

# **Software Architecture Specification**

## **Group 4**

### **Group Members:**

Nico Taljaard (10153285)  
Neels van Rooyen (29052735)  
Stephan Viljoen (11008408)  
Martin Scoeman (10651994)  
Jacques Lewis (28183488)  
Uteshlen Nadesan (28163304)  
Moeletji Semenya (12349136)

**Version 2.0**

03/03/2013	Version 1.0	Document Created	Nico Taljaard
12/03/2013	Version 1.0	API UML Diagrams	Neels van Rooyen
12/03/2013	Version 1.0	System Class Diagrams	Nico Taljaard
12/03/2013	Version 1.0	Framework & Technologies	Stephan Viljoen
13/03/2013	Version 1.0	Database ERD & Description	Uteshlen Nadesan
13/03/2013	Version 1.0	Protocols & Libraries	Martin Scoeman
13/03/2013	Version 1.0	Sequence Diagrams	Jacques Lewis
13/03/2013	Version 2.0	Added Diagrams & Edited layout	Nico Taljaard
13/03/2013	Version 2.0	UI Screen Designs & User Work-Flow Specification	Moeletji Semenya
13/03/2013	Version 2.0	Finalized layout & Structure	Nico Taljaard

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Related Documents . . . . .	4
<b>2</b>	<b>System Description</b>	<b>4</b>
2.1	Technologies . . . . .	4
2.2	Frameworks . . . . .	6
2.3	Protocols . . . . .	6
2.4	Libraries . . . . .	8
<b>3</b>	<b>Details of System</b>	<b>9</b>
3.1	Class Diagram . . . . .	9
3.2	Database Design . . . . .	10
3.3	Logic View . . . . .	12
3.4	Sequence Diagram . . . . .	14
<b>4</b>	<b>UI Screen Designs and User Work-Flow Specification</b>	<b>18</b>
4.1	Log In . . . . .	18
4.1.1	Mobile Interface Screen Designs . . . . .	19
4.1.2	Web Interface Screen Designs . . . . .	21
4.2	Assessment Creation . . . . .	23
4.2.1	Web Interface Screen Designs . . . . .	24
4.3	Marks Management . . . . .	27
4.3.1	Mobile Interface Screen Designs . . . . .	28
4.3.2	Web Interface Screen Designs . . . . .	31
4.4	Reporting . . . . .	34
4.4.1	Mobile Interface Screen Designs . . . . .	35
4.4.2	Web Interface Screen Designs . . . . .	39

Github

# **1 Introduction**

## **1.1 Purpose**

The purpose of this document is to declare and illustrate all lower levels of the system giving a more in-depth view of how the system should be implemented. This is done by designing UMLs, class diagrams and ERDs to depicted how each subsystem should be create as well as how the interconnection between components are to be create and utilized.

## **1.2 Related Documents**

Master Requirement Specification: Provided by the University of Pretoria

# **2 System Description**

These software architecture design decisions are based on the constraints defined by the master requirement specification document:

## **2.1 Technologies**

### **Programming Languages**

- Python
  - A high-level object oriented programming language that can be used for scripting,prototyping, testing and very easy to learn and implement.

- HTML
  - The system must be accessible to users and HTML Markup language will be used to for creating structured displayable content in a web browsers (Mozilla Firefox, Google Chrome, Apple Safari and Microsoft Internet Explorer). Reference 4.1.1.
- CSS
  - Used for formatting of HTML documents creating rich web interfaces.
- JavaScript
  - Dynamic programming language used for client side interaction.
- SQL
  - Used for data managing (insert, delete, update and read) of your relational data using the Object-Relational Mapper.
- Java
  - A Powerful language with vast libraries with which the android interface will be created. Reference 4.1.2

### **Application Server**

- A Django application server will be used to run and deploy the system within the cs.up.ac.za Apache web server. The webserver must be published as SOAP-based.

### **Database Technology**

- MySQL implement the quality requirements stated by the master specification and it decouples the system from the database.
- Scalability and Flexibility:  
Provides scalability, sporting the capacity to handle deeply embedded applications  
Platform flexibility (Linux, UNIX, Mac and Windows)

- Security  
Offering exceptional features that ensure absolute data protection when looking at database authentication, backup and recovery.

## 2.2 Frameworks

### Object-Relational Mapper

- A technique for converting data between incompatible types in object-oriented languages. With our application data to outlive the applications process. The persistence to the relational database must be done using the Object-Relational Mapper bundled with Django. Django is the chosen Object-Relational mapper used to obtain, retrieve and transfer data between a variety of platforms such as from the MySQL database to the web client and android interface. You can define your data model in Python and access through the database API.

### Web Service Frame Work

- The Django framework is based on the model view controller, which is used for creating complex database driven websites with the ability of reusability of components and rapid development. The key quality requirements that Django accommodate are scalability, security, reliability and integration at a lesser extent.

## 2.3 Protocols

The protocols we will be using are HTTPS for security, LDAP for the computer science web site, XML and JSON.

The two main protocols for data transfer in a web application environment are JSON and XML. We will be using JSON as our main data transfer protocol, but also have support for XML.

The reasons will be explained in accordance with the quality requirements.

**Security:**

JSON is limited to storing only classical data like numbers and text, but with XML you can store any data type you want. That could be a security risk, because you can store an executable as well.

**Auditability:**

The simplicity of JSON and the fact that it's only text based, makes it very easy to audit.

**Testability:**

JSON and XML are easy to test, because both of their implementation is self describing and is cross platform independent.

**Usability:**

Both XML and JSON is human readable, but JSON is more so. The reason is the JSON files are more restrictive and a lot less data formats that it supports. JSON uses an array to store data, where XML uses a tree. Seeing as arrays are more a structure of object oriented languages than trees are, it will be much easier to parse the data in the array than in the tree.

**Scalability:**

Using a JSON document, without a schema makes it very easy to scale the JSON document. JSON also has no max size or max length of object. So you can send and receive a document without being depended on size. The limitations of the size will be determined by the server or browser.

**Performance:**

The performance of JSON is very high, because of the fact it is only text based. So there will be no support for large files like images, videos and so forth.

**Conclusion:**

JSON is selected as our main protocol, because it only supports text and no other formats. Seeing how our system will only be transferring text, JSON is the best option. Also JSON is simpler to use, easier to read and many people are changing from XML to JSON, because it is the newer technology.

## 2.4 Libraries

The libraries we will be using, will mostly if not all consist of open source libraries.

**PDF generator:**

The system must be able to generate PDFs on two different platforms, android and with a web service. For android we will be using the iText library that supports creating a PDF on android. For our server side, we will be using a PHP library called TCPDF, which will allow us to generate a PDF server side.

**JSON marshalling/de-marshalling:**

For our JSON marshalling/de-marshalling, we will be using the library EclipseLink MOXy. It works with XML, but also works great with JSON. There is a lot of support with this library, if one needed help.

**LDAP integration:**

We will be using django-auth-ldap 1.1.4 for the Django authentication backend that will authenticate against an LDAP service. For python we will be using the python-ldap library to allow us to use python to communicate with LDAP.



## 3 Details of System

### 3.1 Class Diagram

These diagrams depicts the classes that will be used with all the methods with parameters and return types and attributes to be used.

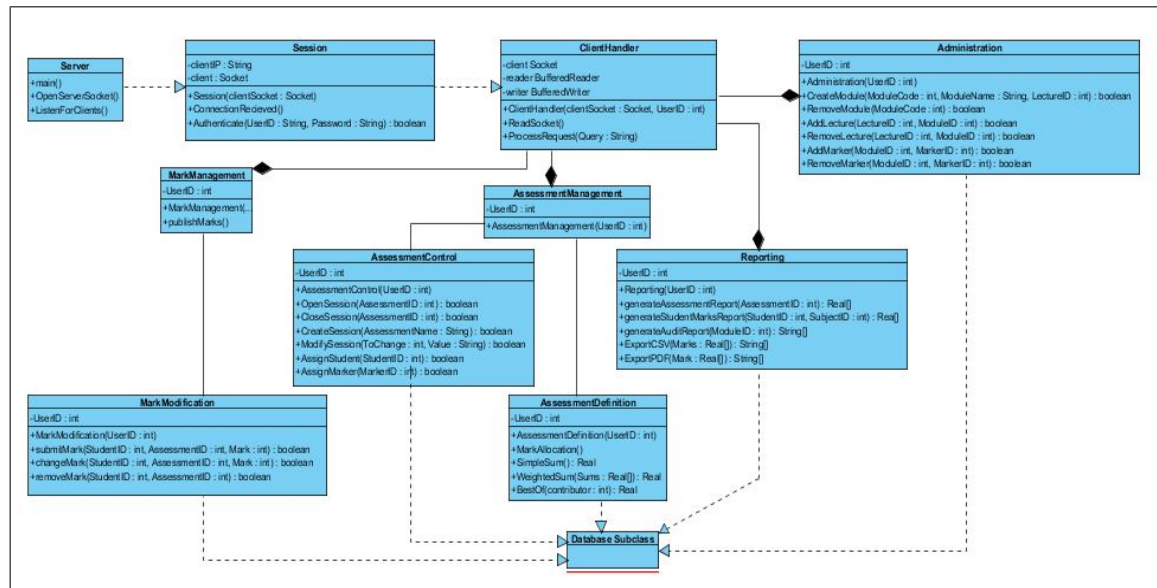


Figure 1: System Class Diagram

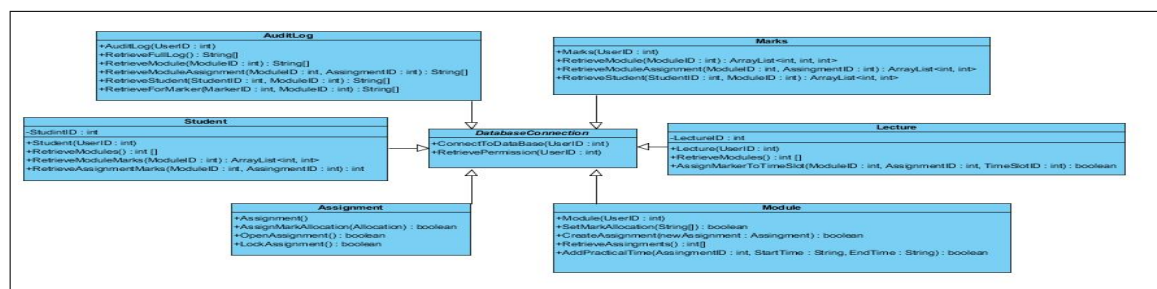


Figure 2: Database Class Diagram

## 3.2 Database Design

The Database software that is to be used is MySQL. The design above will be implemented in MySQL. The population of the database will be taken from the LDAP database of the Department of Computer Science. These will include, but not limited to, student numbers, names, surnames, courses, lecturers' details, etc.

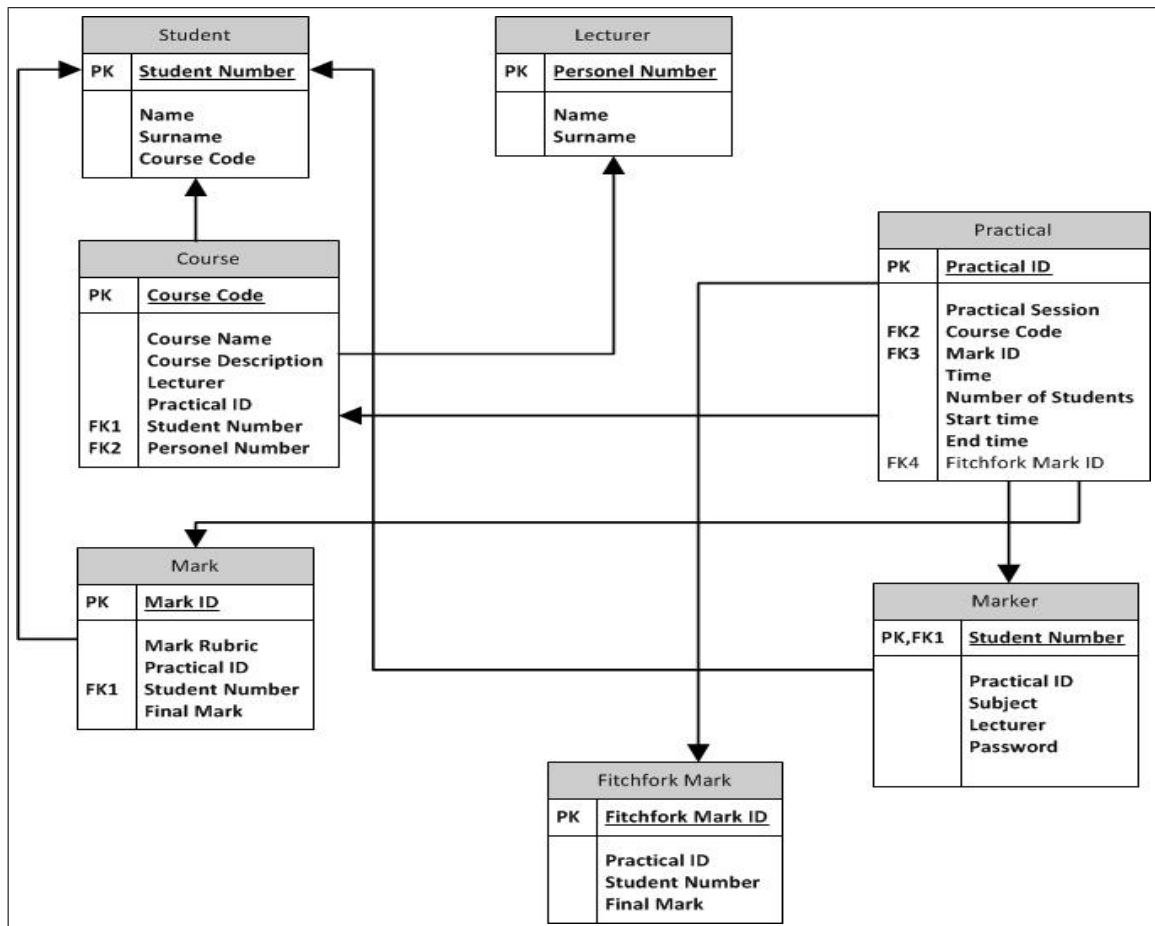


Figure 3: Database ERD

- Student
  - All student details will be imported and added to complete the student table of the database. All students have a unique student number that identifies them, names, surnames and courses can then be added. Adding these details will enable access to all additional tables.
- Practical
  - The Practical table has the session for that specific practical the number of students assigned to that practical, the Fitchfork marks that can be added although it is not mandatory. The start and end times are also added as they are essential for the application to know when to open and close editing of the database. The practicals are indirectly linked to the lecturers who will have access to their subject's practicals.
- Mark
  - A table of marks will capture the marks for each student along with the final mark each student will receive for the practical. Marks are linked to the student and the practical itself via the student number and practical identification number. This will simplify access, not only for the students, but also for the markers and lecturers combined.
- Marker
  - Markers are in a separate database and include a password specific to that marker for accessing the android application and include the overseeing lecturer and subject and practical the marker is assigned to.
- Fitchfork
  - Marks that need to be imported from Fitchfork will be done separately and then added to the above database tables to complete the database entries for the practical.

- Lecturer
  - All courses that require marks to be entered into this database will have a lecturer assigned to the course and hence, the practical.

### 3.3 Logic View

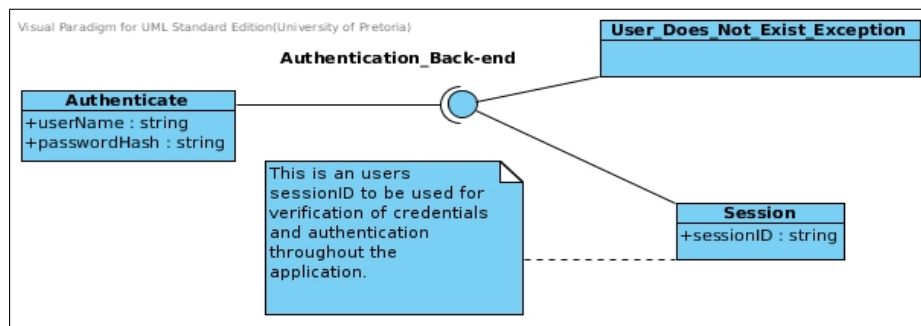


Figure 4: Authentication API

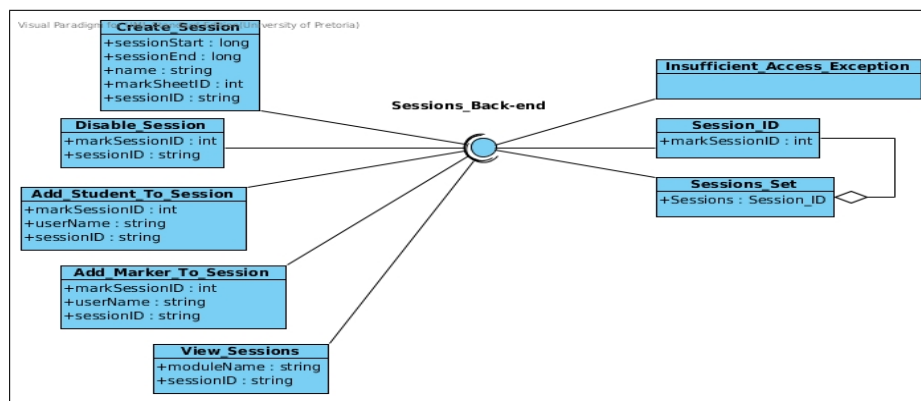


Figure 5: Sessions API

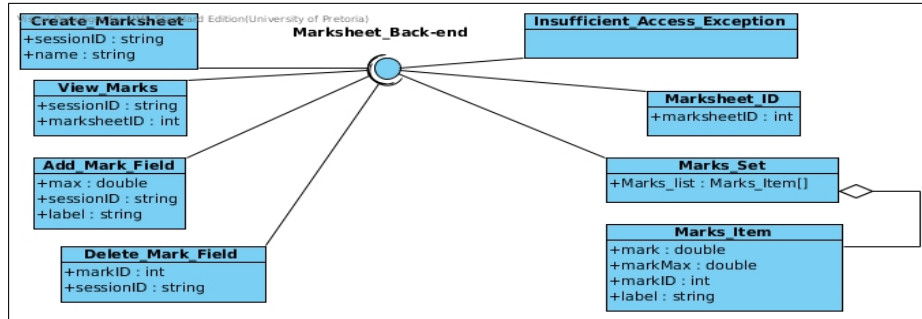


Figure 6: Marksheet API

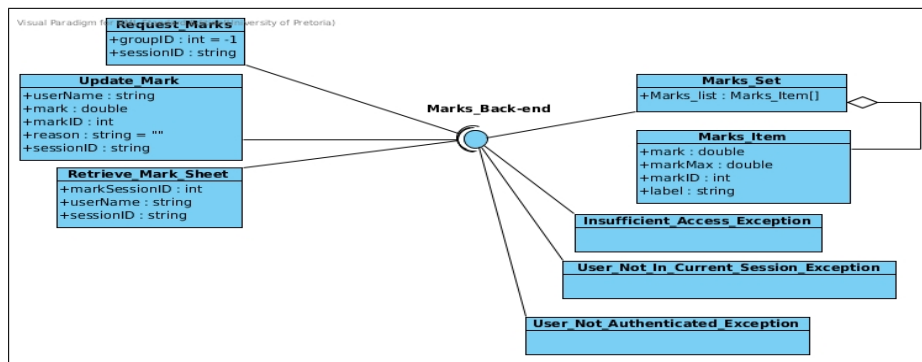


Figure 7: Marks API

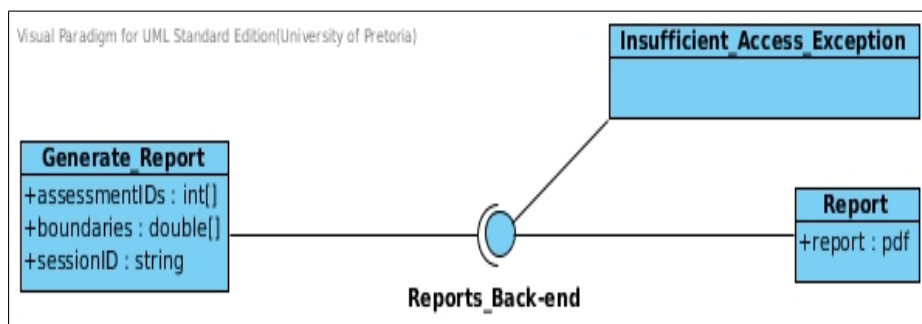


Figure 8: Reports API

### 3.4 Sequence Diagram

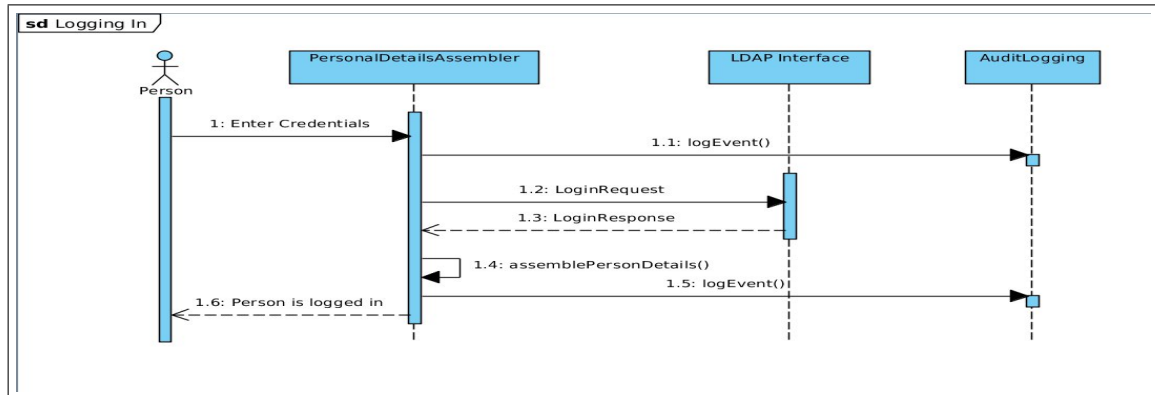


Figure 9: Login

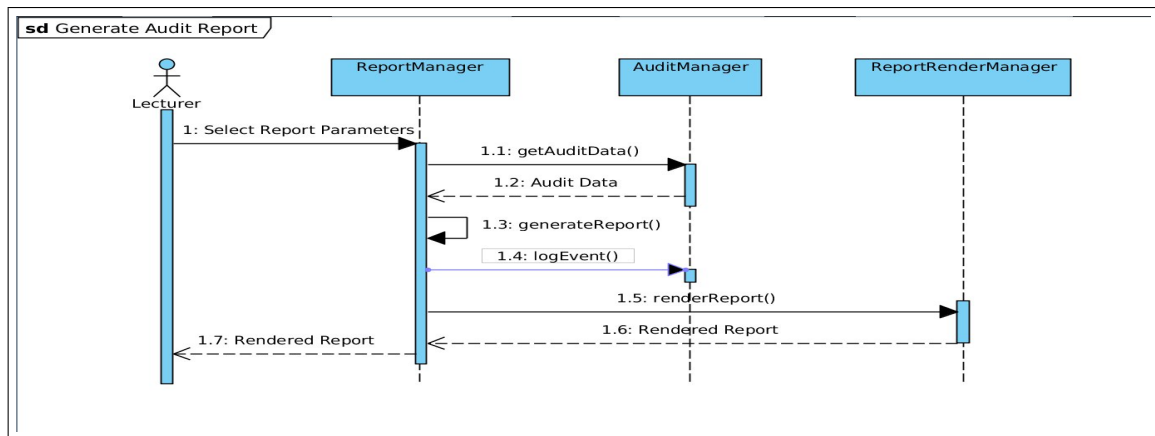


Figure 10: Generate Audit Report

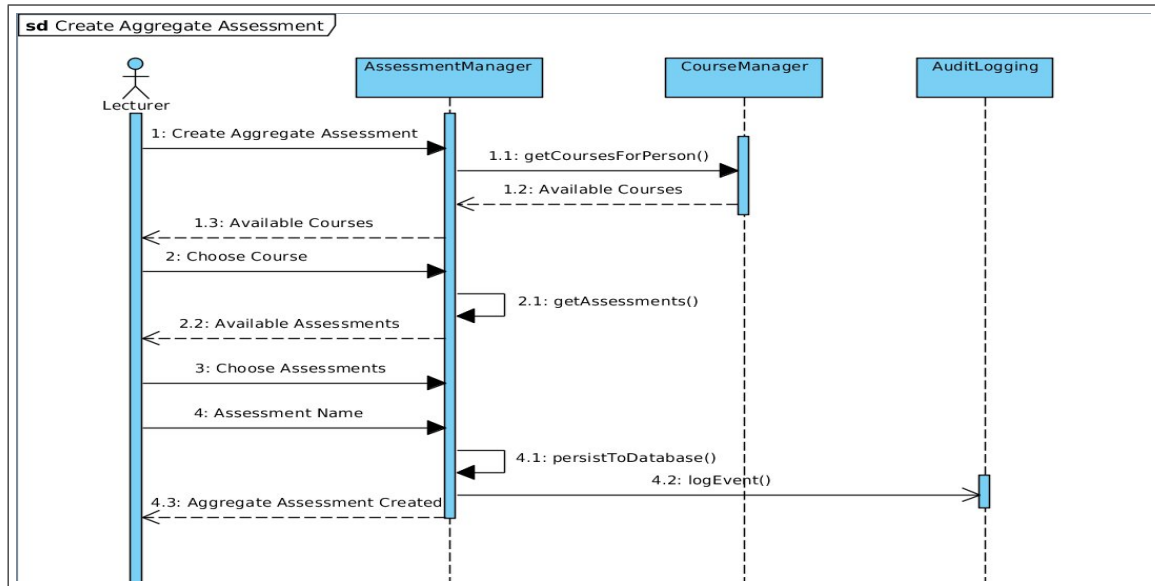


Figure 11: Create Aggregate Assessment

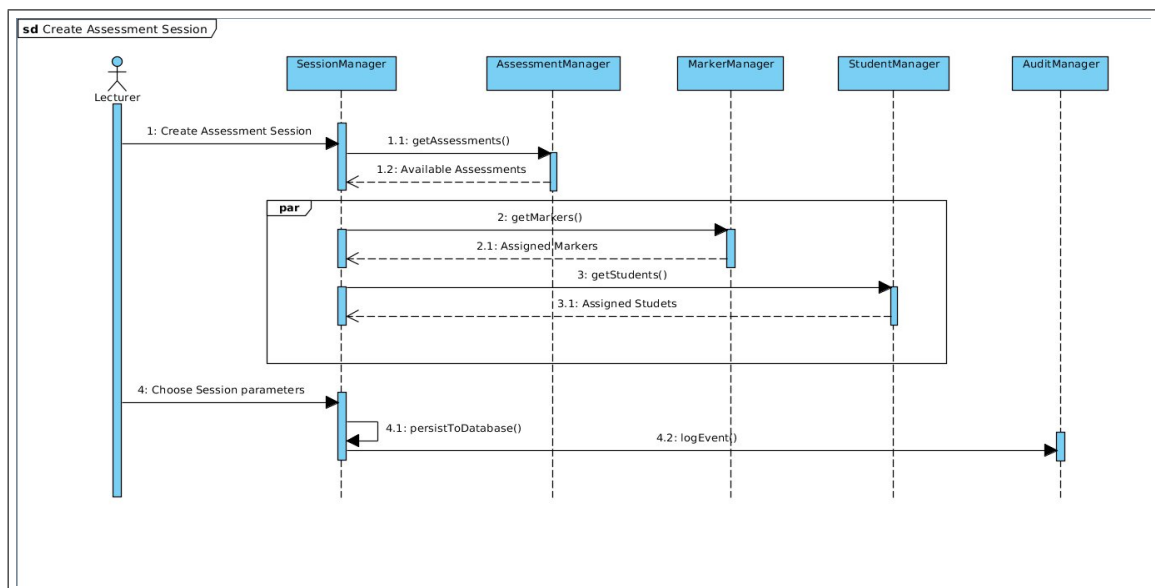


Figure 12: Create Assessment Session

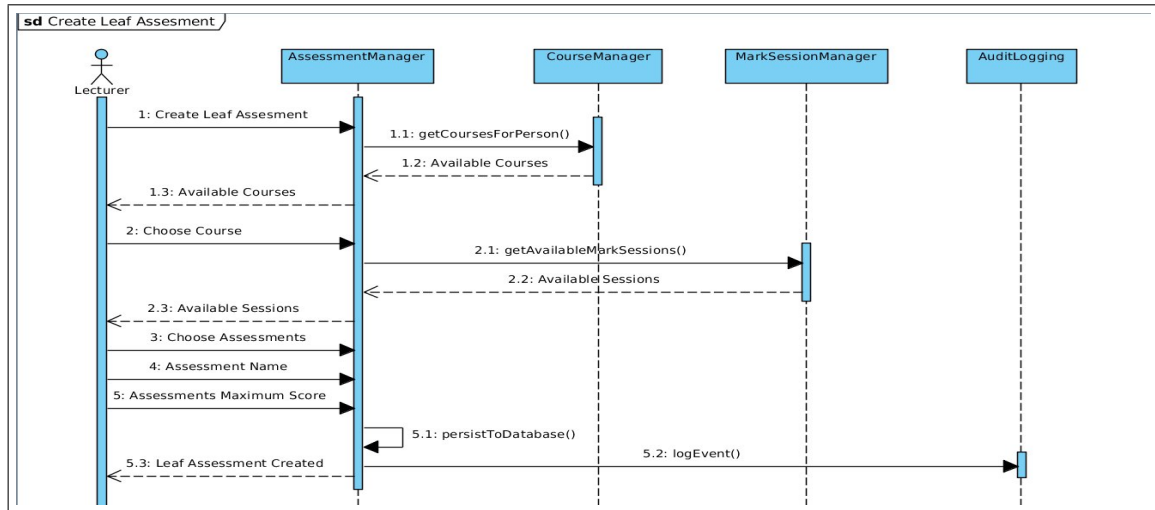


Figure 13: Create Leaf Assessment

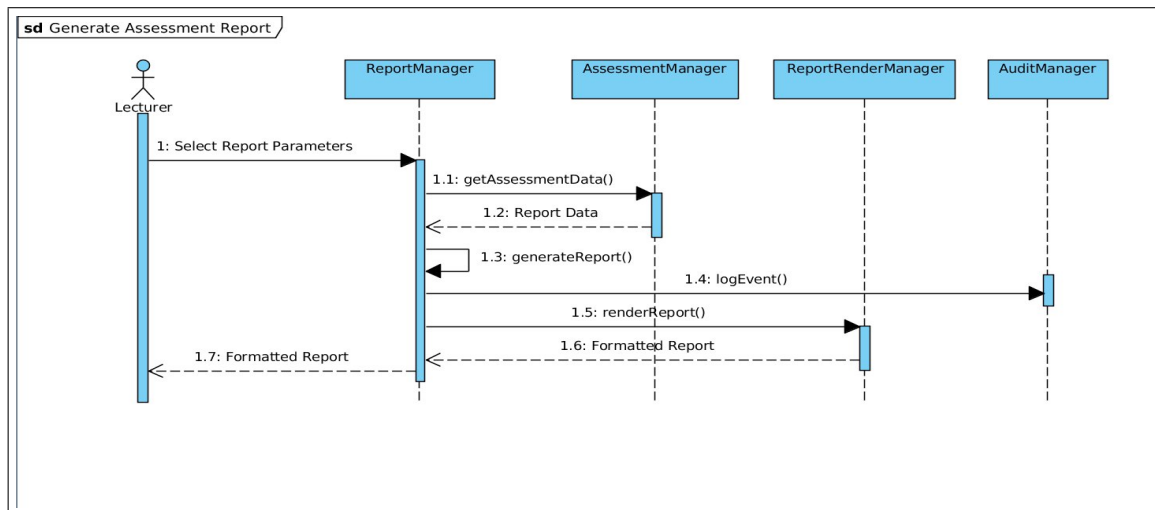


Figure 14: Generate Assessment Report



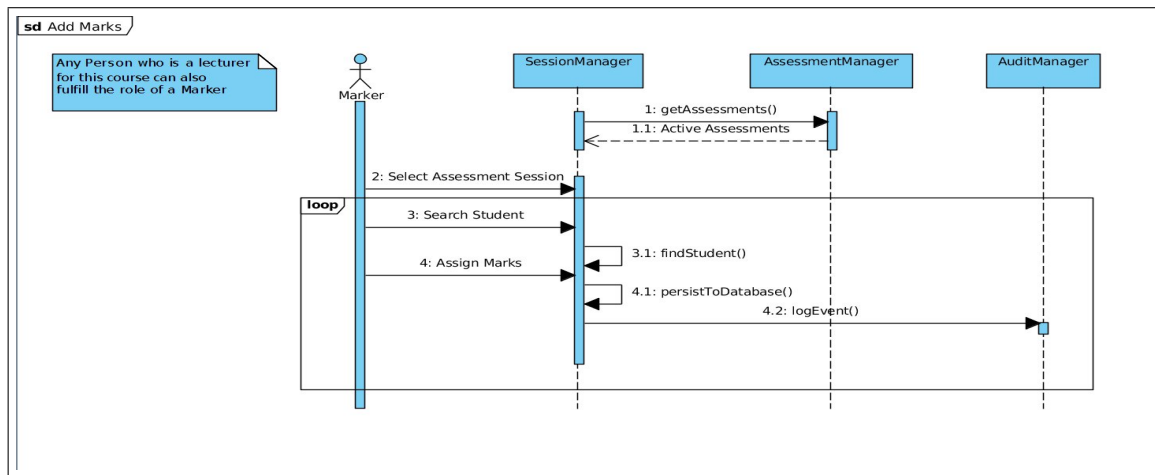


Figure 15: Add Marks

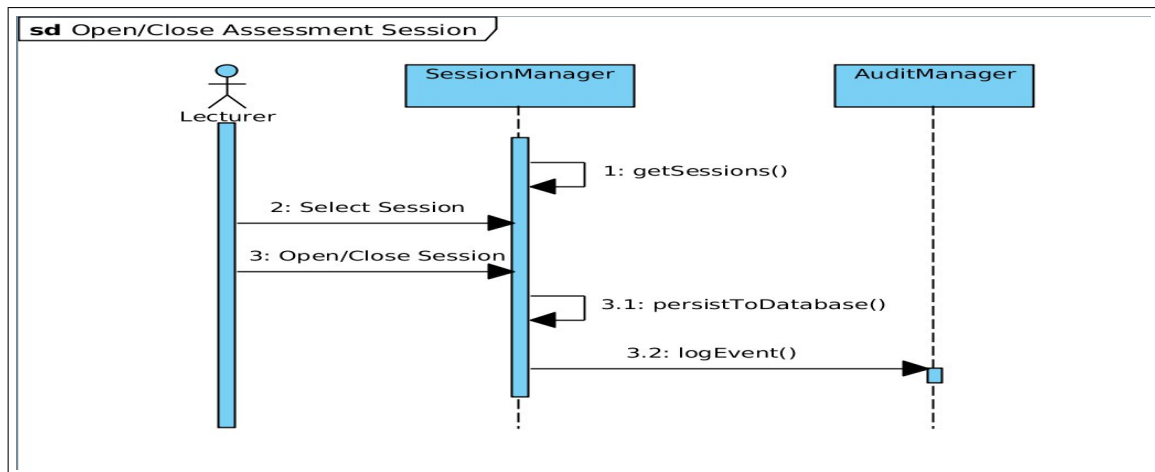


Figure 16: Open Close Session

## 4 UI Screen Designs and User Work-Flow Specification

This section will look at how the system will be used by the different users. Each functional requirement will be accompanied by the appropriate screen designs(The web and mobile user interfaces).

### 4.1 Log In

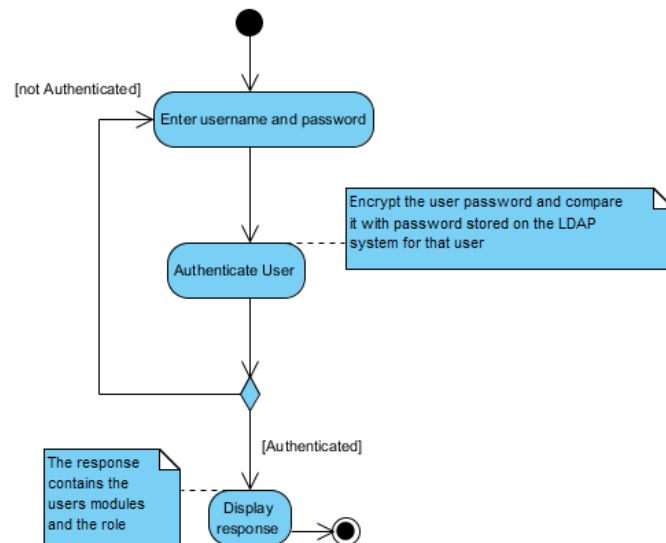


Figure 17: User log in activity diagram.

The activity diagram above is a abstract depiction of what will happen when a user logs into the system. After the user gets authenticated by the system, the authentication response object will contain the users information(the modules and roles) as shown in Figure 3 and 5.

#### 4.1.1 Mobile Interface Screen Designs



The image shows a mobile login screen for the 'SQUIRREL Marking System'. The screen has a dark grey background. At the top, the word 'SQUIRREL' is written in a large, white, serif font, with 'Marking System' in a smaller, white, sans-serif font below it. Below the title, there are two white input fields. The first is labeled 'Username' in a small, grey, sans-serif font. The second is labeled 'Password' in a small, grey, sans-serif font. Below the password field, there is a small, dark grey square checkbox followed by the text 'Remember Me' in a small, grey, sans-serif font. At the bottom center, there is a white rectangular button with the text 'Log In' in a small, grey, sans-serif font.

Figure 18: The mobile log in screen design.

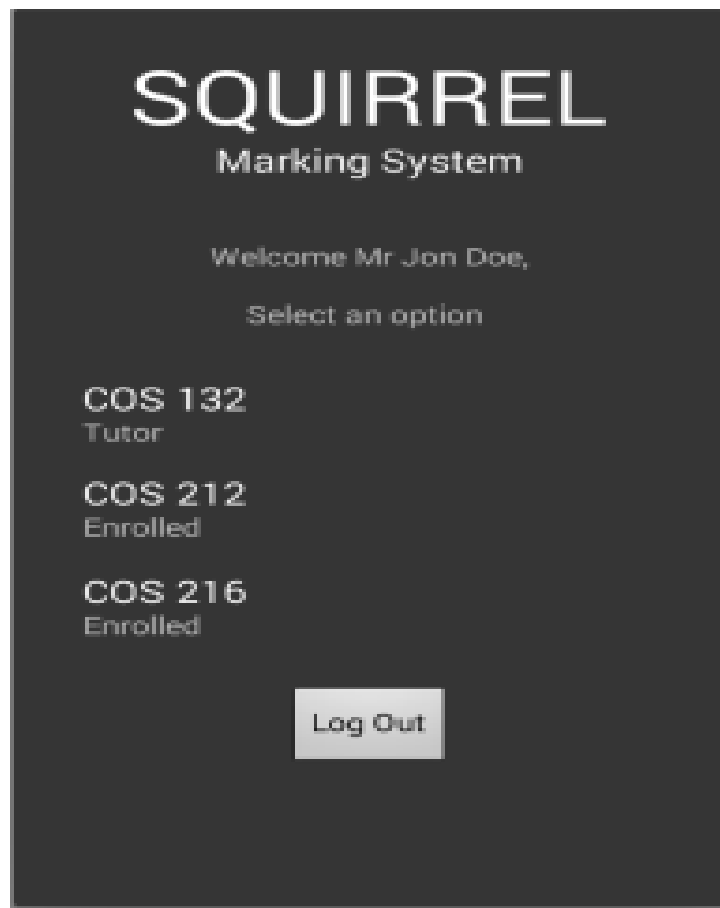
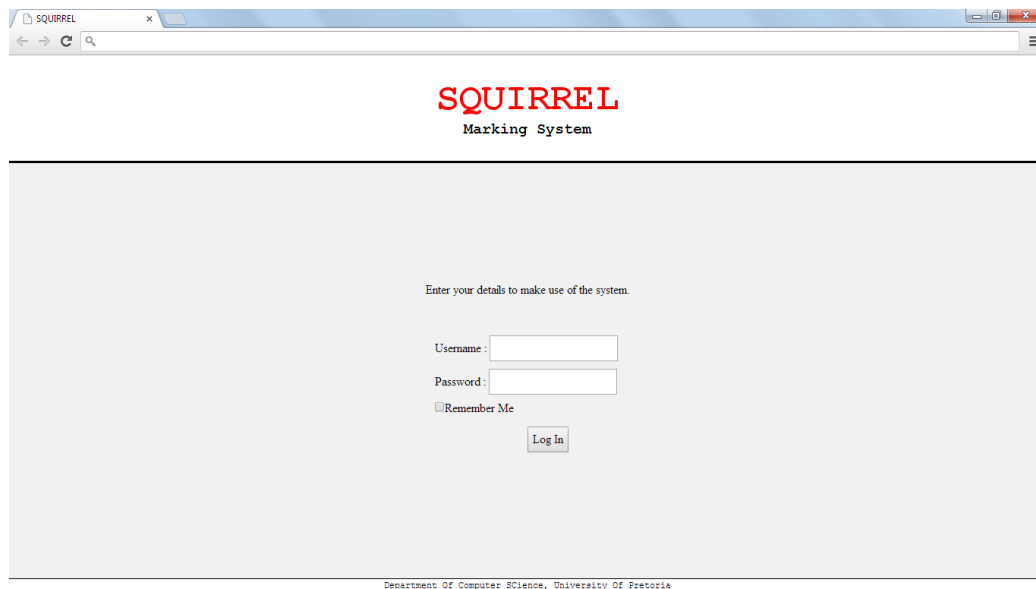


Figure 19: Screen design of a successfull log in request.

Figure 3 shows the modules the user is associated with and the roles users have for particular module.

### 4.1.2 Web Interface Screen Designs



The screenshot displays a web browser window with a single tab titled "SQUIRREL". The address bar is empty. The main content area features the "SQUIRREL" logo in red, with "Marking System" in black text below it. A large, light gray rectangular box contains the login instructions: "Enter your details to make use of the system." Below this, there are two input fields labeled "Username:" and "Password:". A checkbox labeled "Remember Me" is positioned below the password field. A "Log In" button is located at the bottom of the form. The footer of the page, visible below the gray box, reads "Department Of Computer Science, University Of Pretoria".

Figure 20: Screen design for web interface

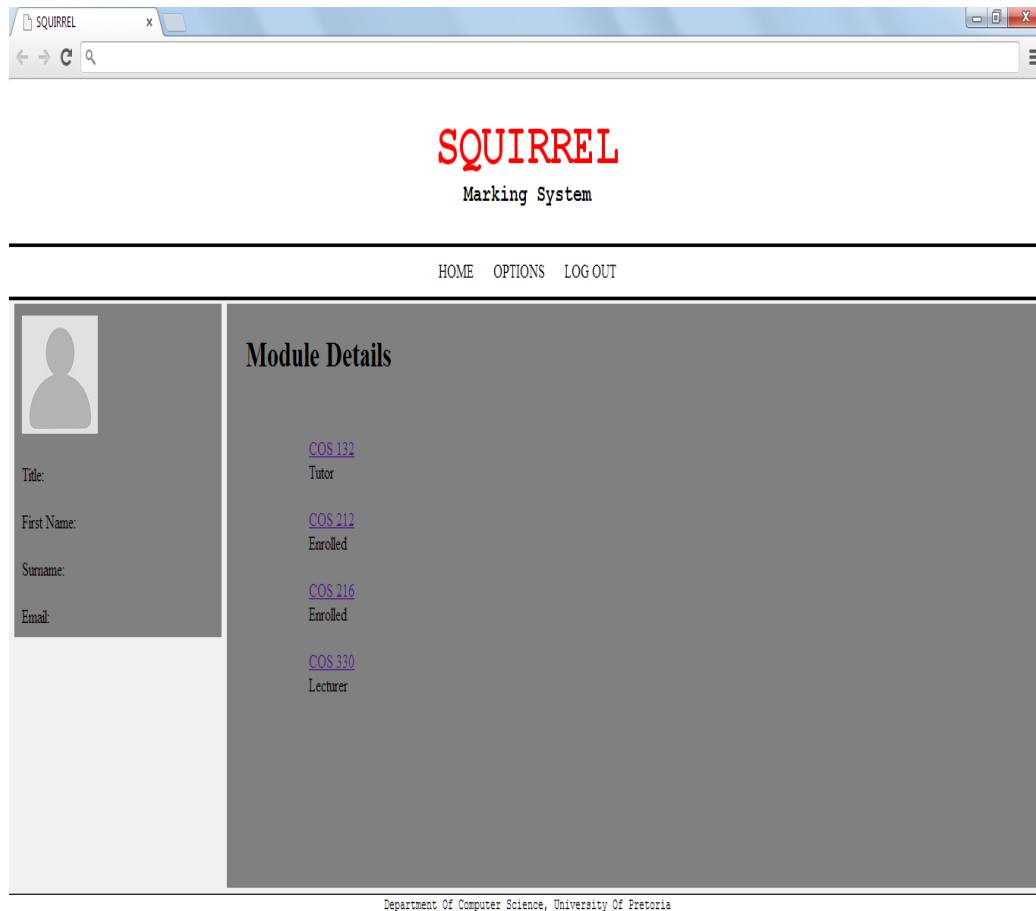


Figure 21: This is the welcome page the user sees after successfully being authenticated.

## 4.2 Assessment Creation

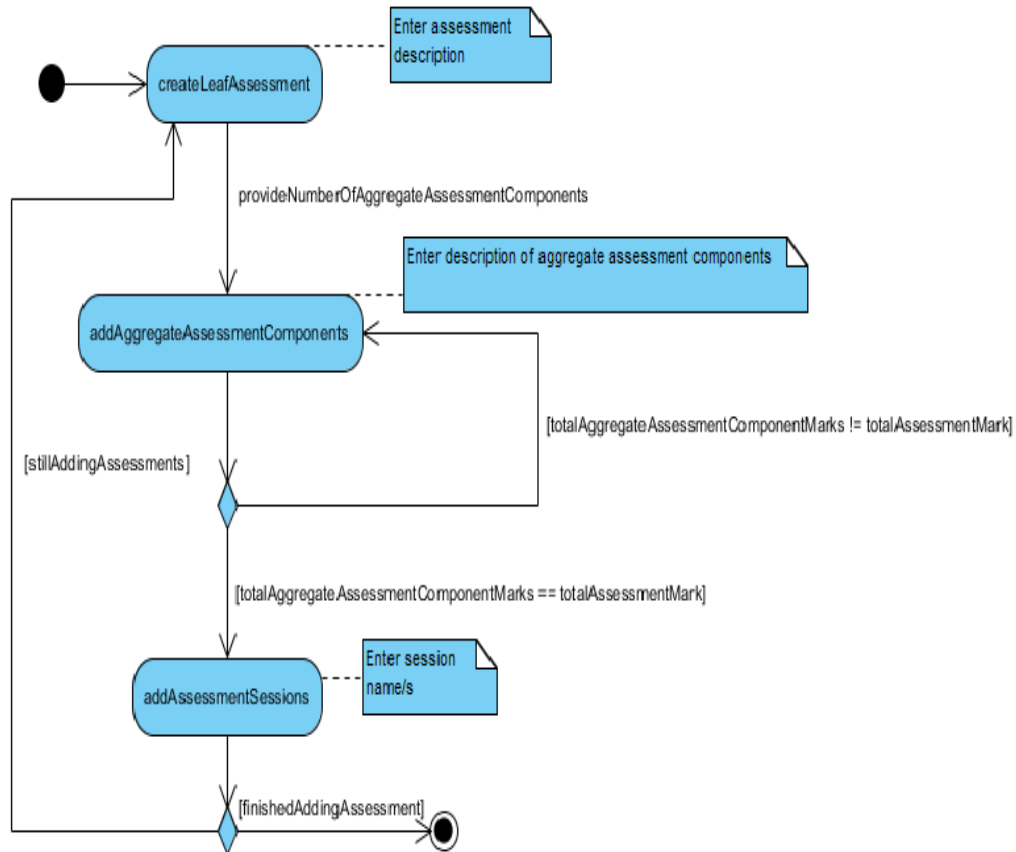


Figure 22: Activity diagram for creating atomic leaf assessments, aggregate assessments as well as sessions for the atomic assessments.

The assessment creation operation is only available to lecturers and only has the web interface as shown below.

### 4.2.1 Web Interface Screen Designs

The screenshot shows a web browser window with a single tab titled 'SQUIRREL'. The address bar is empty. The main content area features the 'SQUIRREL' logo in red, with 'Marking System' in black below it. A navigation bar contains links for 'HOME', 'OPTIONS', and 'LOG OUT'. The interface is divided into a left sidebar and a main content area. The sidebar includes a user profile icon and labels for 'Title:', 'First Name:', 'Surname:', and 'Email:'. The main content area is titled 'Assessment View:' and contains several input fields: 'Assessment Name:' (text box), 'Total Marks' (text box), 'Select a module' (dropdown menu), 'Number Of Assessment Components:' (dropdown menu with '0' selected), and 'Number Of Assessment Sessions:' (dropdown menu with '0' selected). A 'Next' button is located at the bottom of the form. The footer of the page reads 'Department Of Computer Science, University Of Pretoria'.

Figure 23: Screen design for creating an atomic assessment.



SQUIRREL

x


← → ↻ 🔍

☰

# SQUIRREL

Marking System

HOME   OPTIONS   LOG OUT



Title:

First Name:

Surname:

Email:

## Add Assessment Components:

Select an assessment ▾

Component Name:  
Mark Allocation ▾

Component Name:  
Mark Allocation ▾

Component Name:  
Mark Allocation ▾

Next

Department Of Computer Science, University Of Pretoria

Figure 24: Screen design for adding aggregate assessments.

The screenshot shows a web browser window with the address bar displaying `file:///C:/xampp/htdocs/Project/Moeletji's%20Files/sessions1.html`. The browser tabs include 'GitHub for Windows', 'Downloads', and 'SQUIRREL'. The page title is 'SQUIRREL Marking System'. A navigation bar at the top contains links for 'HOME', 'OPTIONS', and 'LOG OUT'. The main content area is titled 'Add Assessment Session:' and features a sidebar on the left with a user profile icon and labels for 'Title:', 'First Name:', 'Surname:', and 'Email:'. The main form contains three identical sets of input fields: a dropdown menu labeled 'Select an assessment', a 'Session Name' text box, and an 'Open' checkbox. At the bottom of the form is a 'Create Assessment' button. The footer of the page reads 'Department Of Computer Science, University Of Pretoria'.

SQUIRREL  
Marking System

HOME OPTIONS LOG OUT

**Add Assessment Session:**

Select an assessment ▼

Session Name:

☐ Open

Session Name:

☐ Open

Session Name:

☐ Open

Create Assessment

Department Of Computer Science, University Of Pretoria

Figure 25: Screen design for adding sessions for an atomic assessment.

### 4.3 Marks Management

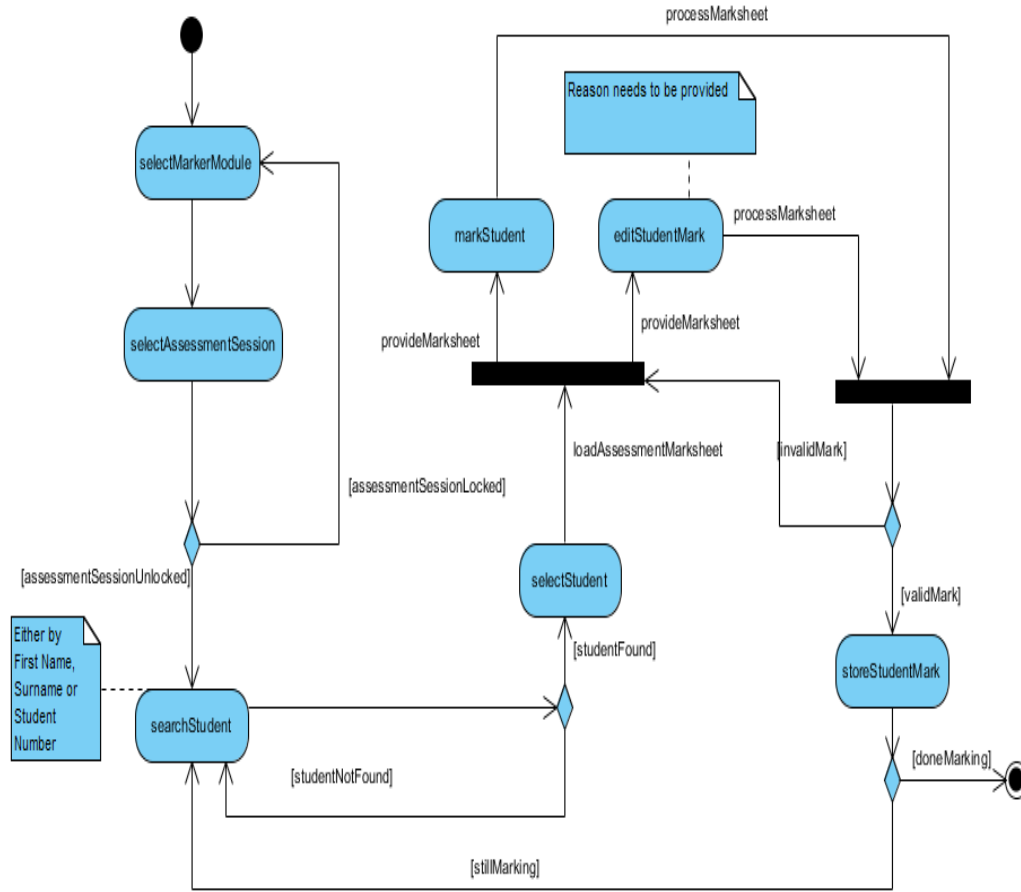


Figure 26: Activity diagram depiction of how a student would get marked.

The activity diagram(Figure 10) depicts how a marker would go about marking a student. The activity diagram is closely linked with the screen designs as it shows how the user would go about marking a student.

#### 4.3.1 Mobile Interface Screen Designs

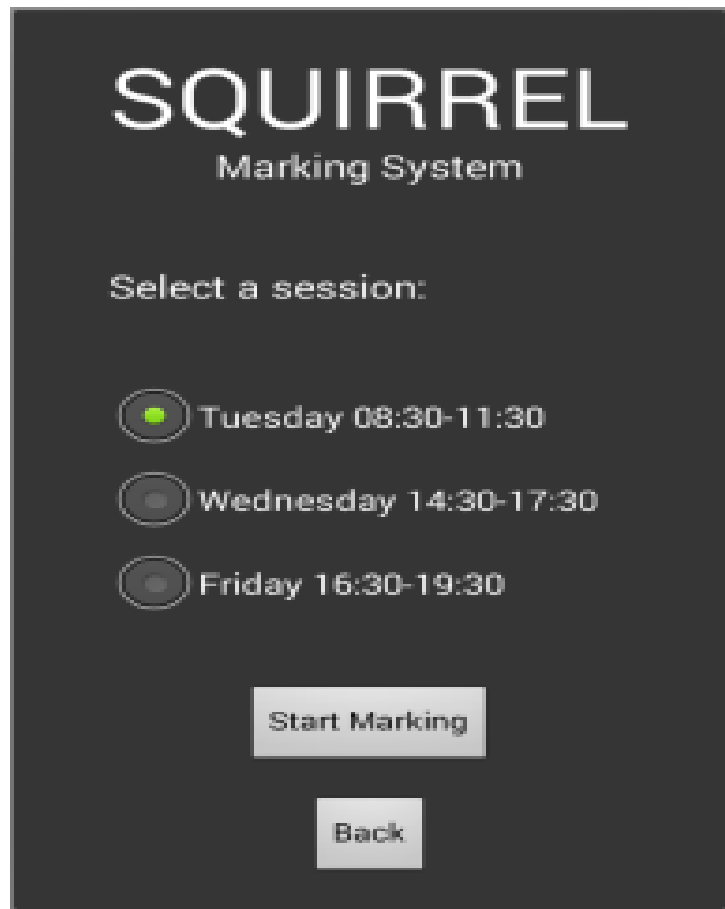


Figure 27: Screen design of assessment sessions.

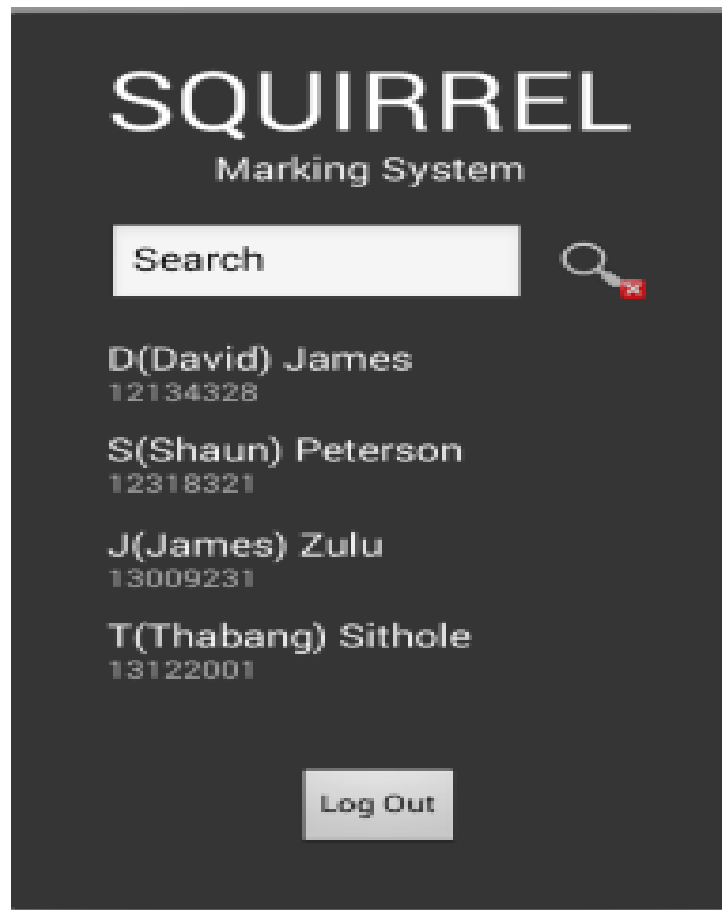


Figure 28: Screen design of how the students would be displayed and search for on the system.

The image shows a web interface for a marking system. At the top, the title 'SQUIRREL' is displayed in a large, white, serif font, with 'Marking System' in a smaller, white, sans-serif font below it. The student's name 'D(David) James' and ID '12134328' are listed in white text. There are two rows of input fields, each preceded by a label in white text: 'Porgram compiles (5)' and 'Correct output generated (5)'. The input fields are empty white rectangles. Below these is a 'Total:' label. At the bottom center is a grey rectangular button with the word 'Submit' in white text.

SQUIRREL  
Marking System

D(David) James  
12134328

Porgram compiles (5)

Correct output generated (5)

Total:

Submit

Figure 29: Screen design of the interface used to enter student marks.

### 4.3.2 Web Interface Screen Designs

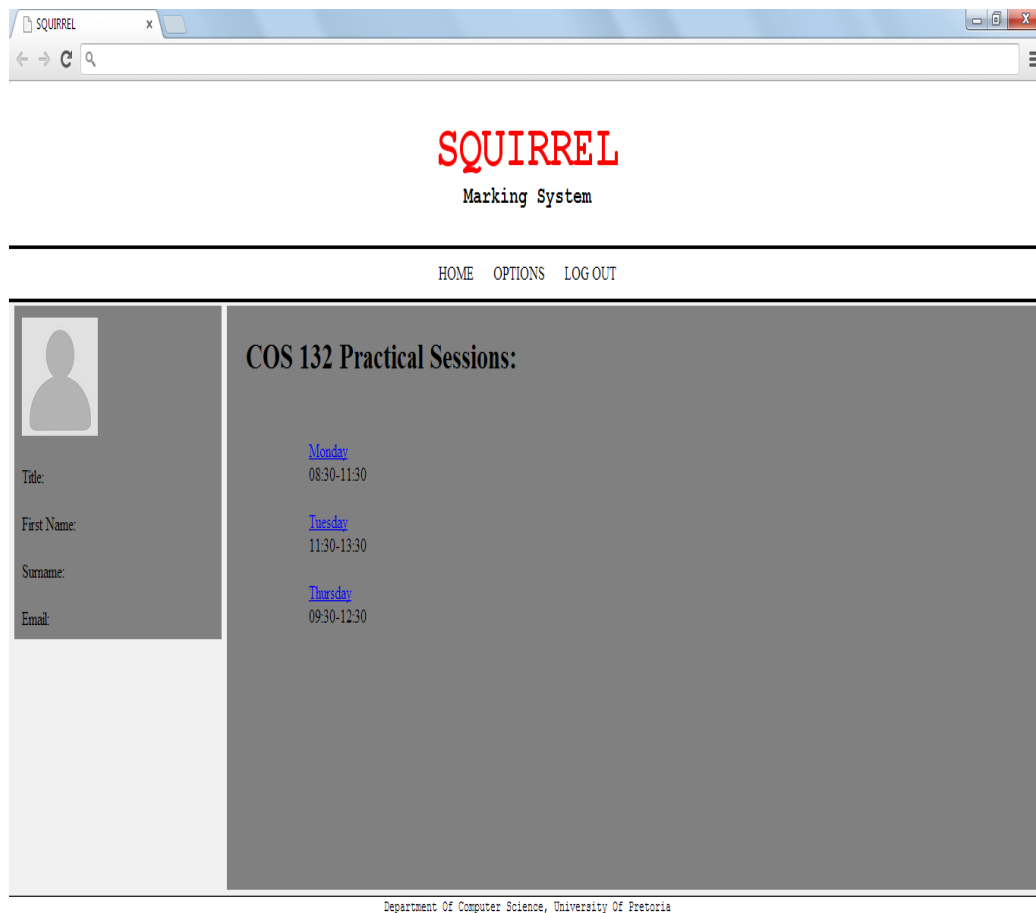


Figure 30: List of available marking sessions.

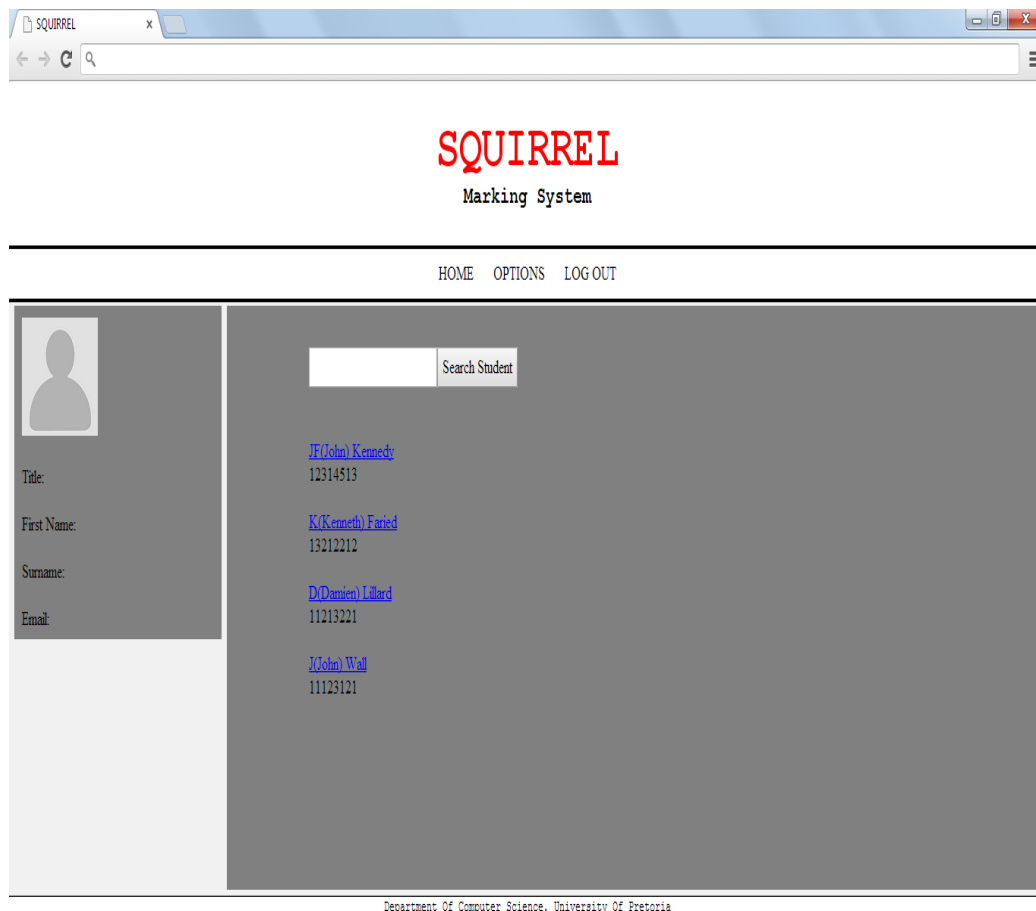


Figure 31: List of students in the sessions who need to be marked.




SQUIRREL

← → ↻ 🔍

# SQUIRREL

Marking System

[HOME](#) [OPTIONS](#) [LOG OUT](#)



Title:

First Name:

Surname:

Email:

**D(Damien) Lillard**  
**11213221**

Program compiles(5)

Correct output generated(5)

TOTAL/10:

Department Of Computer Science, University Of Pretoria

Figure 32: A student being marked.

## 4.4 Reporting

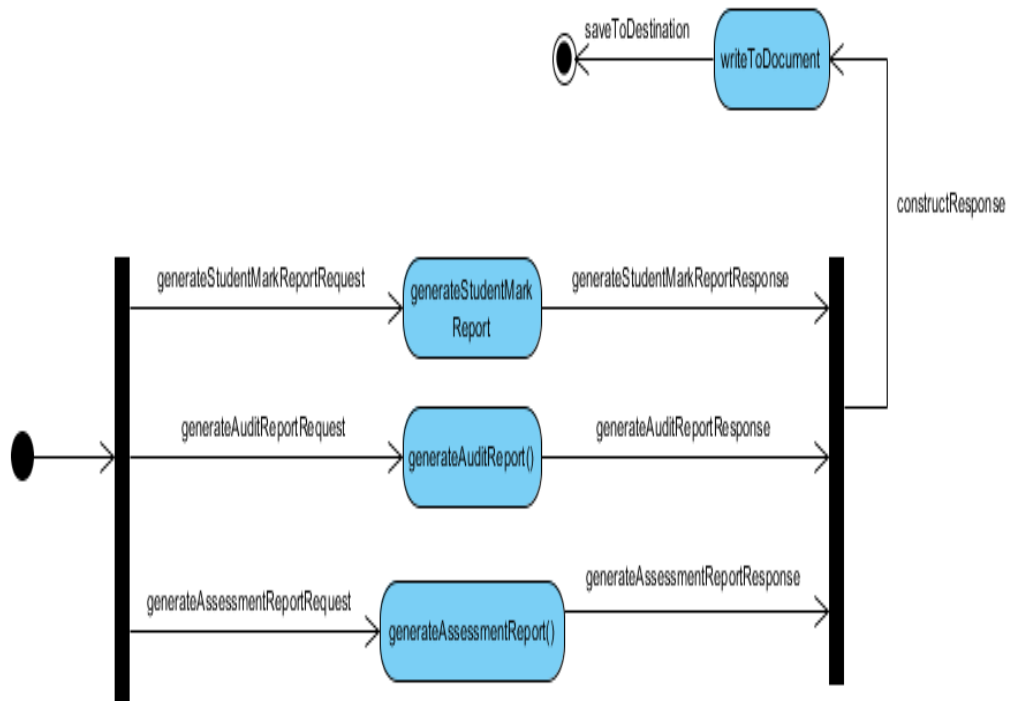


Figure 33: Activity diagram of how reports would be generated by users.

Students will be able to generate their mark reports on either platform (mobile or web interface), while lecturers will only be able to generate assessment and audit reports via the web interface.

#### 4.4.1 Mobile Interface Screen Designs

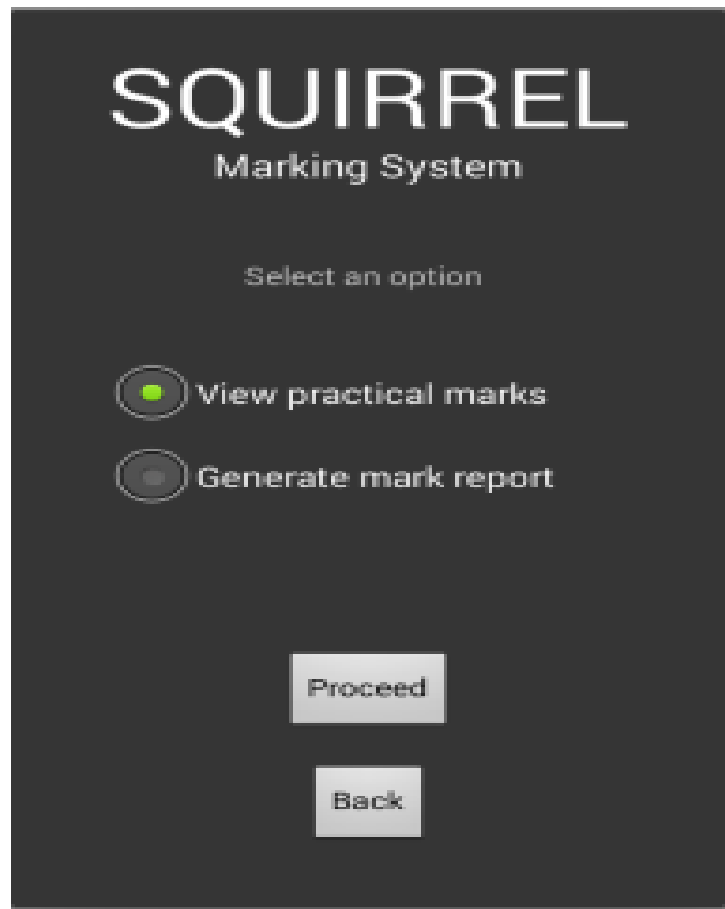


Figure 34: Options of reports a student can generate.

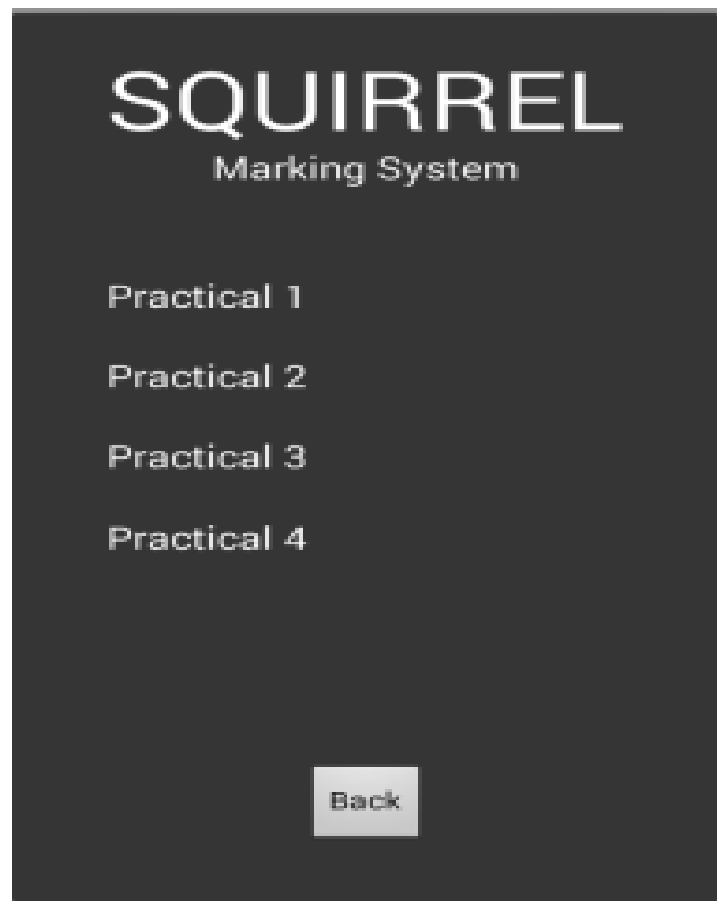


Figure 35: List of practicals.

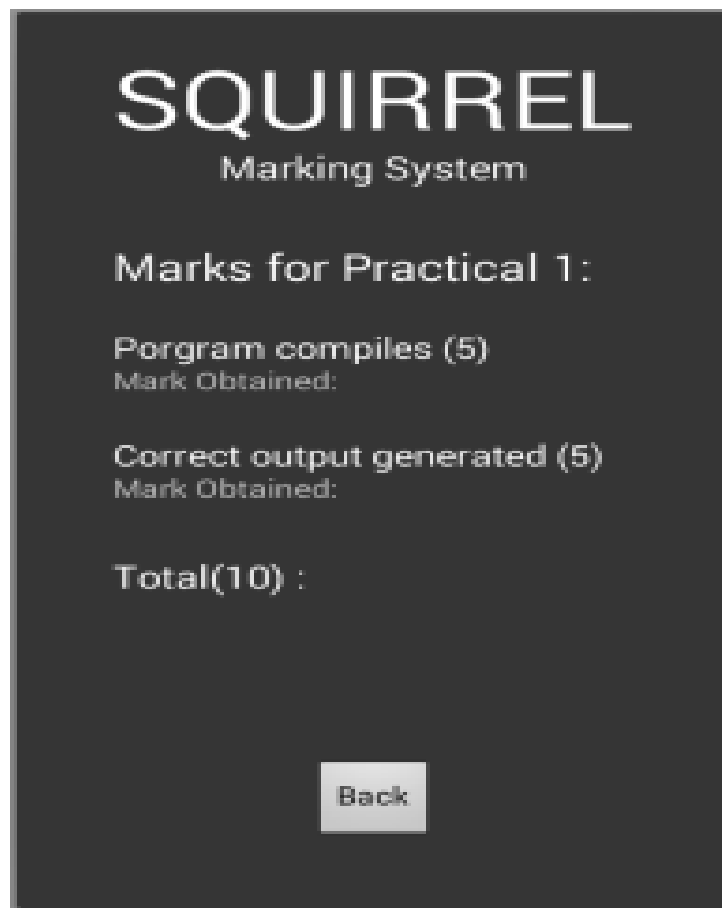


Figure 36: Screen design of a report for a students practical 1.

The image shows a dark-themed user interface for the 'SQUIRREL Marking System'. At the top, the title 'SQUIRREL' is in large white letters, with 'Marking System' below it in smaller white letters. Below the title, there are three items, each consisting of a green checkmark inside a square box followed by text: 'All modules', 'COS 212', and 'COS 216'. At the bottom of the interface, there are two buttons: a larger 'Generate mark report' button and a smaller 'Back' button below it.

SQUIRREL  
Marking System

☒ All modules

☒ COS 212

☒ COS 216

Generate mark report

Back

Figure 37: Screen design for generating a students' mark report.

#### 4.4.2 Web Interface Screen Designs

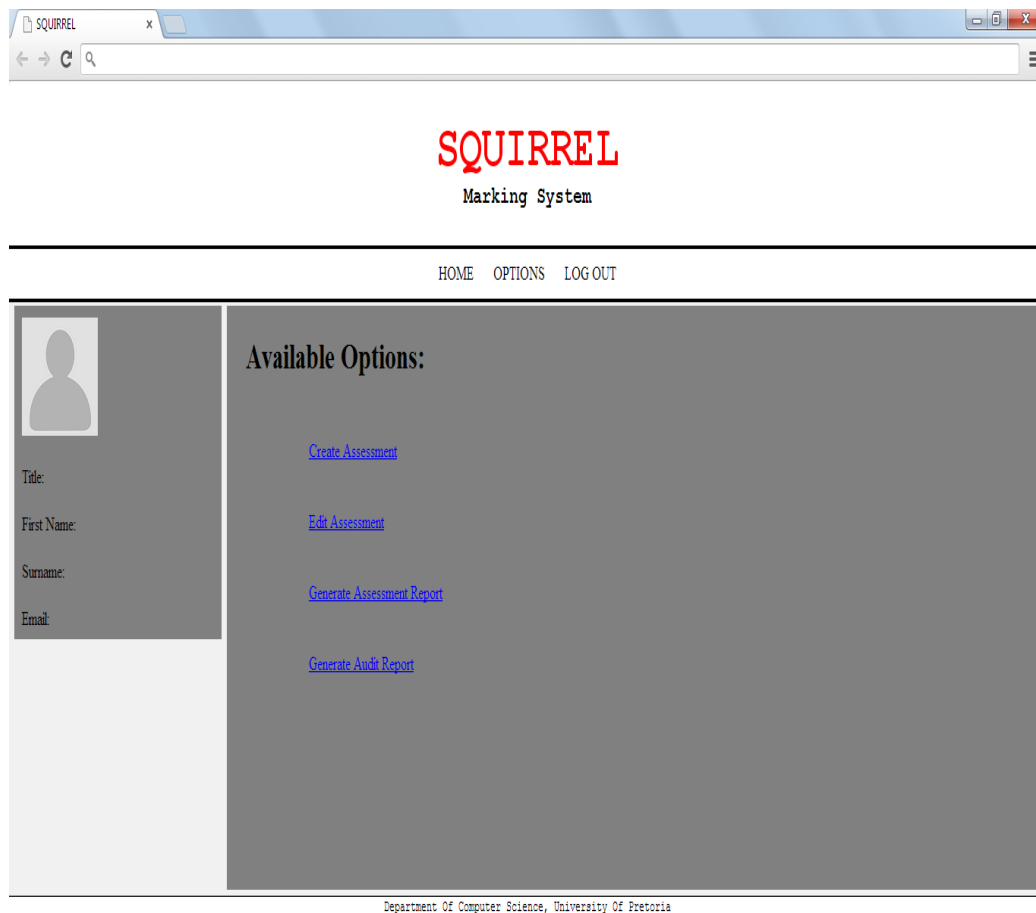


Figure 38: Report options lecturers have.

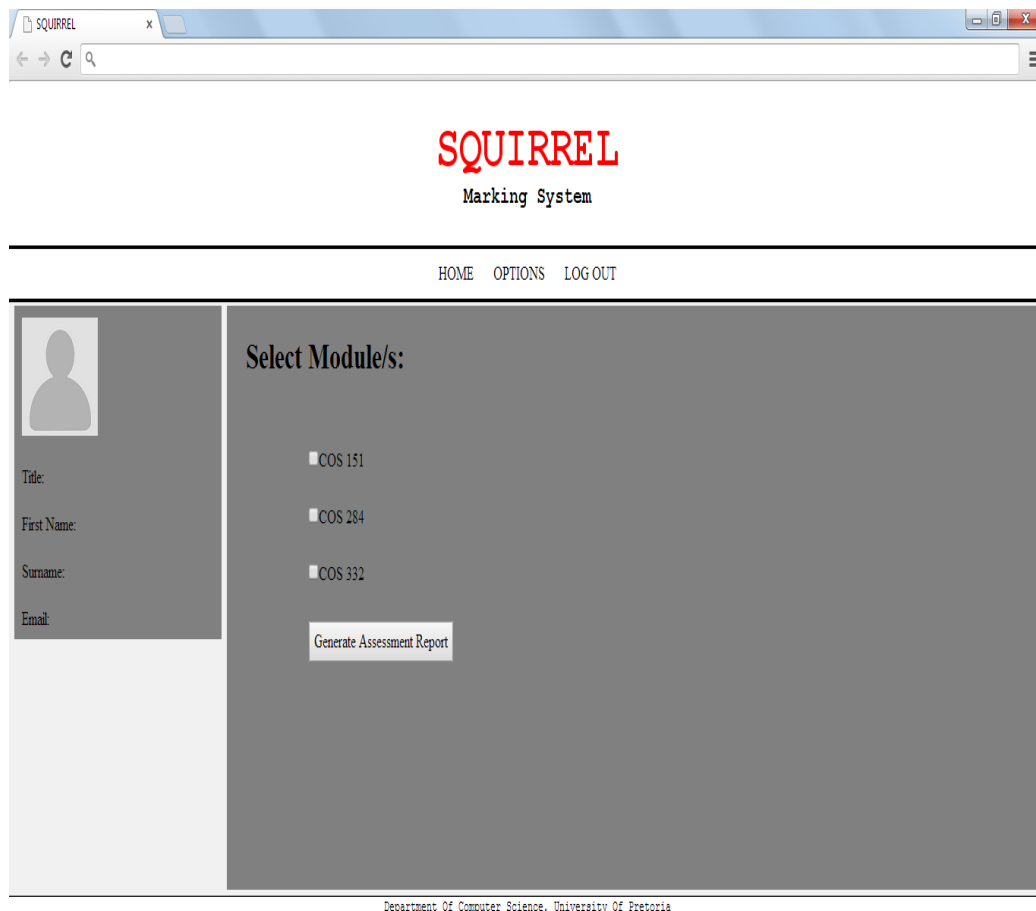



Figure 39: Screen design for generating assessment reports.



SQUIRREL

MARKING SYSTEM

HOME   OPTIONS   LOG OUT



Title:  
First Name:  
Surname:  
Email:

Select Events To Include In Audit Report:

☐ All assessment creations, modifications and removals

☐ All assessment session creations, modifications and removals

☐ All mark submissions, modifications and removals

☐ Any requests to open/close assessment sessions

☐ Any requests to publish marks

☐ Any requests for any reports including assessment reports, students marks reports and auditreports

Generate Audit Report

Department Of Computer Science, University Of Pretoria

Figure 40: Screen design for generating audit reports.