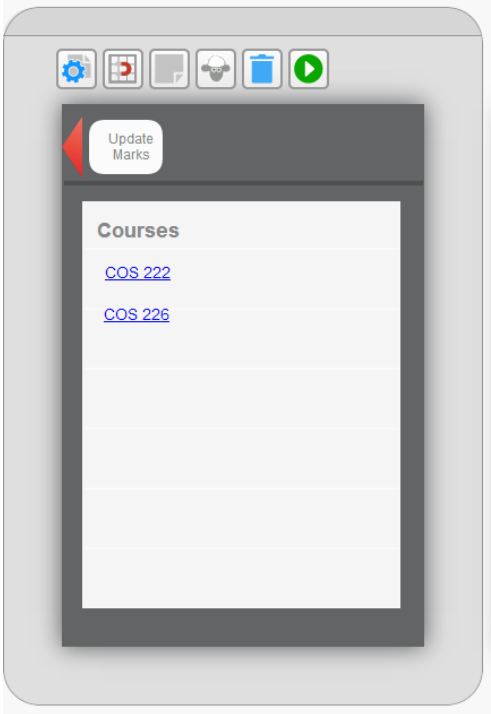
Students Menu:



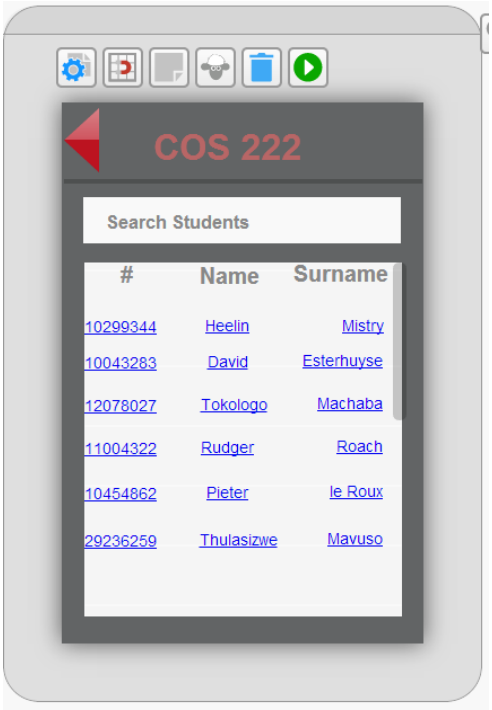
Students Marks:



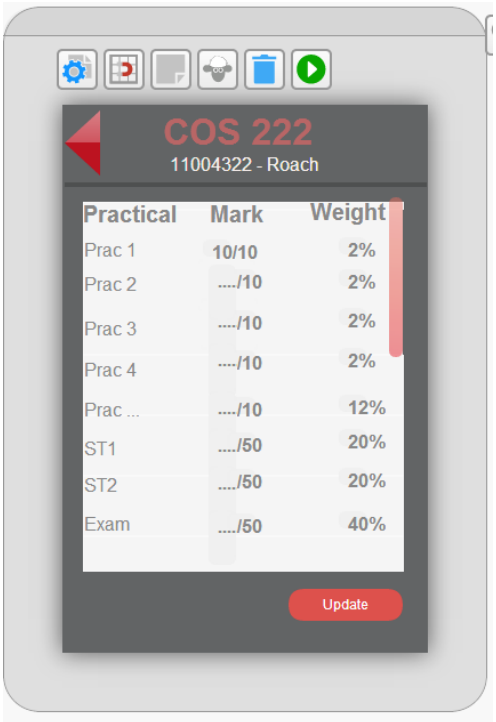
Markers Menu:



Markers Select:

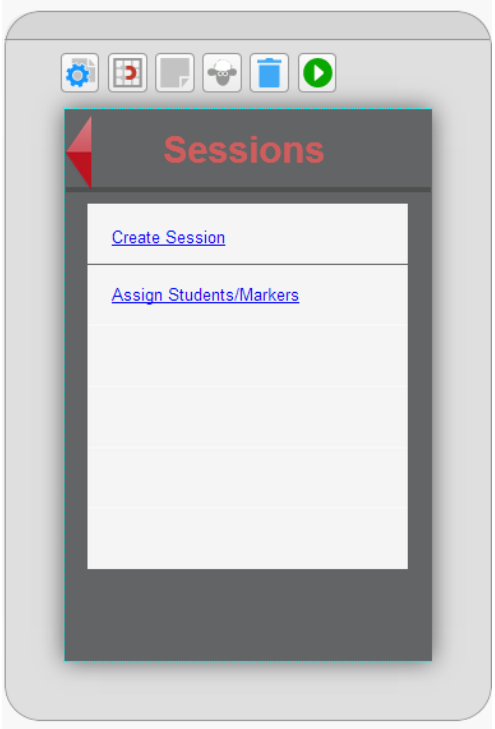


Markers Update:

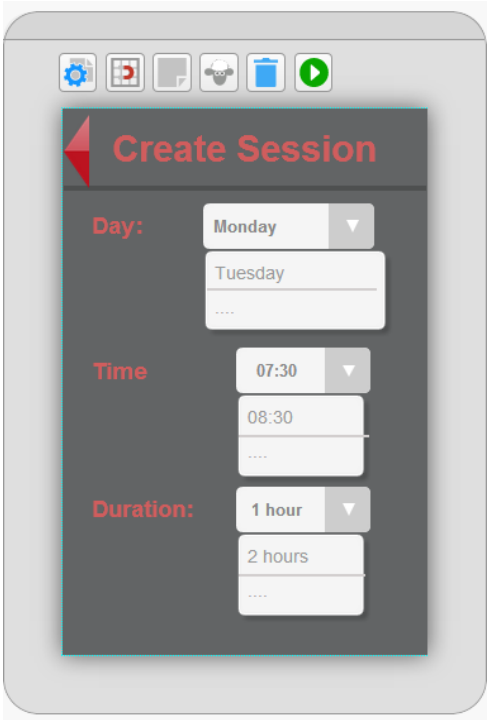


Lecturer Menu:

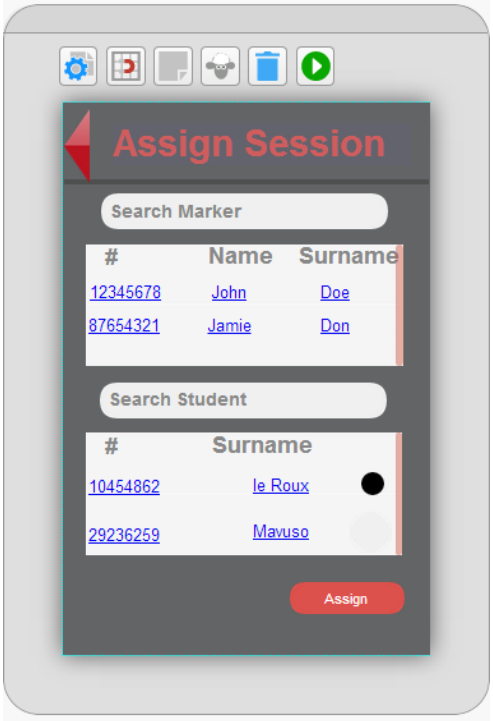
Lecturer Session:



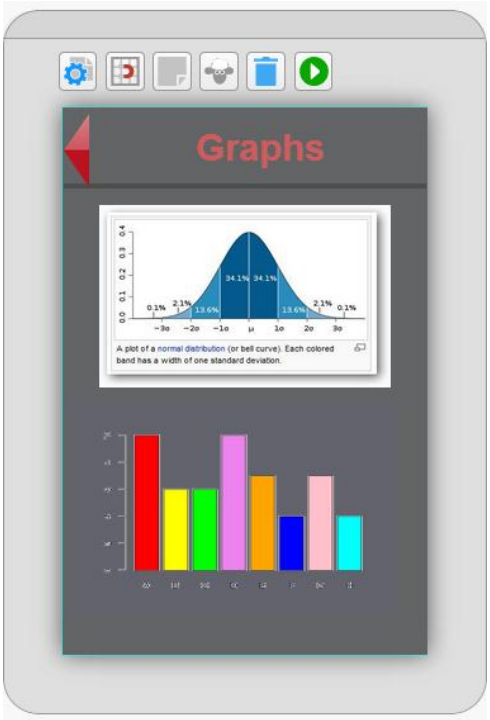
Lecturer Create:



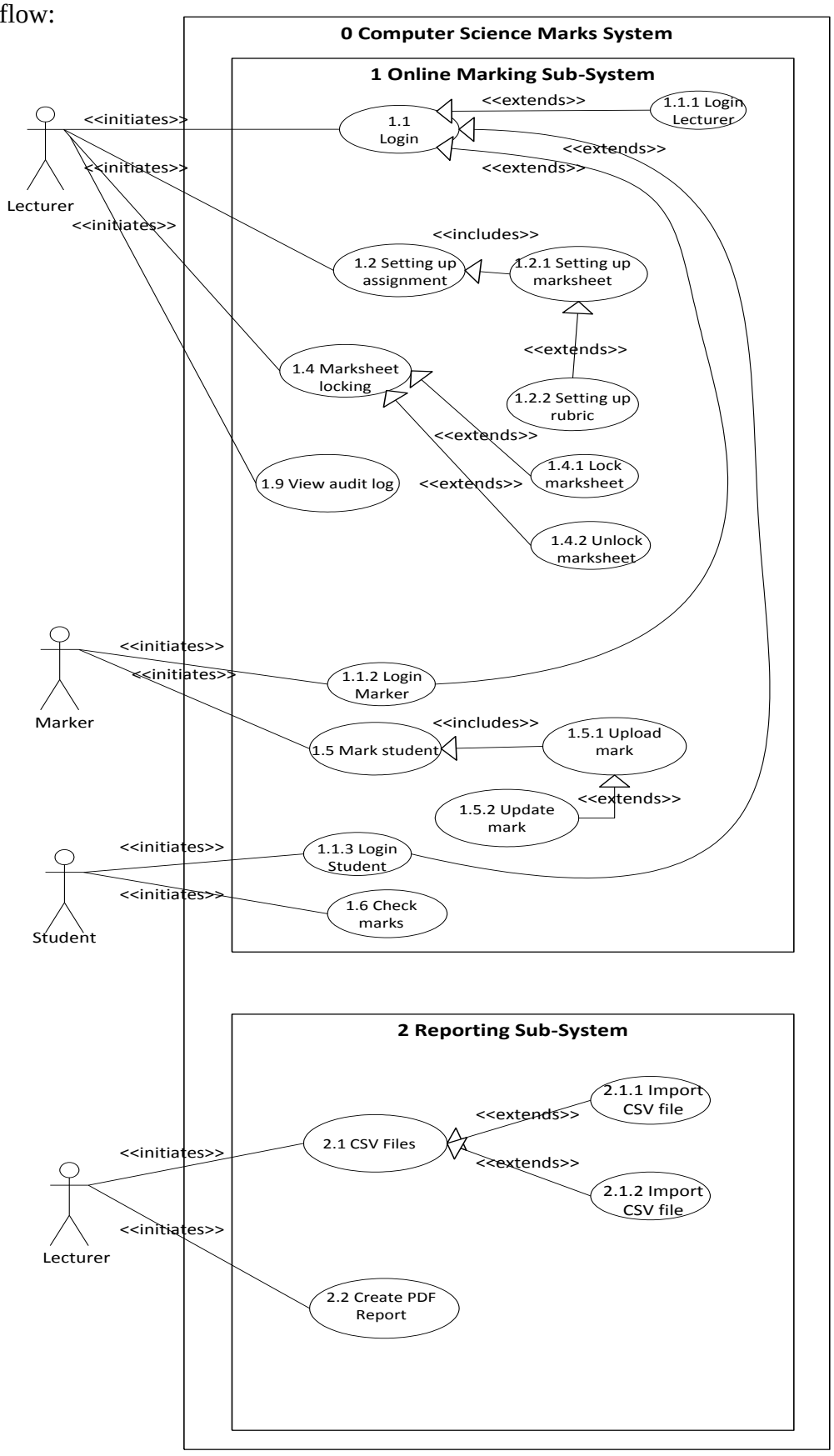
Lecturer Assign:



Lecturer Graphs:



Workflow:


0 Computer Science Marks System
1 Online Marking Sub-System
Lecturer
<<initiates>>
1.1 Login
<<extends>>
1.1.1 Login Lecturer
<<initiates>>
<<extends>>
<<initiates>>
<<includes>>
1.2 Setting up assignment
1.2.1 Setting up marksheet
<<extends>>
1.4 Marksheet locking
1.2.2 Setting up rubric
<<extends>>
1.9 View audit log
<<extends>>
1.4.1 Lock marksheet
1.4.2 Unlock marksheet
Marker
<<initiates>>
<<initiates>>
1.1.2 Login Marker
<<includes>>
1.5 Mark student
1.5.1 Upload mark
<<extends>>
1.5.2 Update mark
Student
<<initiates>>
1.1.3 Login Student
<<initiates>>
1.6 Check marks
2 Reporting Sub-System
Lecturer
<<initiates>>
2.1 CSV Files
<<extends>>
2.1.1 Import CSV file
<<extends>>
2.1.2 Import CSV file
<<initiates>>
2.2 Create PDF Report