

Design Documentation

Group 1

Group Members:

Mbulungo Musetsho (10176382)

Ndivhuwo Ntambeleni (10001183)

Pule Legodi (29302732)

Lutfiyya Razak (10198408)

Luan van der Westhuizen (10134043)

Handre Watkins (10115405)

Matthew J Hughes (11371910)

Contents

1	Background	3
2	Vision	3
3	Software Architecture Design	3
3.1	Choices of Technologies	3
3.2	Chosen Frameworks	5
3.3	Chosen Protocols	6
3.4	Chosen Libraries	7
4	Application Design	7
4.1	Lower Levels of Granularity Specification	7
4.2	API Specifications	9
4.3	System Class Diagrams	10
4.4	System Process Specification	12
4.5	User Interface Designs	15
4.6	Database Desgin	23
5	Glossary	27

1 Background

Lecturers assign tutors or teaching assistants, also known as markers, to mark practical's and assignments. Markers are given a marking sheet with all the students registered for a marking session, which they record the students mark on. The mark sheet is then given to the lecturer for publication. However, most of the time the mark sheet is lost or marks are not recorded properly. Also there are privacy issues when mark sheets are published with all student names, student numbers and marks on it. Because of these issues the client, Jan Kroeze, proposed a mark management system which is accessible from a web browser and mobile devices.

2 Vision

The purpose of the project is to develop a software solution which provides a web and mobile platform for markers, students and lecturers handle marks and marking in the Department of Computer Sciences. It will uphold the privacy of students so that students cannot see another students mark. It will reduce paper work and the chances that marks can be lost and provide a centralized repository for student marks. It will provide functionality to generate customizable reports at different levels of granularity.

3 Software Architecture Design

3.1 Choices of Technologies

Programming Language

We are making use of Python as our programming language in order to implement the Django framework. Python is often viewed as a scripting language overlooking the fact that being as dynamic as it is, aids in rapid development. Python's standard library has more than 100 modules which coupled with Django makes it one of the most extensive rapid development packages.

Architectural Advantages:

- It is robust: Python has a relative small quantity of lines of code, which makes it less prone to issues, easier to debug, and more maintainable.
- Flexible: Python is highly scalable because it wasn't originally created to answer a specific need, Python isn't driven by templates or specific APIs, and is therefore well-suited to rapid development of all kinds of applications.

- Easy to LEARN AND USE: Since Python was not a language taught to us through the university, its ease of learning is definitely a aspect which is greatly valued

Database Technologies

- When starting a Django project it present you with a settings file in which database specifications and configurations can be made in order to manage the the systems models will work.
- It supports multiple databases - MySQL, PostgreSQL, Oracle and SQLite, MySQL being the chosen database we will incorporate into the system. The choice of MySQL deeply aids in Flexibility by providing the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. It's unique storage-engine architecture will aid in the concurrent querying that will take place.
- Django's object relational mapper is an incredibly powerful database tool. It handles creation of your database, as well as insert, update, delete queries and some quite advanced querying. This solves the architectual persistence constraint to a relational database.

Testing in Django

- Testing a Web application is a complex task, because a Web application is made of several layers of logic – from HTTP-level request handling, to form validation and processing, to template rendering. With Django's test-execution framework and assorted utilities, you can simulate requests, insert test data, inspect your application's output and generally verify your code is doing what it should be doing.

Application Servers

- Support for both development and production servers has to be evident for operations in decoupling the various capabilities for the respective situations.
- Django has a built in server which can be used for testing on one local machine. The buiilt in local development server is however not applicable in a production setting due to the fact that this server does not go through any form of security or performance tests its simply for local development before production development.
- Django can be run in conjunction with Apache, NGINX using WSGI, or Cherokee using a Python module called flup. Django also includes the ability to launch a FastCGI server, enabling use behind any web server which supports FastCGI, such as Lighttpd or Hiawatha

3.2 Chosen Frameworks

Software architecture framework provides the specification to developers how to organize and display the chosen libraries and design framework of a program. The framework provides a tool for the developers to use during the design of the software to include and implement the non-functional requirements as specified by the master specifications.

Object/Relational Mappers

- Object and relational mappers can be used by the developers to allow the integration and access of repositories without the continuous repetition of code. The mappers serve as a developer defined interface translator, that allows the integration of various systems without violating the non-functional requirements of security. This technique can be implemented in the design of the program allowing the cross communication from the repositories to the object orientated design.
- The use of new database with fields from the LDAP repository and CS MySQL database, as specified in the requirement documentation, will be used repetitively in all process interactions (4.1.2). Continuous use of these repository lookups can be minimised with the use of ORM model. The integration technique allows the developer to minimise the coding used in the interaction points by separating the query lookup and the implementation of the lookup request. In the implementation of the mappers, the developer must ensure that the data crossover is in the correct format and type as multiple data structures can be accessed that differ from the original data input.

Web Frameworks

- Web application framework allows the design and support of dynamic website and web services. As specified by the client, Django web framework will be used for the development of the web services. Django is an open source high level framework that implements the model-view-controller architecture pattern. The application is a Python based program that emphasizes the reusability of code and integration without the need of repetition. The framework implements an Object-relational mapper with a dynamic API for database access.
- One of the most attractive aspects of Django is its built in Admin capabilities, in other words creating separate roles and capabilities for separate users. This is a prerequisite for the system in order to provide differentiation between lecturers and students and allow us to manage access capabilities for each stakeholder.
- Django consists of a built in forms module which is the other greatly appreciated aspect of this framework. It provides extensive capabilities of validation based on various specifications, can even generate and update your database from a database model you create, make your job even easier.

Web service frameworks

- Web service framework allows the developer to decentralise inoperability and design front end services for web based applications. Frameworks allow the developer to improve on stability and design and allows connectivity on a word wide basis.
- As per client request the web service framework must be an Apache based web service to be allowed to be run on an Apache web server. Apache CXF web service is an open source web service framework that allows the development of services using up frontend programming APIs. The web service allows the incorporation of a variety of protocols such as the SOAP, RESTful HTTP. This conforms to the constraints set by the client in regards to the protocols that must be used in the development of the program.

3.3 Chosen Protocols

1. Simple Object Access Protocol (SOAP)
All systems (devices) should be able to access the system's content through the SOAP-based web services.
 - (a) The devices (systems) will send a request, procedure call, or a message such as getMark to search or retrieve information (average mark for the module, number of distinctions) through the SOAP-based web services.
 - (b) The SOAP-based web service will process the request or message and send a response to the devices (systems) in a form of a structured XML-based data.
2. Lightweight Directory Access Protocol (LDAP)
 - (a) It is used to authenticate the system users (Lecturers, Students, and Teaching Assistants) on login for single Sign On.
 - (b) It is used to retrieve the student's personal details and class lists, including courses assigned to lectures and students.
 - (c) It is used to retrieve system users roles such as, is the user a Lecturer, Student, or teaching assistant for the Module
3. Hypertext Transfer Protocol over Secure Socket Layer (HTTPS and HTTP)
 - (a) Used to send and retrieve sensitive data(marks and modules) through a secured (HTTPS) channel.
 - (b) Used to send a GET and POST requests to the web server and web services. Such as adding or updating a student mark.

3.4 Chosen Libraries

1. Python ldap library

It is used to integrate LDAP to the Mark sheet system.

- (a) It provides a connection to the CS LDAP system for authentication using `ldap.initialize()` and `ldap.open()` functionalities.
- (b) It also provides a mechanism for retrieving information from the directory, using functions such as `LDAPObject.result()`
- (c) It provides also a list of other LDAP request and response methods

2. pyPDF library

It is a Python library for generating PDFs.

- (a) It is used to generate pdf files or reports using the python programming language.
- (b) It provides different functionalities such as `PdfFileWriter` class which writes and add pages to the pdf and the merging of different pages.

3. MySQLdb Library

It is a python database library that integrates the MySQL database.

- (a) It provides functions such as `MySQLdb.connect()` for establishing a connection to the MySQL database.
- (b) It provides also a methods that executes database queries such as INSERT, DELETE, UPDATE, and SELECT.

4. The CSV File library

It is the python based library, for creating CSV files.

- (a) It provides the functionality to import and export CSV files.

4 Application Design

4.1 Lower Levels of Granularity Specification

- User Log In
 - View: A log in screen is displayed, wherein the user enters their details then submit to log in. The user will receive a log in result based on the validity of their credentials

- Controller: The Controller receives the credentials entered by the user and transfers them to the model for validation. The controller will create a response object based on the validity of the credentials.
- Model: The model receives the credentials and simply finds a match from LDAP. It sends the result of the validation back to the controller.
- Leaf Assessments
 - View: On the displayed screen, the user selects one of the following:
 1. Add New Leaf Assessment: The user enters all the required information to create a leaf assessment then submits. This will cause a new Assessment to be created and saved into the database, unless there was an error or exception (in which case an exception is thrown and the user is informed of the error)
 2. Search Existing Leaf Assessment (for Update/Delete): If record(s) matching the search were found, they are displayed for the user to update or delete, else then the user is informed of anything that possibly went wrong.
 - Controller: The controller receives, based on the kind of operation the user selects, the required information and sends it to the Model for either insertion, update, or deletion.
 - Model: The model will respectively receive the required information and either save a new leaf assessment, update or delete an existing leaf assessment. After the respective operation, the model will notify the view of changes made to the database.
- Aggregate Assessment:
 - View: The user enters all Aggregate Assessment information as required. When they submit, a new Aggregate Assessment is created based on the specified information.
 - Controller: The controller uses the provided aggregator to process the actual aggregate assessment then it sends it to the Model for storage.
 - Model: The model receives the aggregate assessment information and provides the controller with the specified aggregator. Then the model receives the processed aggregate assessment and stores it into the database then notifies the user about the results of the operation.
- Assessment Reports
 - View: The user specifies all required information (including the Frequency Analysis Information) and then submit. They will then choose whether they want it rendered on their device application or downloaded in either PDF or CSV format. Based on their choice of view, the results will be displayed.

- Controller: The controller retrieves the report specification submitted by the user and sends it to the model to find matches. The controller will then create a response object with the results and send it to the user for viewing.
 - Model: The model receives the report specification from the controller and then searches against the database for possible matches then it returns the matches back to the controller.
- Student Marks Reports
 - View: The user specifies all required information (including aggregation specifications) and then submit. They will then choose whether they want the results rendered on their device application or downloaded in either PDF or CSV format. Based on their choice of view, the results will be displayed.
 - Controller: The controller retrieves the marks report specification submitted by the user and sends it to the model to find matches. The controller will then create a response object with the results and send it to the user for viewing.
 - Model: The model receives the marks report specification from the controller and then searches against the database for possible matches then it returns the matches back to the controller.
- Audit Reports
 - View: The user enters all the specifications for the Audit Report then submit. They will then choose whether they want the results rendered on their device application or downloaded in either PDF or CSV format. Based on their choice of view, the results will be displayed.
 - Controller: The controller retrieves the Audit Report specifications submitted by the user then send it to the Model to retrieve matches. These matches will be sent back to the controller and then sent back to the user for viewing.
 - Model: The model receives the Audit Report specifications, retrieves matches of these specifications, and then return them to the controller.

4.2 API Specifications

- Different API's are to be used throughout the whole project, these are summarised below according to the interface they fall under:
 - View: Android API, HTTP/HTML 5 API, CSS API.
 - Controller: Python Django API (Main API). (PHP API and JAVA API might be used under certain conditions).

- Model: mySQL API, LDAP API.

4.3 System Class Diagrams

- Leaf Assessment, Mark allocation and assessment aggregation Class diagram
 - The class diagram uses the abstract factory design pattern to represent the lower level relationship between an assessment, it's aggregation and the mark allocation.
 - Although all these processes are done in different APIs, this class diagram represents them in a manner that seems like they are on one interface just to make it better to understand.

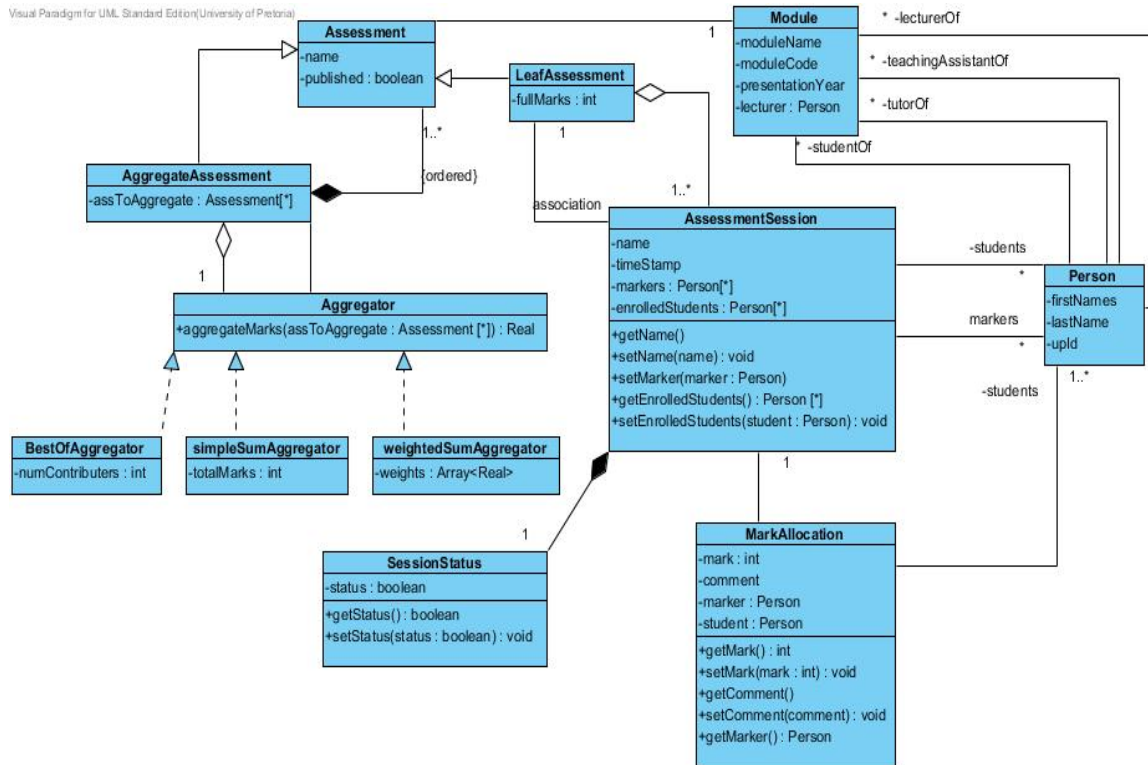


Figure 1: Assessment Marking and aggregation Class Diagram

- Assessment Report Class Diagram
 - Just like the previous class diagram, this the assessment report class diagram uses the abstract factory design pattern to show how each report is related to a specific assessment.

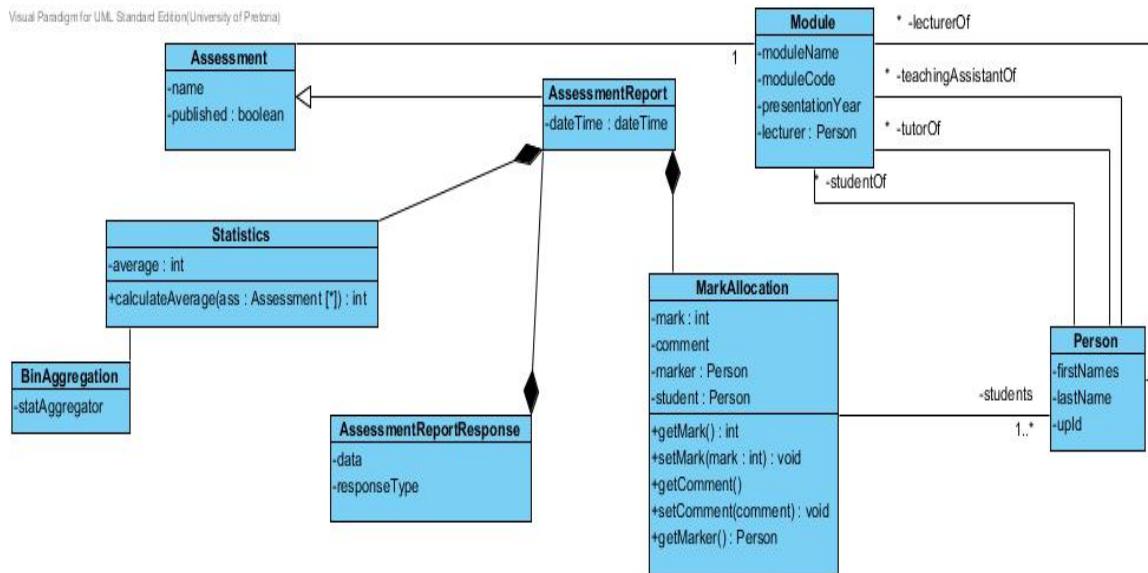


Figure 2: Assessment Report Class Diagram

- Student Mark Report Class Diagram:

- This is an abstract factory class diagram, like the others
- Shows how the marks are generated for each student when they are logged on to the system.

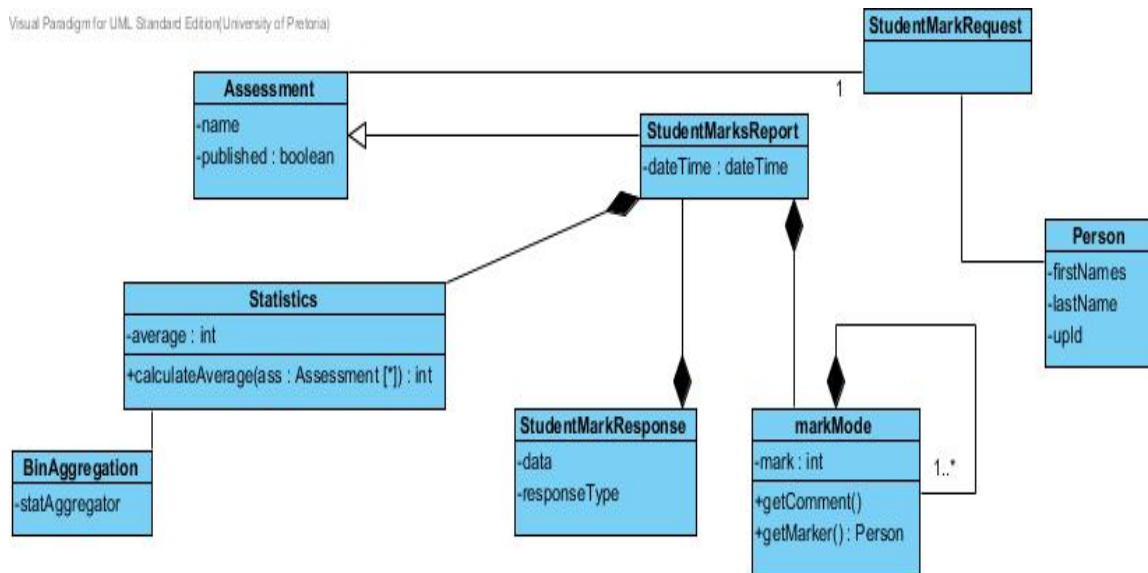


Figure 3: Student Mark Report Class Diagram

4.4 System Process Specification

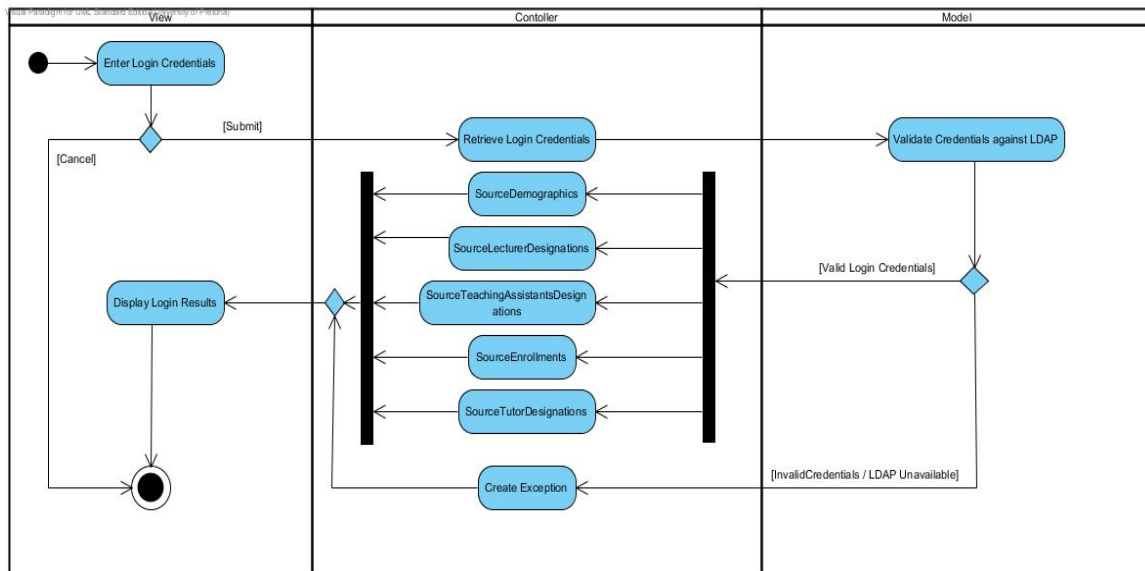


Figure 4: User Log In Activity Diagram

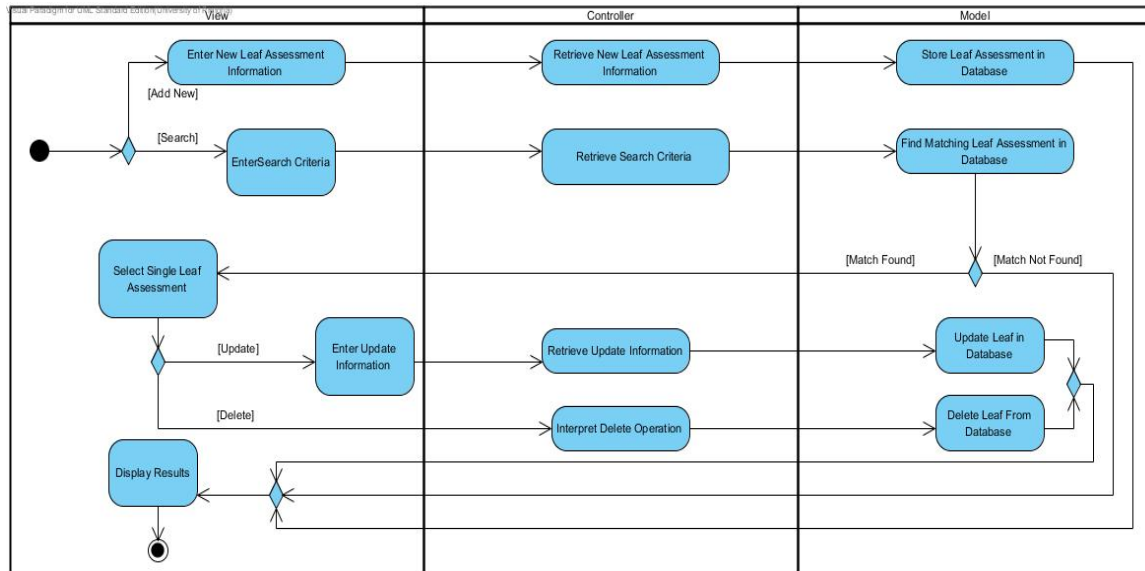


Figure 5: Leaf Assessment Activity Diagram

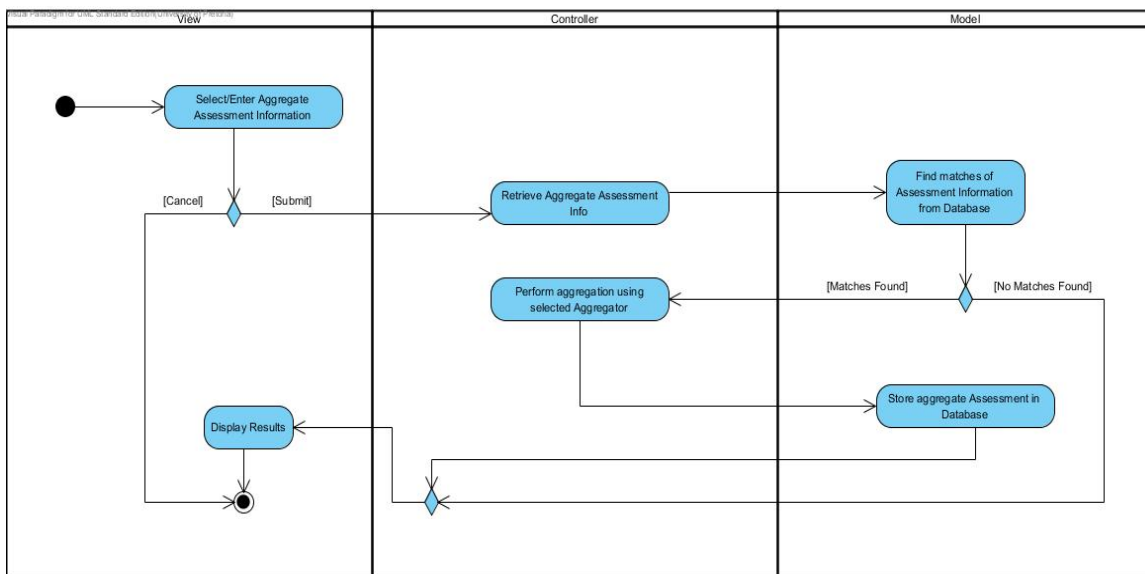


Figure 6: Aggregate Assessment Activity Diagram

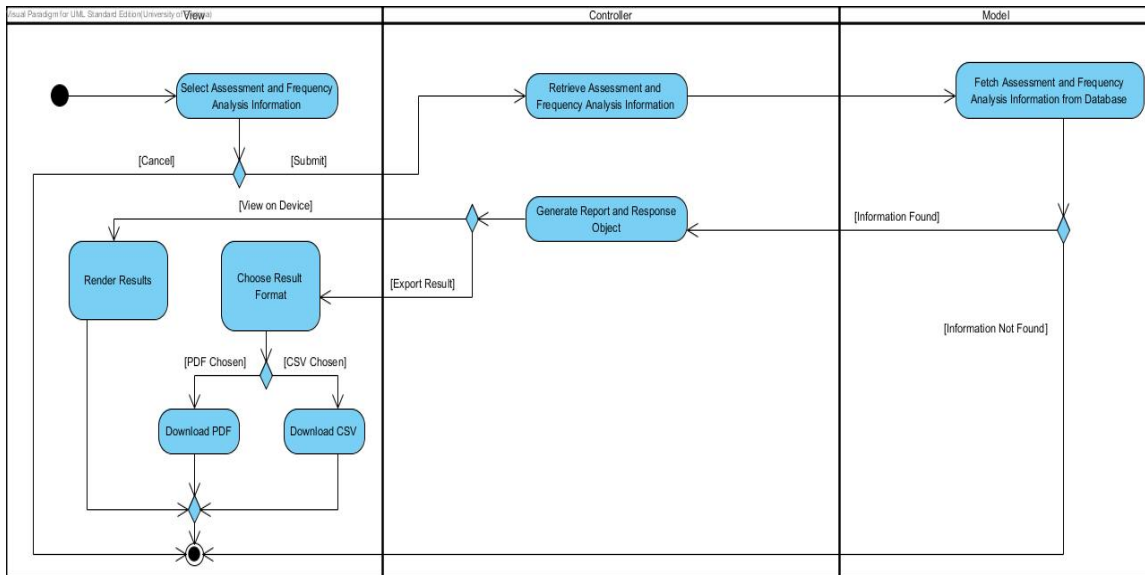


Figure 7: Assessment Report Activity Diagram

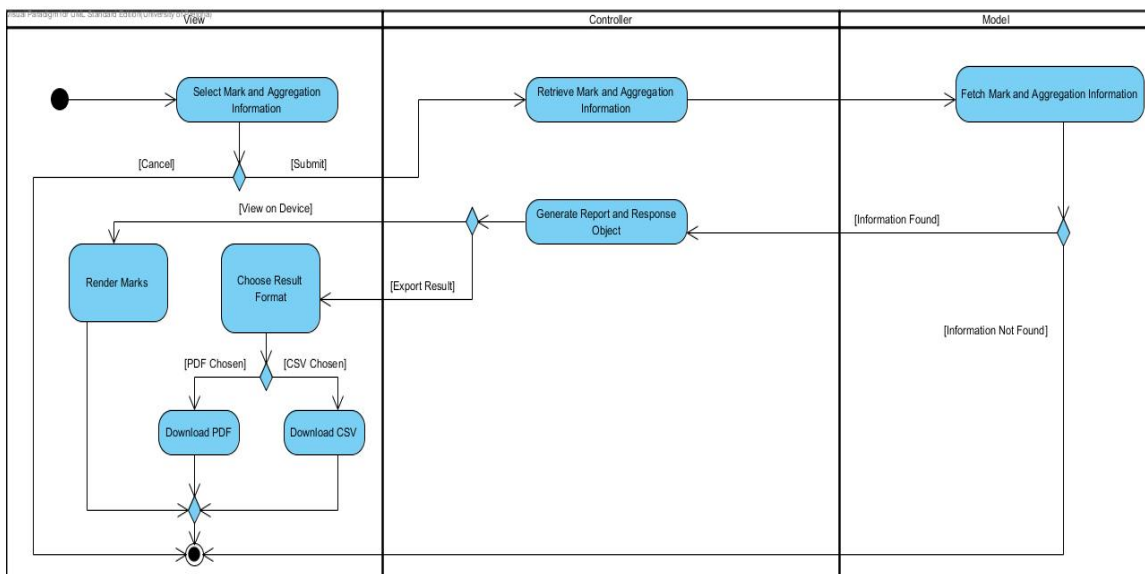


Figure 8: Student Marks Report Activity Diagram

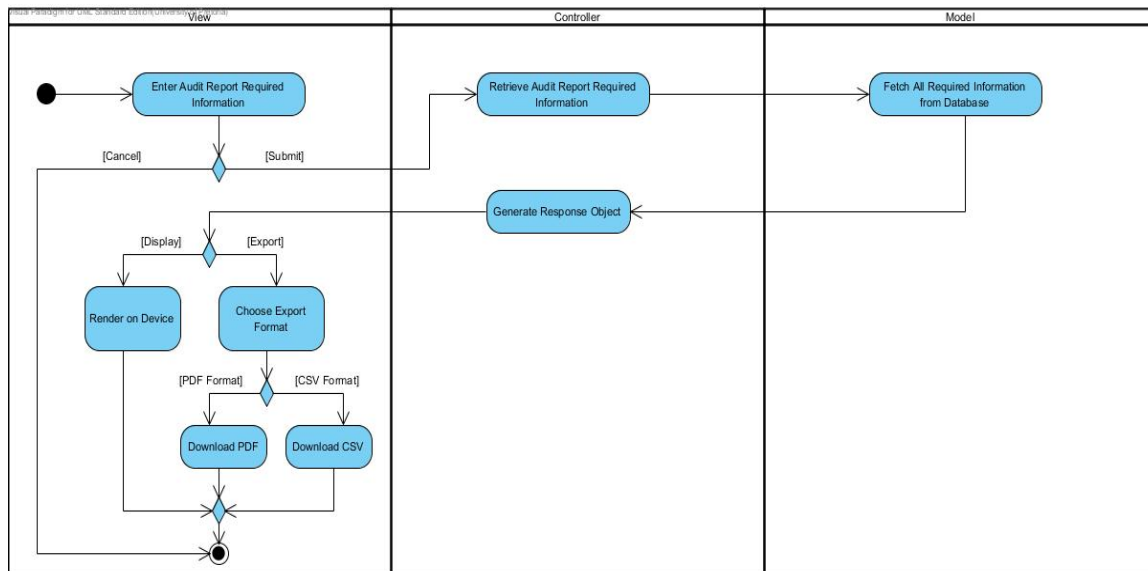
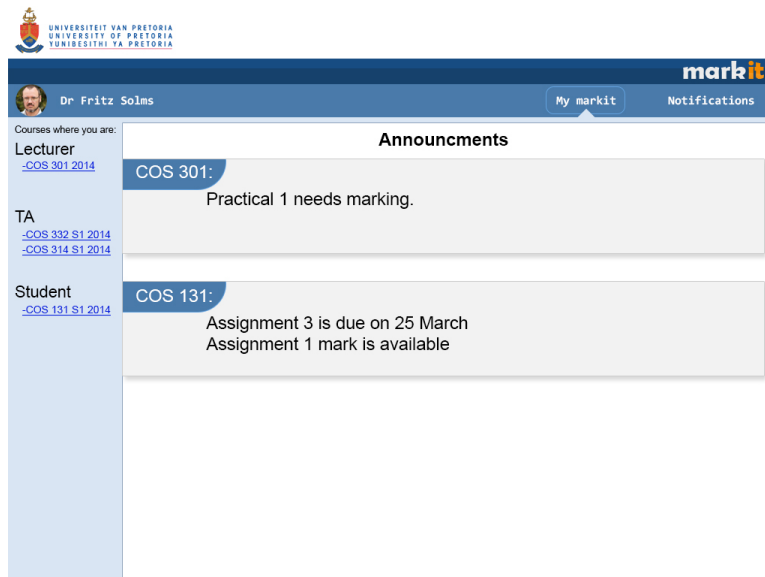


Figure 9: Audit Report Activity Diagram

4.5 User Interface Designs

The screenshot shows the 'markit' login page for UP Staff and Students. It includes a header with the University of Pretoria logo and name in three languages. A blue banner contains a note about using 'u' + your 8 digit number for the username. Below this is a 'Login' section with 'Username' and 'Password' input fields and a 'Login' button.

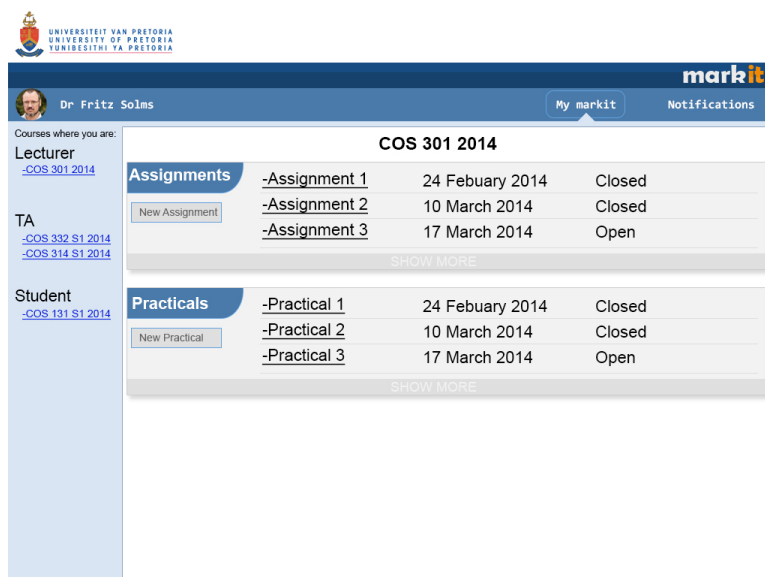
Figure 10: Log In Interface



The screenshot shows the Markit Home Interface. At the top, there is a header with the University of Pretoria logo and name in three languages, a user profile for Dr. Fritz Solms, and navigation links for 'My markit' and 'Notifications'. The main content area is titled 'Announcements'. On the left, a sidebar lists roles: Lecturer (with link -COS 301 2014), TA (with links -COS 332 S1 2014 and -COS 314 S1 2014), and Student (with link -COS 131 S1 2014). The announcements list includes:

- COS 301:** Practical 1 needs marking.
- COS 131:** Assignment 3 is due on 25 March, Assignment 1 mark is available.

Figure 11: Home Interface




The screenshot shows the Markit Subject Interface for 'COS 301 2014'. The header and sidebar are identical to Figure 11. The main content area is titled 'COS 301 2014' and contains two sections: 'Assignments' and 'Practicals'. Each section has a 'New' button and a table of activities.


Assignments			
-Assignment 1	24 Febuary 2014	Closed	
-Assignment 2	10 March 2014	Closed	
-Assignment 3	17 March 2014	Open	
SHOW MORE			

Practicals			
-Practical 1	24 Febuary 2014	Closed	
-Practical 2	10 March 2014	Closed	
-Practical 3	17 March 2014	Open	
SHOW MORE			

Figure 12: Subject Interface



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA


Dr. Fritz Solms

My markit
Notifications

Courses where you are:
Lecturer
[-COS 301 2014](#)

TA
[-COS 332 S1 2014](#)
[-COS 314 S1 2014](#)

Student
[-COS 131 S1 2014](#)

COS 301
Assignment 1


Details

Created on: 18 February 2014
Due Date: 24 February 2014
Total Marks 20


Students

Student No.	Submission Date	Mark Last Edited	Mark
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked
u123456789	23/03/2014 16:31	n/a	unmarked

Figure 13: Assignments Interface



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA


Dr. Fritz Solms

My markit
Notifications

Courses where you are:
Lecturer
[-COS 301 2014](#)

TA
[-COS 332 S1 2014](#)
[-COS 314 S1 2014](#)

Student
[-COS 131 S1 2014](#)

COS 301
Assignment 1


Details

Student No: u123456789
Submission Date: 23/03/2014 16:31
Mark n/a


Submission

Question	Answer	Mark Obtained	Total Marks	Comment
1.1	[Answer text.... Answ...]	-	2	
1.2	[Answer text.... Answ...]	-	1	
1.3	[Answer text.... Answ...]	-	1	
1.4	[Answer text.... Answ...]	-	1	
1.5	[Answer text.... Answ...]	-	10	
2.1	[Answer text.... Answ...]	-	5	
2.2	[Answer text.... Answ...]	-	2	
2.3	[Answer text.... Answ...]	-	2	
2.4	[Answer text.... Answ...]	-	20	
3.1	[Answer text.... Answ...]	-	12	
4.1	[Answer text.... Answ...]	-	1	

Figure 14: Leaf Assignment Interface



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA


Dr. Fritz Solms

My markit
Notifications

Courses where you are:
Lecturer
[-COS 301 2014](#)

TA
[-COS 332 S1 2014](#)
[-COS 314 S1 2014](#)

Student
[-COS 131 S1 2014](#)

COS 301
Practical 1

Details

Created on: 18 Febuary 2014
Due Date: 24 Febuary 2014
Total Marks 20

Students


Student No.	Mark
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62
u123456789	62

Marking


View Markers

Students: 76
Unmarked Students: 2
Average Mark: 56%
Median Mark: 62%
Pass Rate: 79%
Distinctions 5%

Figure 15: Practicals Interface



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA


Dr. Fritz Solms

My markit
Notifications

Courses where you are:
Lecturer
[-COS 301 2014](#)

TA
[-COS 332 S1 2014](#)
[-COS 314 S1 2014](#)

Student
[-COS 131 S1 2014](#)

COS 301
Practical 1

Details

Created on: 18 Febuary 2014
Due Date: 24 Febuary 2014
Total Marks 20

Markers

Click a marker to edit their details
Add Marker

Student No.	Name	Group No.	Room
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2
u123456789	John Smith	1	Blue 2

Figure 16: Practical Markers Interface



Figure 17: Mobile Interface: Login Screen

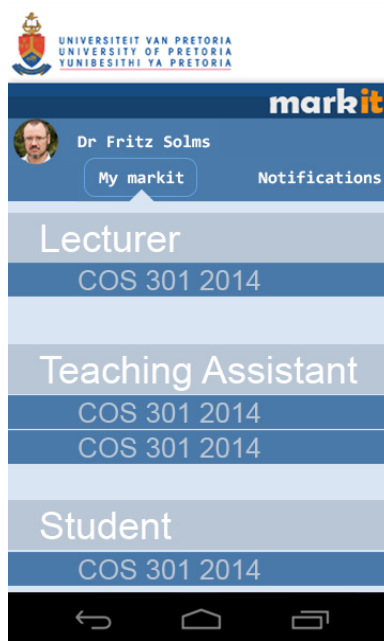


Figure 18: Mobile Interface: Home Screen

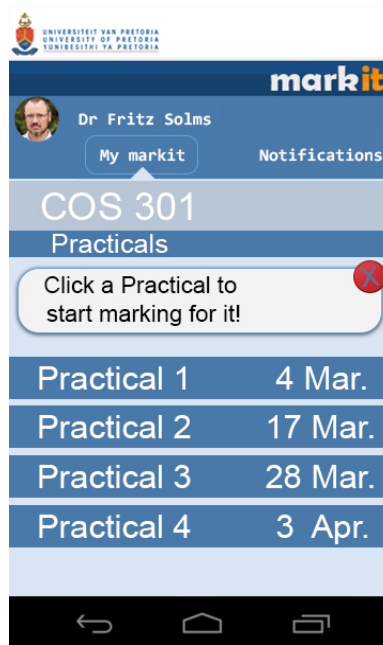


Figure 19: Mobile Interface: Subject Screen

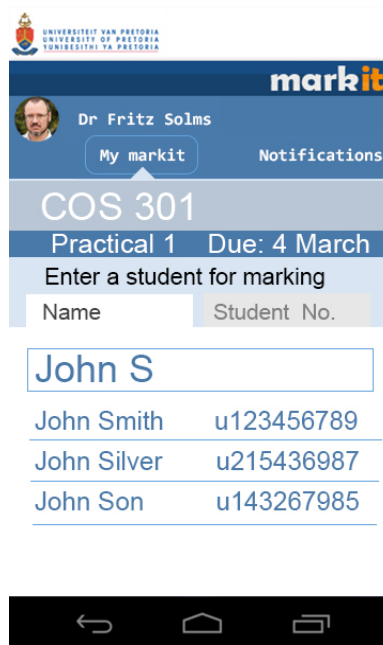


Figure 20: Mobile Interface: Marking Menu Screen

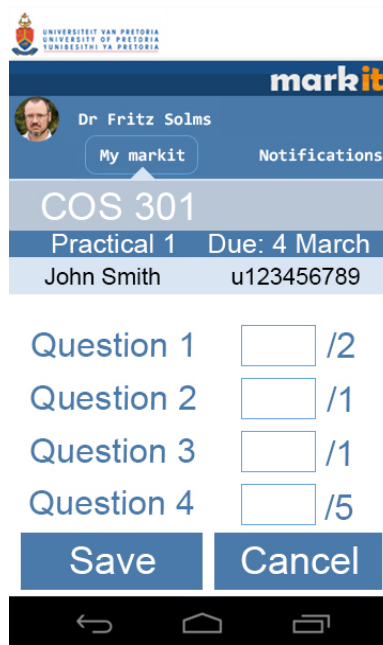


Figure 21: Mobile Interface: Leaf Marking Screen

4.6 Database Desgin

Table 1: Module

Field	Type	Null	Key	Default	Extra
id	int(11)	No	Primary	Null	auto_increment
code	varchar(20)	No		0	
name	varchar(20)	No		Null	
lecturer	varchar(20)	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	

Table 2: Session

Field	Type	Null	Key	Default	Extra
id	int(11)	No	Primary	Null	auto_increment
code	varchar(20)	No		0	
marker	varchar(20)	No		Null	
status_open	bool	No		0	
start_time	datetime	No		0000-00-00 00:00:00	
end_time	datetime	No		0000-00-00 00:00:00	
assessment_id	int(11)	No	Foreign	Null	
session_number	int(11)	No		Null	

Table 3: Assessment

Field	Type	Null	Key	Default	Extra
assessment_id	int(11)	No	Primary	Null	auto_increment
type	varchar(11)	No		0	
description	text	Yes		Null	
assessment_name	varchar(11)	No		0	
code	varchar(11)	No		0	
date_issued	datetime	No		0000-00-00 00:00:00	
date_due	datetime	No		0000-00-00 00:00:00	
mark_id	int(11)	No	Foreign	Null	
submission_details	text	No		Null	

Table 4: Mark Allocation

Field	Type	Null	Key	Default	Extra
mark_id	int(11)	No	Primary	Null	auto_increment
mark	varchar(10)	Yes		0	
comment	varchar(30)	Yes		0	
marker_name	varchar(20)	No		0	
timestamp	datetime	No		0000-00-00 00:00:00	
totalMark	varchar(20)	No		0	

Table 5: Simple Sum Aggregator

Field	Type	Null	Key	Default	Extra
ssa_id	int(11)	No	Primary	Null	auto_increment
assessment_id	int(11)	No	Foreign	Null	
Num_of_questions	varchar(20)	No		0	
full_marks	varchar(20)	No		0	

Table 6: Weighted Sum Aggregator

Field	Type	Null	Key	Default	Extra
wsa_id	int(11)	No	Primary	Null	auto_increment
assessment_id	int(11)	No	Foreign	Null	
num_of_assessments	varchar(20)	No		0	
calculate_weight	int(11)	No		Null	

Table 7: Best Of Aggregator

Field	Type	Null	Key	Default	Extra
boa_id	int(11)	No	Primary	Null	auto_increment
assesment_id	int(11)	No	Foreign	Null	
Best_aggregates	varchar(20)	No		0	
sum_marks	varchar(20)	No		0	

Table 8: Student Report

Field	Type	Null	Key	Default	Extra
sReport_id	int(11)	No	Primary	Null	auto_increment
timestamp	datetime	No		0000-00-00 00:00:00	
assessment_name	varchar(11)	No	Foreign	0	
ssa_id	int(11)	No	Foreign	Null	
wsa_id	int(11)	No	Foreign	Null	

Table 9: Audit Report

Field	Type	Null	Key	Default	Extra
ar_id	int(11)	No	Primary	Null	auto_increment
datestamp	datetime	No		0000-00-00 00:00:00	
userId	int(11)	No		Null	
action_performed	varchar(30)	Yes		0	
assessment_CRUD	varchar(30)	No		0	
sessions_CRUD	varchar(30)	No		0	
mark_CRUD	varchar(30)	No		0	
session_status_request	varchar(30)	yes		0	
publish_mark_request	varchar(30)	Yes		0	
report_request	varchar(30)	Yes		0	

Table 10: Assesment Report

Field	Type	Null	Key	Default	Extra
as_id	int(11)	No	Primary	Null	auto_increment
datestamp	datetime	No		0000-00-00 00:00:00	
class_average	varchar(20)	No		0	
frequency_analysis	varchar(20)	No		0	
assesment_id	int(11)	No	Foreign	Null	

5 Glossary

Term	Defintion
Android	A smart phone operating system developed by Google. It is used by a variety of mobile phone manufacturers including HTC, Sony, etc.
API	Application Programming Interface
CS	Computer Science
CSS	Cascading Style Sheet
Database	A collection of all information that is necessary to the functioning of a system
granularity	Extent to which a system is broken down into small parts
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
LDAP	Lightweight Directory Access Protocol
MySQL	open-source relational database management system (RDBMS)
ORM	Object-Relational Mapping
repository	central place where data is stored and maintained
RESTful	Representational state transfer
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language