Software Architecture Design

Group 6

March 13, 2014

D Doman [11002566]
P Hammond [11025477]
BW Müller [11037157]
JF Oberholzer [12039803]
AD Pretorius [12022404]
ER Rosslee [12223426]
R Schnetler [12220702]

Git Repository:
`https://github.com/BWMuller/COS301-MiniProject-G6-P2`

1 Software Architecture Design

This section discusses the software architecture requirements. It will specify how the software infrastructure that is to be developed will address the functional requirements. All of this is done within the architecture constraints given by the client. What follows is firstly a summary of the functional requirements, then a summary of the constraints given by the client, then a list of the chosen technologies, frameworks, protocols and libraries and a justification of why that choice was made and how it addresses the problem.

2 Functional Requirements

2.1 Access

The system will be accessible by humans from a browser through a rich web interface, this interface must be accessible through all recent versions of mayor browsers. There must also be an Adriod application to be accessed through mobile Android devices with all recent versions of Android.

2.2 Integration

The system must access a UP CS server and database to retrieve students' personal information and access course information. The server is LDAP and the database is MySQL. When the marks are exported that should be done in a CSV file. All communication must be done using secure HTTPS.

2.3 Security

Users must authenticate to the LDAP server before using any of the services. The service should also ensure that users only access the parts that they have access to.

2.4 Auditability

The system should keep track of any changes made to any entity in the database. It should record who makes that change, when do they make it and from what to what it was changed. The audit log cannot be modified.

2.5 Testability

All the services offered by the system should be testable. It should test if the service is provided under the right conditions and that post-conditions hold true after service has been provided.

2.6 Usability

The software should be easy to understand and use without prior training. It should be in English but it should be open to be translated to other languages.

2.7 Scalability

The system should be open to be scaled and expanded to hadle all the assesments for all the modules of the Department of Computer Science. The system should be able to hanle 100 users at the same time.

2.8 Performance

All non-reporting operations should report within less than 1 second. Report queries should be processed in no more than 10 seconds.

3 Architecture Constraints

The system should be developed using the Django web framework. Persistence to a relational database must be done using the Object-Relational Mapper bundled with Django. The Django unittest module should be used for testing. The system should be deployed on a Django application server running within the cs.up.ac.za Apache web server. The mobile client should be running on an Android application. The system will use the MySQL database. Web services must be either SOAP-based or Restful web services.

4 Architecture Design

4.1 Access

There should be a web application and a mobile application. For the web application we are going to use the Djanga Web Framework. This is the framework that the users wants, but also has many advantages. Django works with Python, one of the simplest programming languages out there. The structure is also very good and resolves around building the website in phases. Django also adheres strongly to the DRY (Don't repeat yourself) principle. For the mobile app we will use Android, since it is the most popular mobile OS out there.

4.2 Integration

The CS server already uses MySQL and so we will keep consistency and also use this. MySQL is also cost effective since it is open source. MySQL can work cross platform and has good security. We will also use the Object-Relational Mapper bundled with Django since we use Django.

4.3 Security

The security of MySQL is very, very good. The server has its own security, then the database has more security.

4.4 Auditability

Using the Django framework we will create an auditability program that will keep track of all the changes and store it on the server.

4.5 Testability

The Django framework includes a unittest module we can use to thouroughly test this system.

4.6 Usability

We will apply proper design principles like affordances to make the system clear and easy to use.

4.7 Scalability

Both Django and MySQL are scalable to the level we desire.

4.8 Performance

Both Django and MySQL can perform to the level we desire.

5 Application Design

5.1 The Back-End System

Specifications of Lower Levels of Granularity

Mobile

- **Interface**: Implements an interface to the system from the API (andoid.view).
- **Communication**: This is accomplished by using client server communication (secure script protocols). The communication must be done by implementing the Android API client side.
- **API**: Interactions to the interface must be done using the android.gesture class.
- All interactions must be validated and audited server-side.

User Login

- **Back End**: Receives user login information and confirms its validity using a compare functionality outside the Database interaction.

Leaf Assessment

- **Back End**: Get information on a leaf with condition to either create, delete or update. It the performs the operation and returns the success of the operation

Aggregate Assessment

- **Back End**: Get details of aggregation and applies them to relevant leaf and returns the result of the aggregation.

Assessment Report

- **Back End**: Receives the details specifying the range of assessments to include and collect all the relevant data and transmit it back.

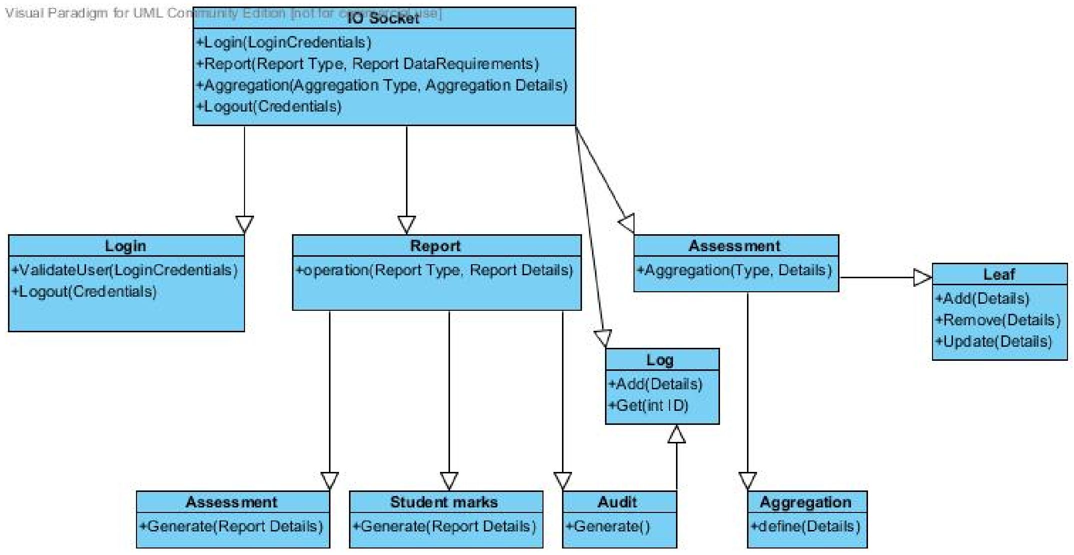
Student Report

- **Back End**: Receives the student details and collects the data required and returns it.

Audit Report

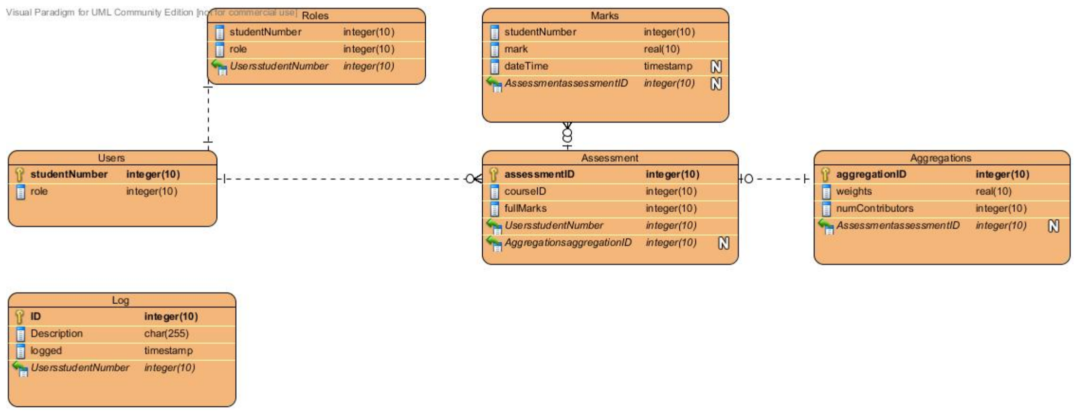
- **Back End**: Gathers the data for the required Audit and includes all default and relevant log information and returns the collection.

API Specifications



Visual Paradigm for UML Community Edition [not for commercial use]
IO Socket
+Login(LoginCredentials)
+Report(Report Type, Report DataRequirements)
+Aggregation(Aggregation Type, Aggregation Details)
+Logout(Credentials)
Login
+ValidateUser(LoginCredentials)
+Logout(Credentials)
Report
+operation(Report Type, Report Details)
Assessment
+Aggregation(Type, Details)
Leaf
+Add(Details)
+Remove(Details)
+Update(Details)
Log
+Add(Details)
+Get(int ID)
Assessment
+Generate(Report Details)
Student marks
+Generate(Report Details)
Audit
+Generate()
Aggregation
+define(Details)

Figure 1: Back-End API Specification



Visual Paradigm for UML Community Edition [not for commercial use]
Roles
studentNumber integer(10)
role integer(10)
UsersstudentNumber integer(10)
Marks
studentNumber integer(10)
mark real(10)
dateTime timestamp N
AssessmentassessmentID integer(10) N
Users
studentNumber integer(10)
role integer(10)
Assessment
assessmentID integer(10)
courseID integer(10)
fullMarks integer(10)
UsersstudentNumber integer(10)
AggregationsaggregationID integer(10) N
Aggregations
aggregationID integer(10)
weights real(10)
numContributors integer(10)
AssessmentassessmentID integer(10) N
Log
ID integer(10)
Description char(255)
logged timestamp
UsersstudentNumber integer(10)

Figure 2: Back-End Entity Relationships

5.2 Web Application

The web application of the Squirrel Marking System provides a browser-based user interface in the application.

Lower Levels of Granularity

The lower levels of granularity consists of 3 main users:

Student: This user is first required to log in before he/she can use the system. The only functions permitted to the student to perform on this system is to view his registered modules' marks and render his marks onto the web screen, CSV or PDF file.

Marker: This user is also required to log in before he/she can use the system. A marker can be a lecturer or student. A marker can only mark leaf or aggregate assessments of certain students. He can submit,add,modify and delete students' marks.

Lecturer: This user is also required to log in before he/she can use the system. The lecturer is by default also a marker.The lecturer can define and control assessments,that is he/she can create,modify and delete assessments as well as create, modify, lock and unlock assessment sessions. He/she can also generate reports for a specified assessment, specified student or audit log , having it rendered on the web user interface, CSV or PDF document and publish students' marks.

User Work Flow Diagrams

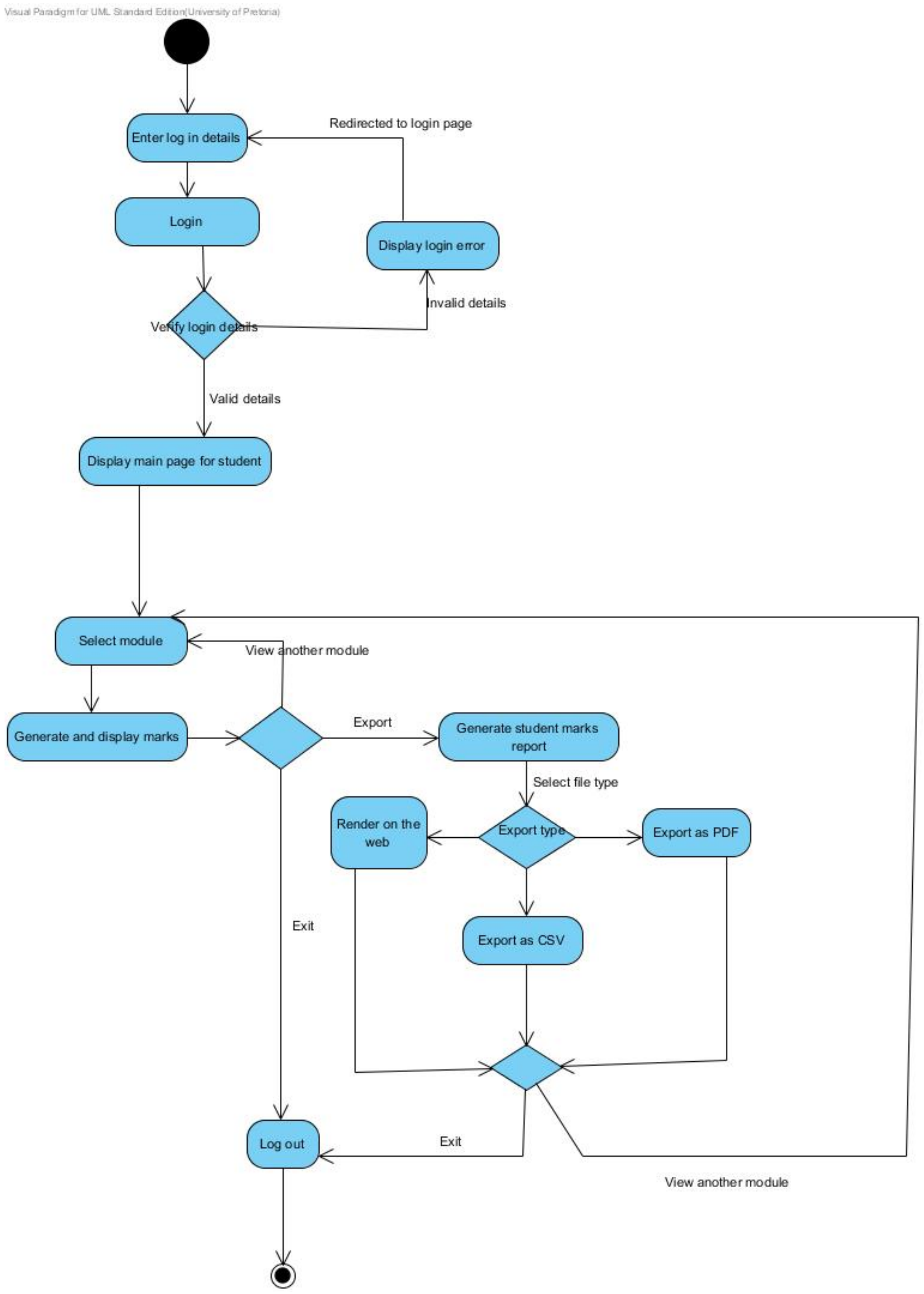


Figure 3: Activity Diagram for the Student.

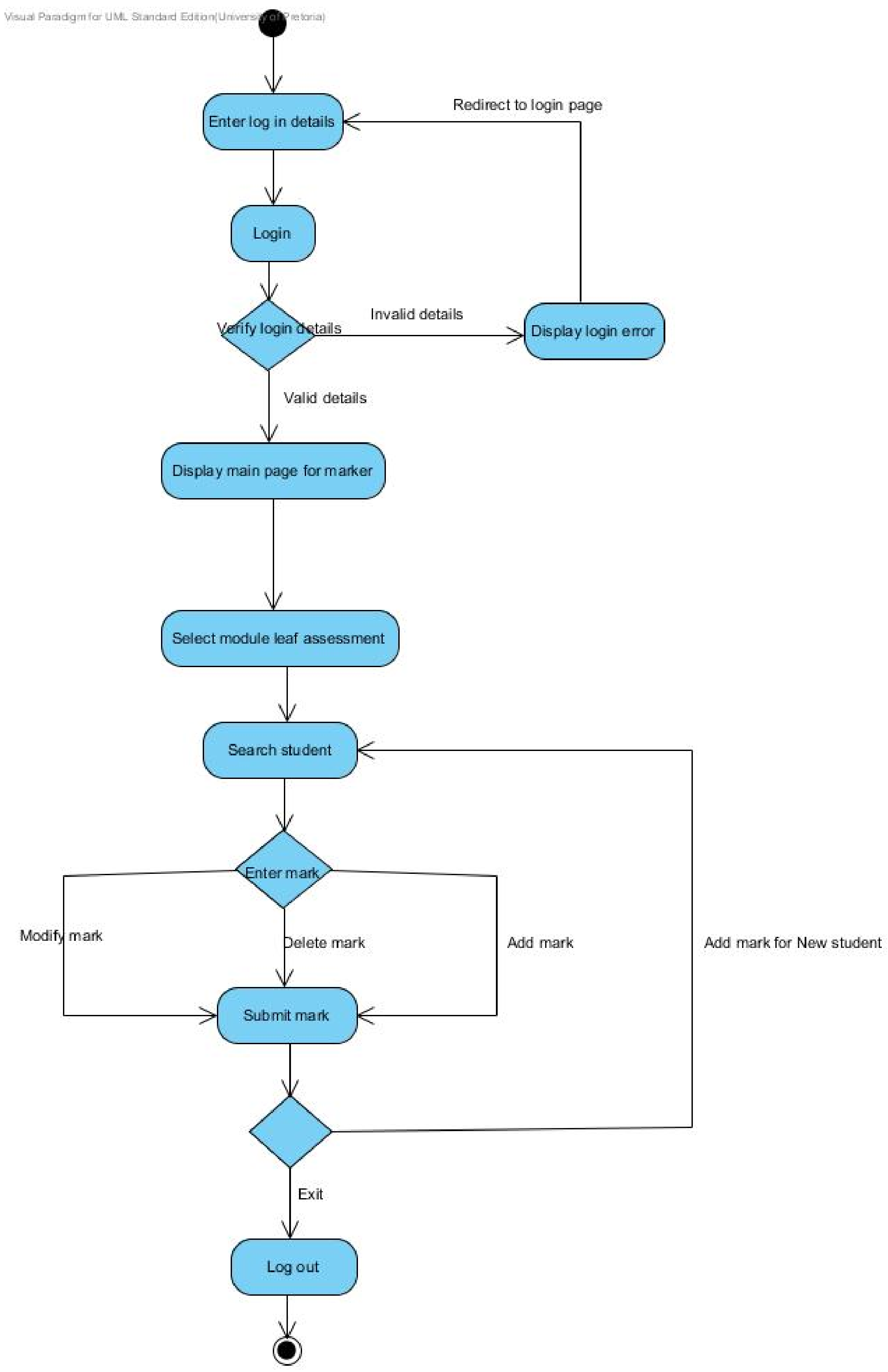


Figure 4: Activity Diagram for the Marker.

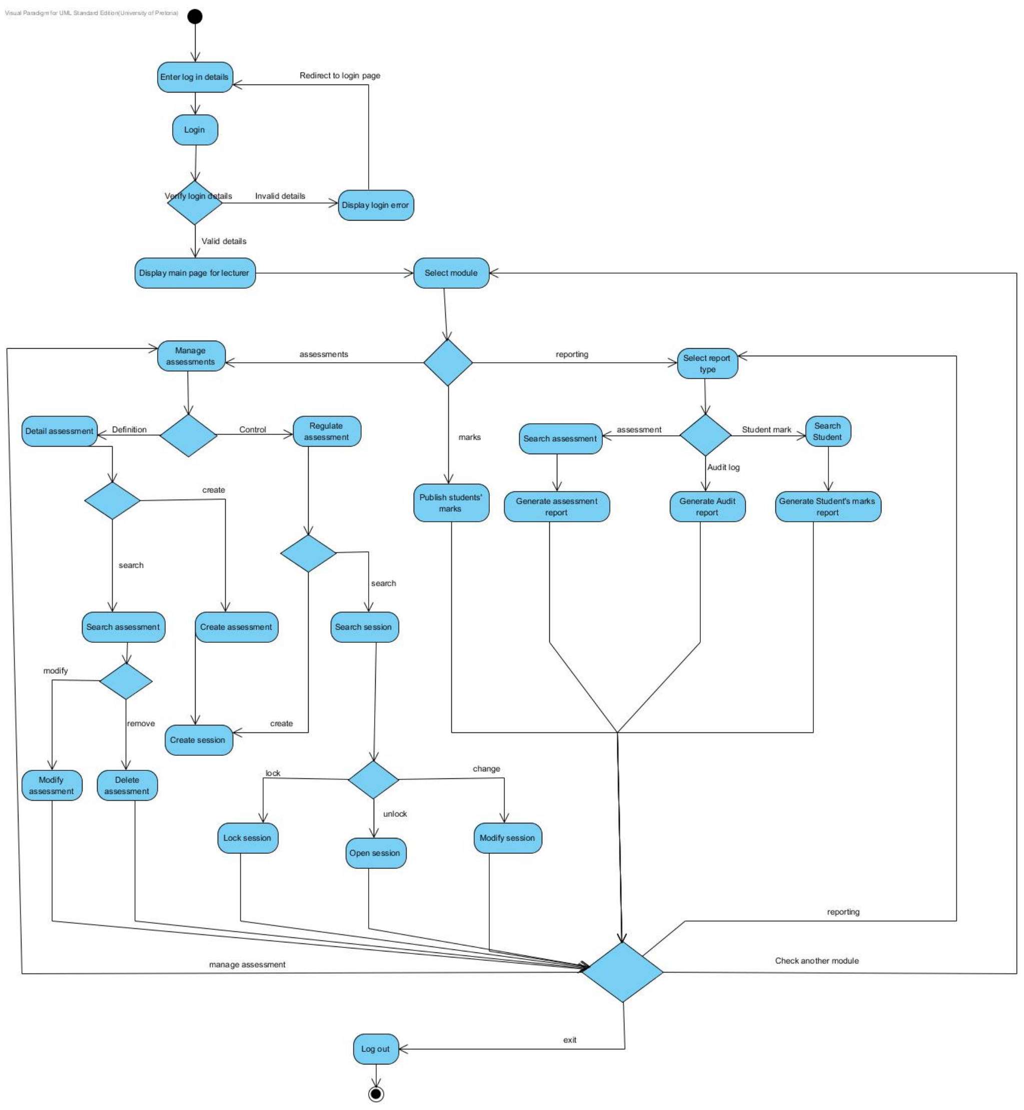
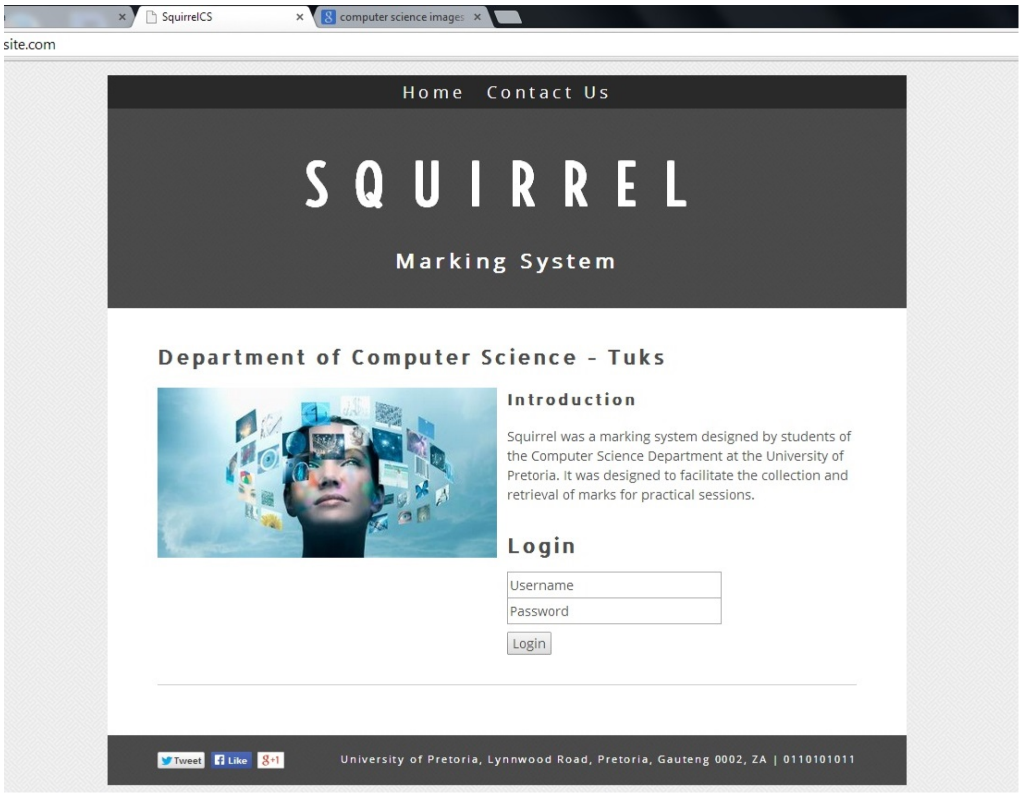


Figure 5: Activity Diagram for the Lecturer.

User Interface Diagrams


SquirrelCS
computer science images
site.com

Home Contact Us

SQUIRREL

Marking System

Department of Computer Science - Tuks

Introduction

Squirrel was a marking system designed by students of the Computer Science Department at the University of Pretoria. It was designed to facilitate the collection and retrieval of marks for practical sessions.

Login

Username
Password
Login

Tweet Like g+1 University of Pretoria, Lynnwood Road, Pretoria, Gauteng 0002, ZA | 0110101011

Figure 6: The Login Screen where a lecturer or student logs in using his/her CS details

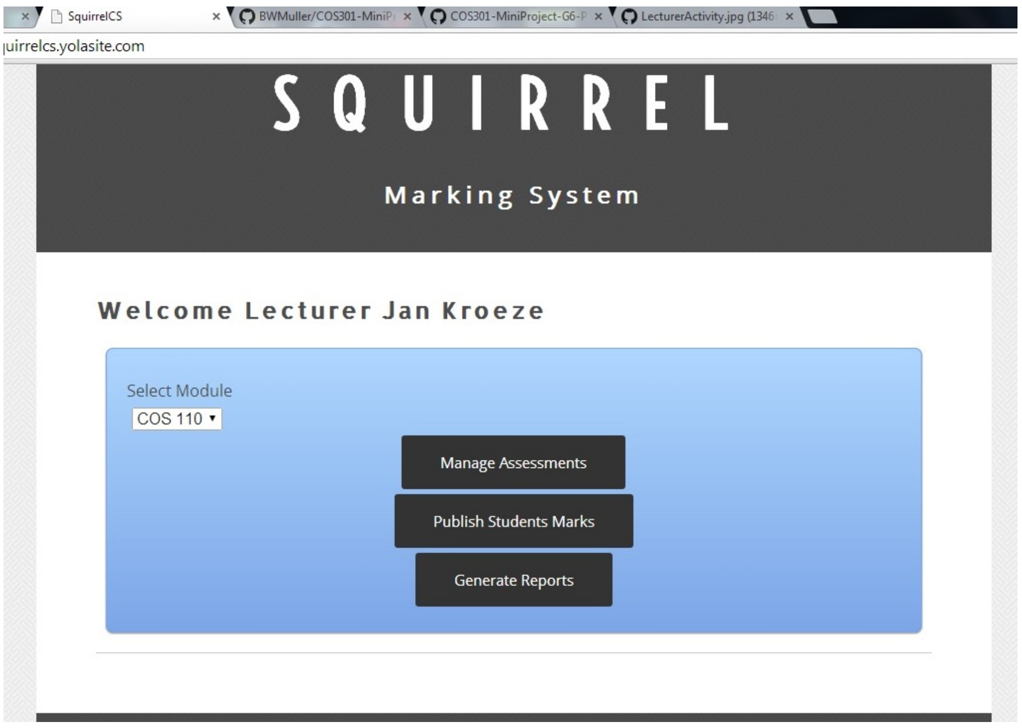
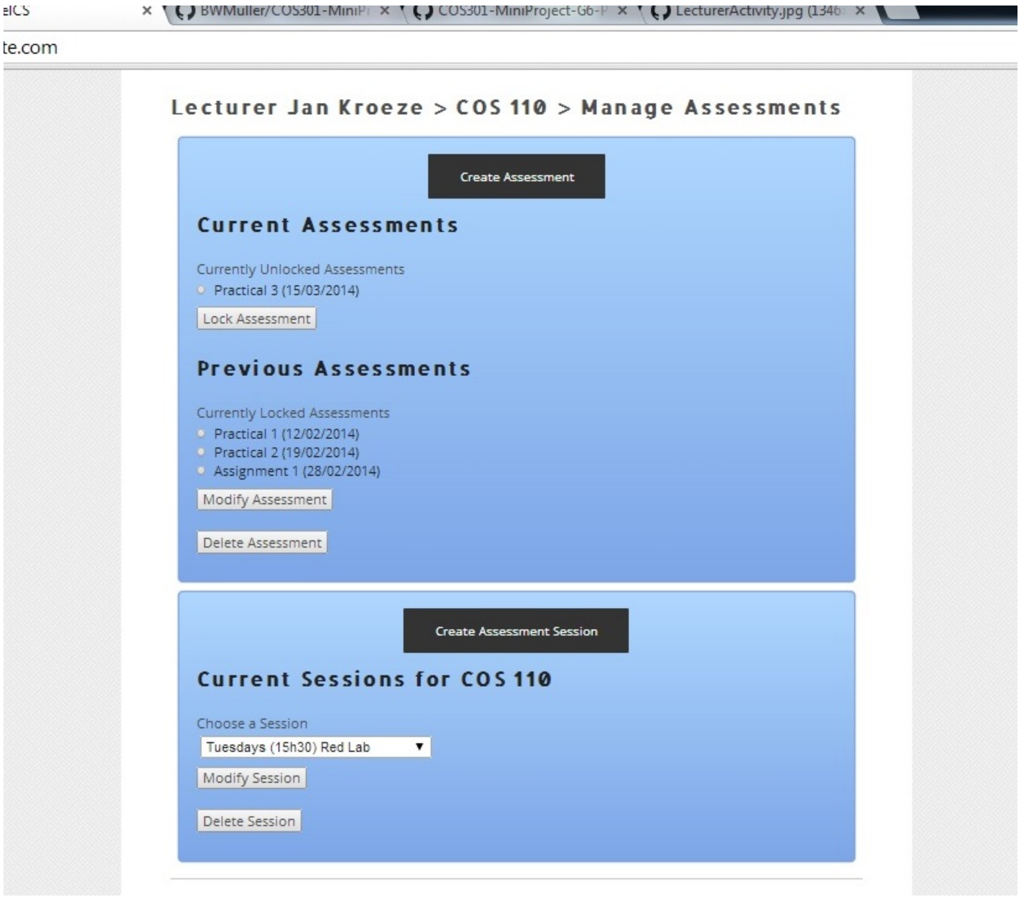


Figure 7: If a lecturer logs in, he/she chooses a module from a list of modules that they are teaching.



Lecturer Jan Kroeze > COS 110 > Manage Assessments

Current Assessments

Currently Unlocked Assessments
- Practical 3 (15/03/2014)

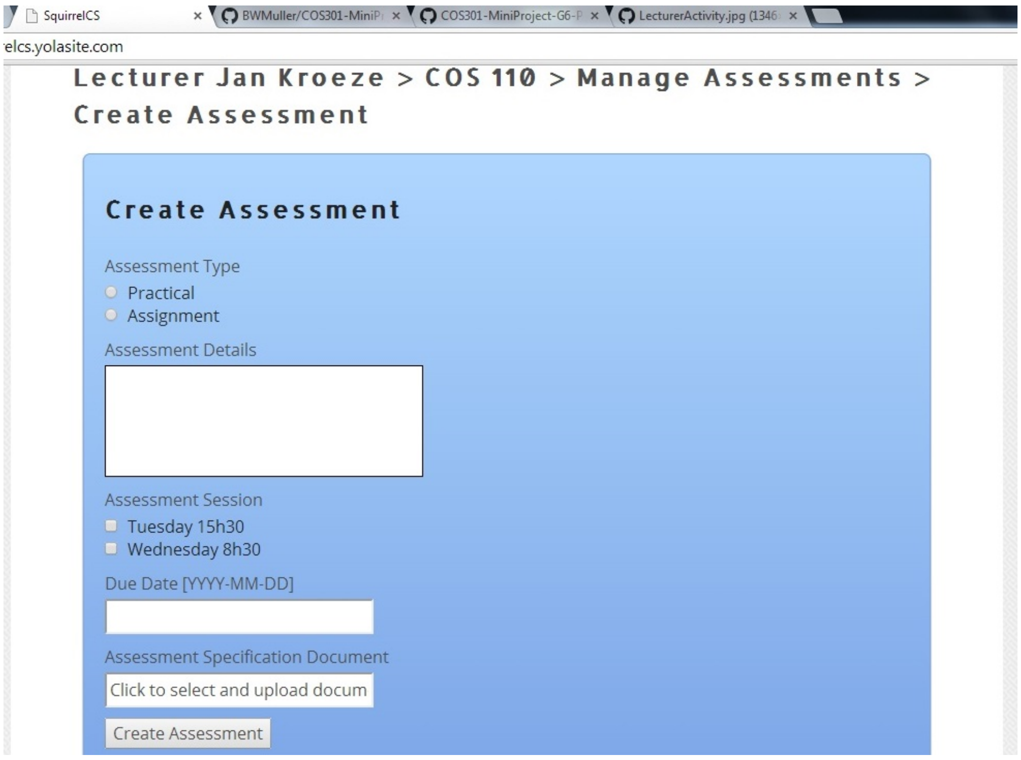
Previous Assessments

Currently Locked Assessments
- Practical 1 (12/02/2014)
- Practical 2 (19/02/2014)
- Assignment 1 (28/02/2014)

Current Sessions for COS 110

Choose a Session: Tuesdays (15h30) Red Lab

Figure 8: The lecturer then can choose to manage an assessment for that module.


SquirrelCS | BWMuller/COS301-MiniP | COS301-MiniProject-G6-P | LecturerActivity.jpg (1346

elcs.yolasite.com

Lecturer Jan Kroeze > COS 110 > Manage Assessments > Create Assessment

Create Assessment

Assessment Type
- Practical
- Assignment

Assessment Details

Assessment Session
- Tuesday 15h30
- Wednesday 8h30

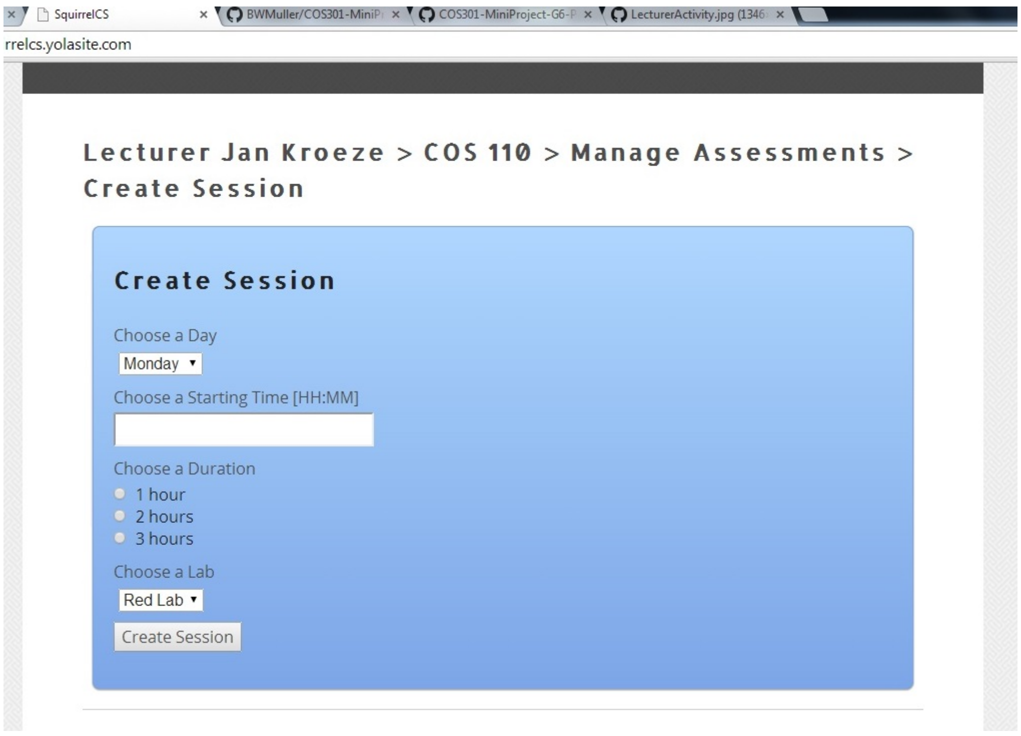
Due Date [YYYY-MM-DD]

Assessment Specification Document

Click to select and upload docum

Create Assessment

Figure 8: A lecturer can create or modify an assessment.


SquirrelCS | BWMuller/COS301-MiniP | COS301-MiniProject-G6-P | LecturerActivity.jpg (1346

rrelcs.yolasite.com

Lecturer Jan Kroeze > COS 110 > Manage Assessments > Create Session

Create Session

Choose a Day

Monday

Choose a Starting Time [HH:MM]

Choose a Duration
- 1 hour
- 2 hours
- 3 hours

Choose a Lab

Red Lab

Create Session

Figure 9: A lecturer may also create or modify a marking session.

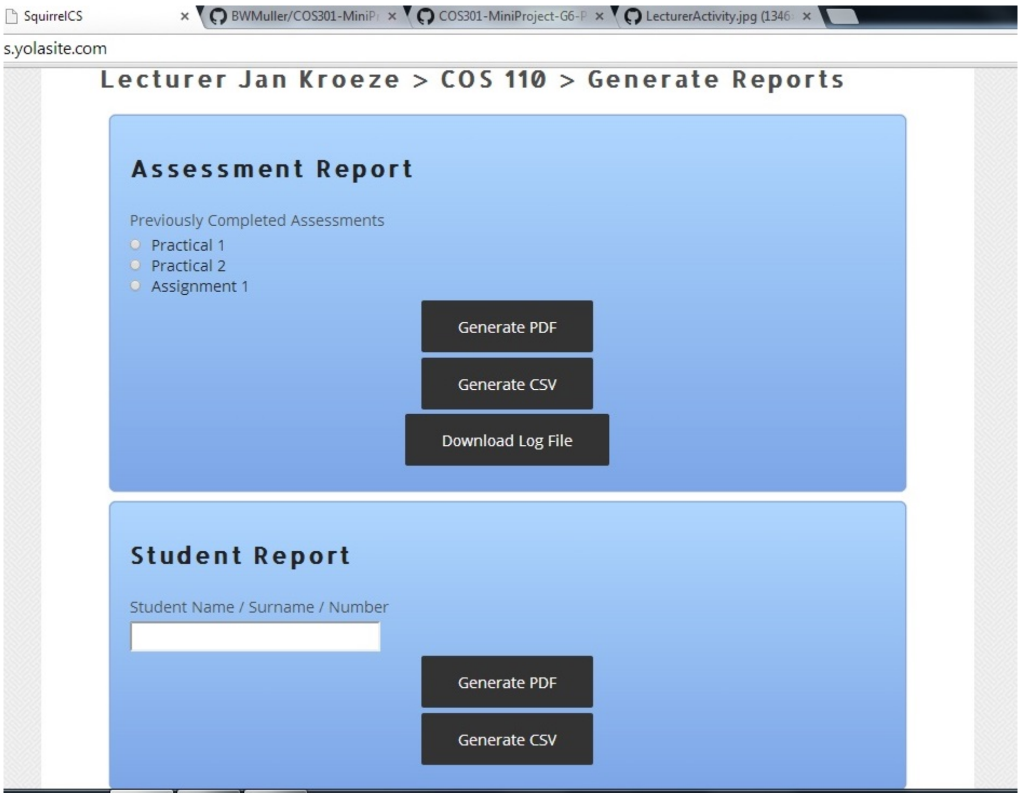


Figure 10: The lecturer can also download reports on assessments or individual student marks as PDF or CSV files.

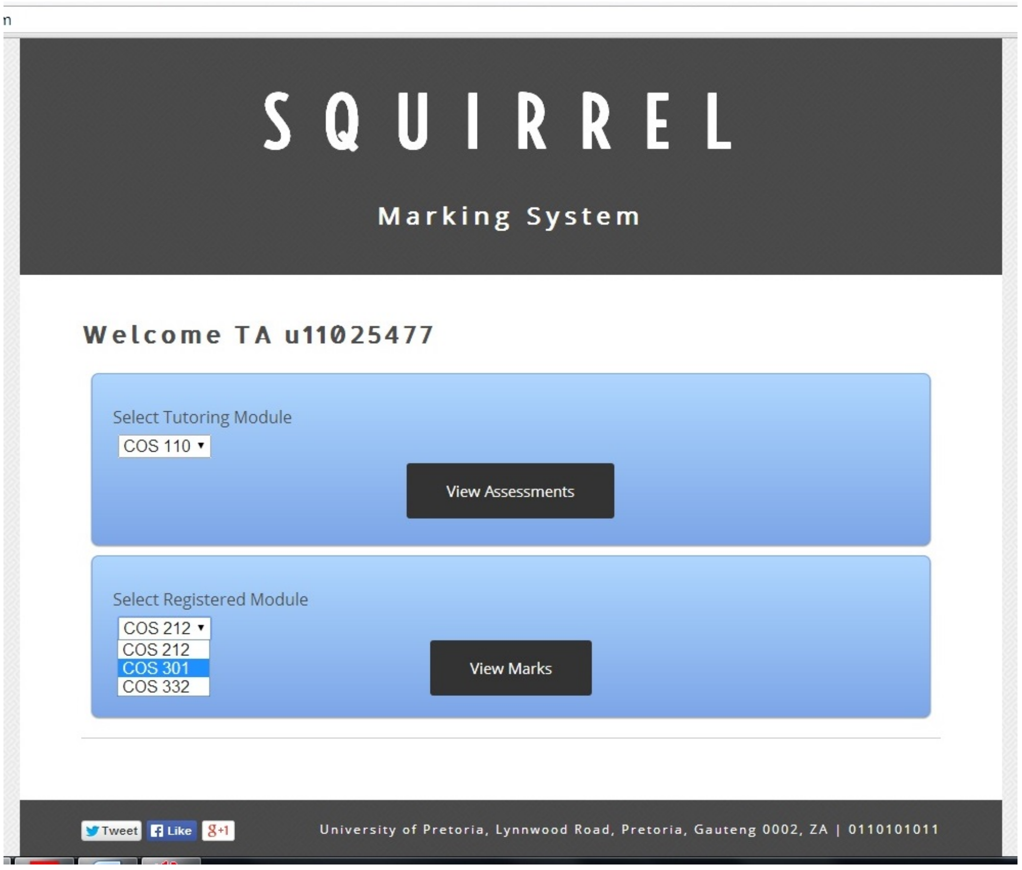
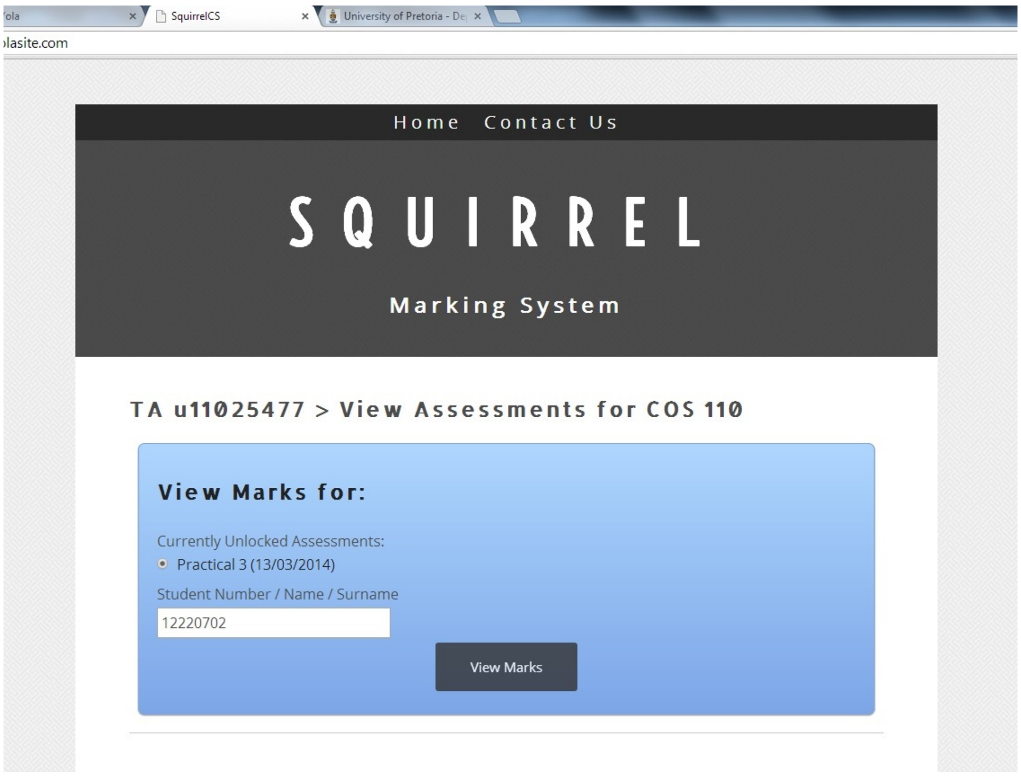


Figure 11: When a TA logs in , they either choose a module for which they are currently registered, or a module they are a TA for.



Home Contact Us

SQUIRREL

Marking System

TA u11025477 > View Assessments for COS 110

View Marks for:

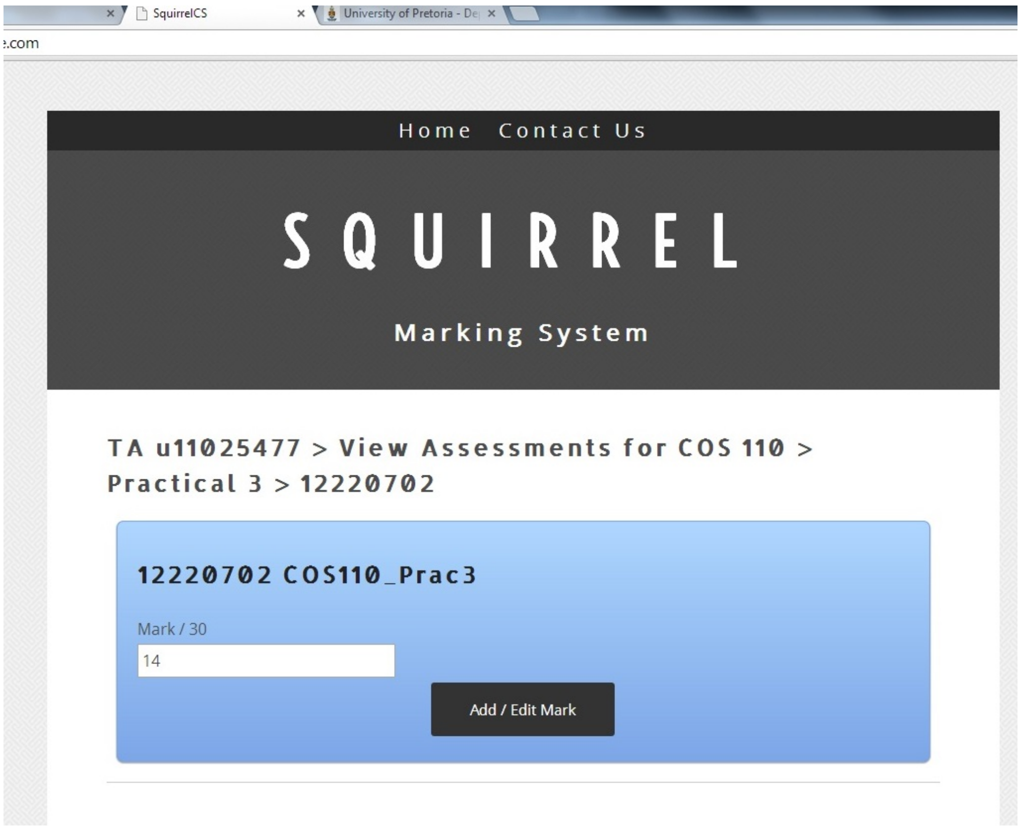
Currently Unlocked Assessments:

● Practical 3 (13/03/2014)

Student Number / Name / Surname

12220702

Figure 12: They can then select the unlocked assessment they are interested in.



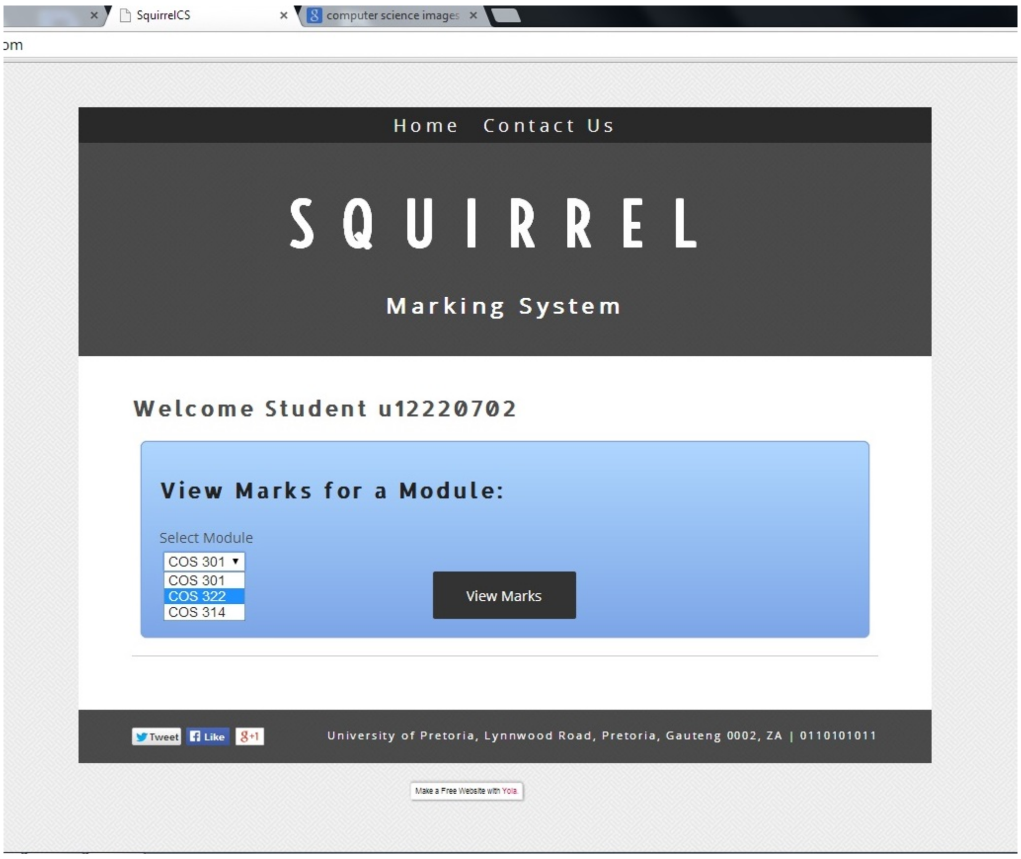
TA u11025477 > View Assessments for COS 110 > Practical 3 > 12220702

12220702 COS110_Prac3

Mark / 30

14

Figure 13: And then select a student and edit his/her marks.



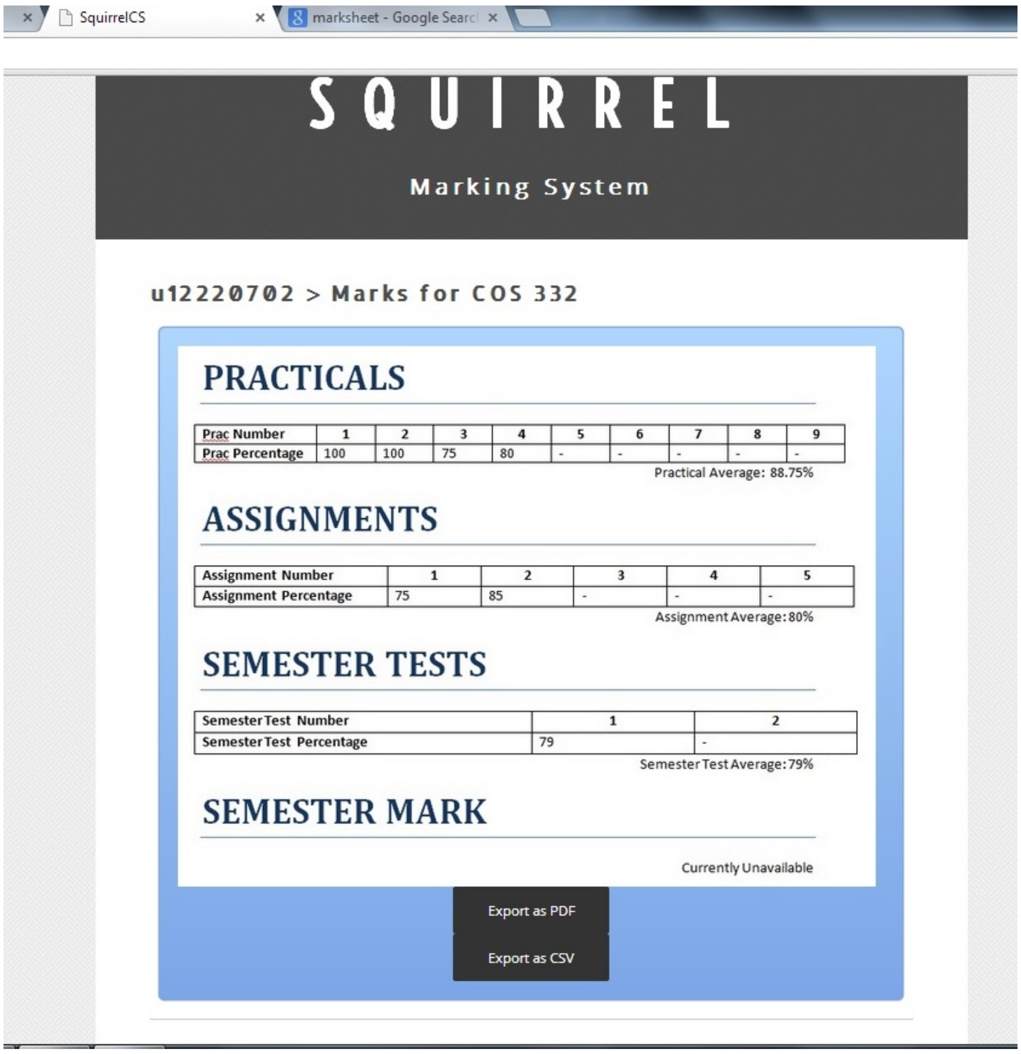
Welcome Student u12220702

View Marks for a Module:

Select Module
COS 301
COS 301
COS 322
COS 314

University of Pretoria, Lynnwood Road, Pretoria, Gauteng 0002, ZA | 0110101011

Figure 14: When a student logs in , they can only select a module they are registered for.



u12220702 > Marks for COS 332

PRACTICALS

Prac Number	1	2	3	4	5	6	7	8	9
Prac Percentage	100	100	75	80	-	-	-	-	-

Practical Average: 88.75%

ASSIGNMENTS

Assignment Number	1	2	3	4	5
Assignment Percentage	75	85	-	-	-

Assignment Average: 80%

SEMESTER TESTS

SemesterTest Number	1	2
SemesterTest Percentage	79	-

Semester Test Average: 79%

SEMESTER MARK

Currently Unavailable

Figure 15: And then view all the current marks for that module once the lecturer has released them.

UML Diagrams

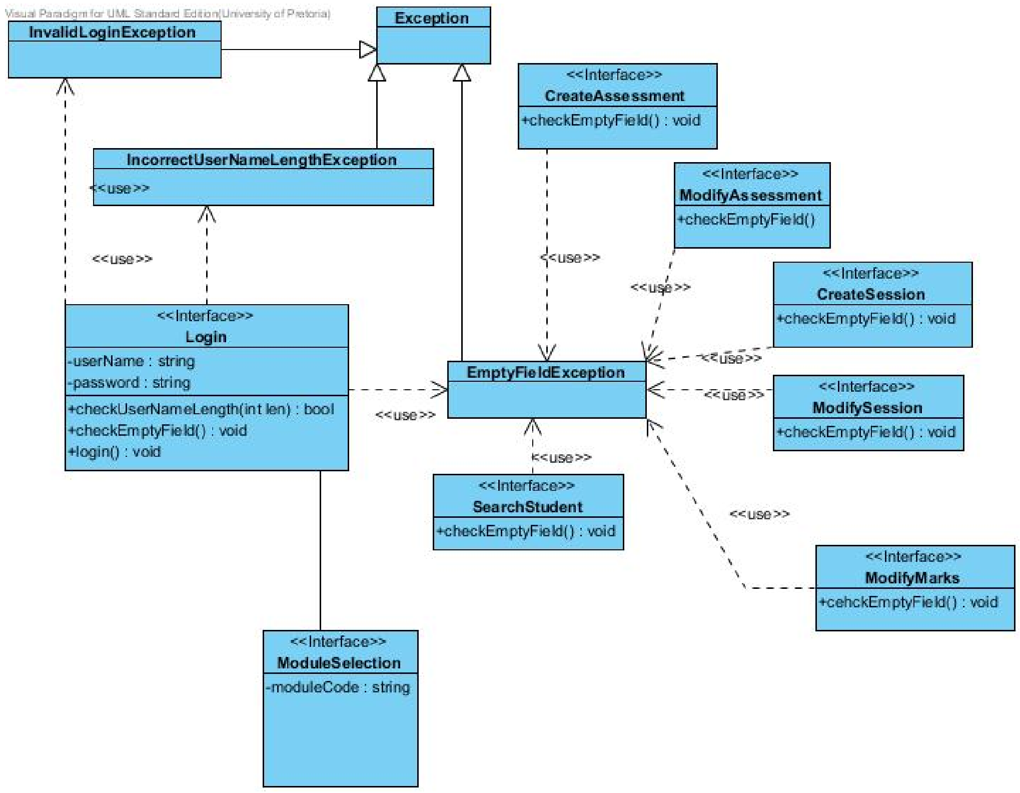


Figure 16: All possible exceptions for an empty field or student number of invalid length.

5.3 Android Application

Detailed System-Process Specifications for Android

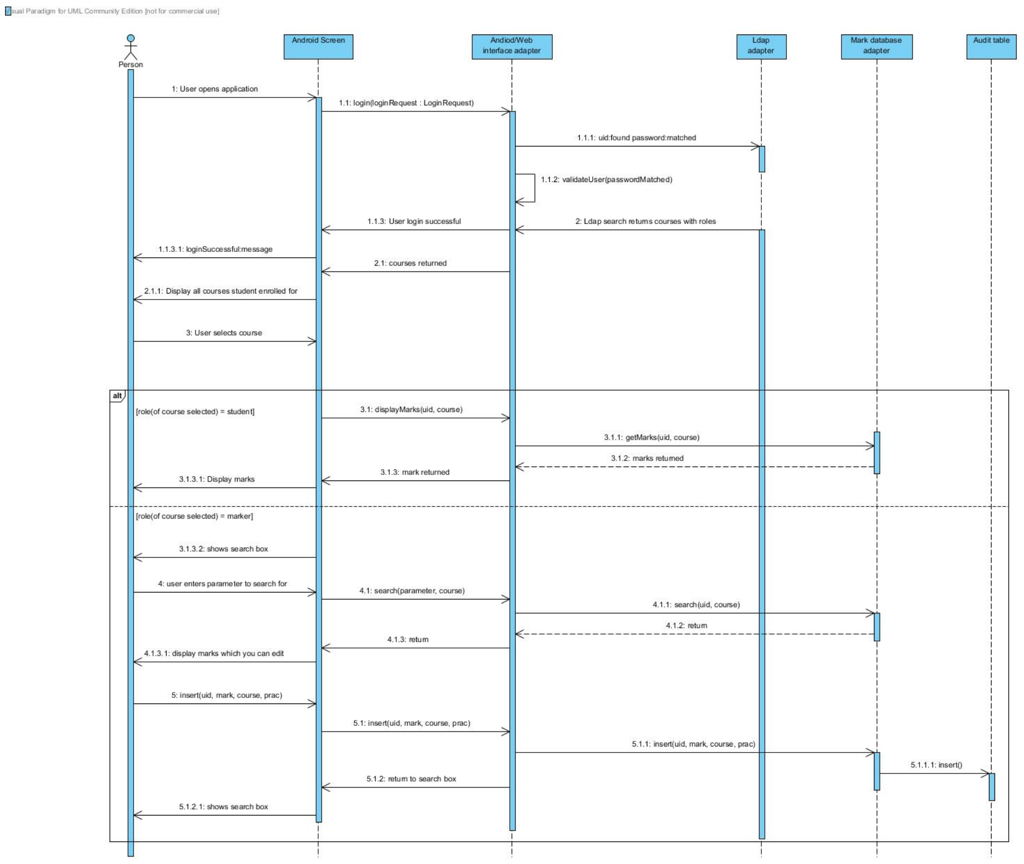

Visual Paradigm for UML Community Edition [not for commercial use]
Person
Android Screen
Andiod/Web interface adapter
Ldap adapter
Mark database adapter
Audit table
1: User opens application
1.1: login(loginRequest : LoginRequest)
1.1.1: uid:found password:matched
1.1.2: validateUser(passwordMatched)
1.1.3: User login successful
2: Ldap search returns courses with roles
1.1.3.1: loginSuccessful:message
2.1: courses returned
2.1.1: Display all courses student enrolled for
3: User selects course
alt
[role(of course selected) = student]
3.1: displayMarks(uid, course)
3.1.1: getMarks(uid, course)
3.1.2: marks returned
3.1.3: mark returned
3.1.3.1: Display marks
[role(of course selected) = marker]
3.1.3.2: shows search box
4: user enters parameter to search for
4.1: search(parameter, course)
4.1.1: search(uid, course)
4.1.2: return
4.1.3: return
4.1.3.1: display marks which you can edit
5: insert(uid, mark, course, prac)
5.1: insert(uid, mark, course, prac)
5.1.1: insert(uid, mark, course, prac)
5.1.1.1: insert()
5.1.2: return to search box
5.1.2.1: shows search box

Figure 17: Android App Sequence Diagram.

This sequence diagram covers the system-process specifications for android application. When the user opens the application the user will be prompted with a login screen. The user will login. The details he entered will be sent to interface(to let the android application and web interface connect an adapter/interface to improve flexibility, if for example ios application wants to be developed will be easier to integrate it with current system).
This interface will then communicate with ldap adapter to authenticate the user. Then personal details will be constructed displaying by course this will be provided by ldap since all that information is stored there.
For example(Appendices in master specification):
LDAP search for memberuid=u29052735 returns:

    cn: stud COS333
cn: stud COS301
cn: stud COS326
cn: stud COS216
cn: stud COS330

User can select course that the user is enrolled for or marker or lecturer of. The sequence diagram alt is for when user is a student or when user is a marker. The reason for the course selector is since users can both be markers and students on different courses. If user is a marker his marks will be fetched by the mark database adapter and then will be sent to the interface and then will be displayed on android application.
If user is marker then search box will appear where marker can search students or names. He will only be able to search for students in his assigned practical groups. Then practical's that are not currently locked that needs to be marked will be available for that student and marker can enter mark. Update must be made in audit-table when any mark is changed.

UI Screen Designs and Work-Flow Specifications

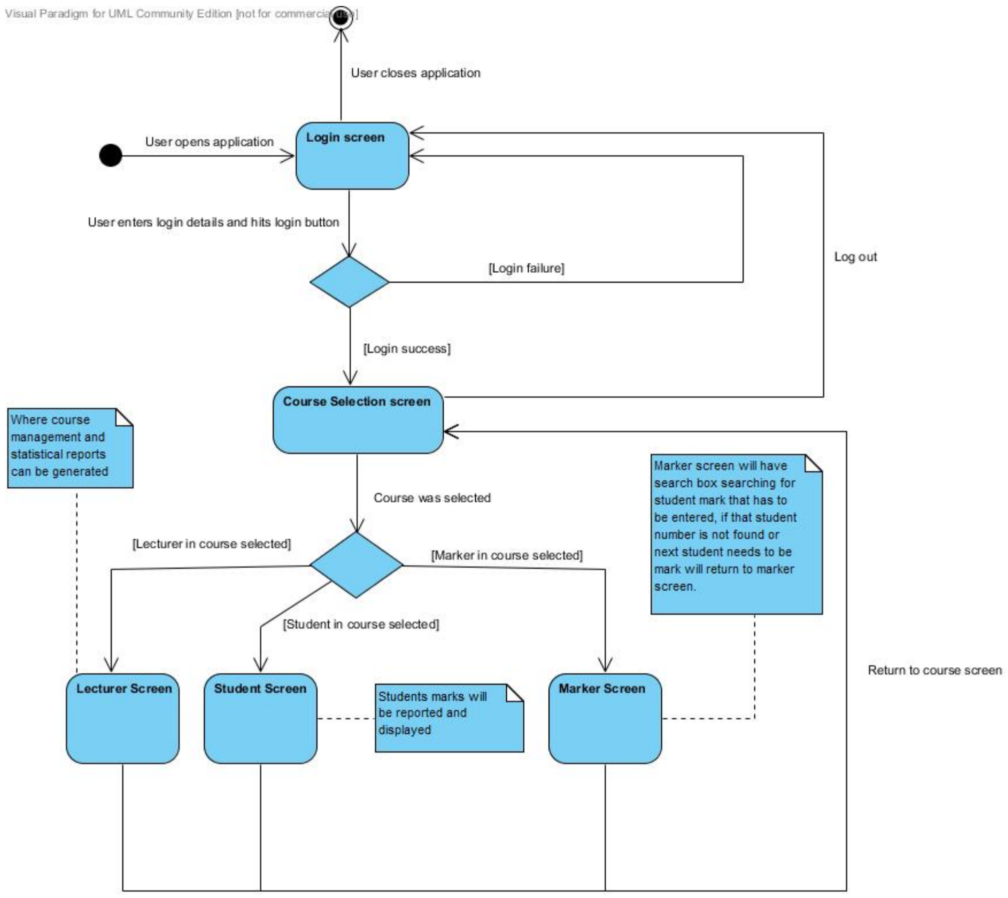

Visual Paradigm for UML Community Edition [not for commercial use]
User closes application
User opens application
Login screen
User enters login details and hits login button
[Login failure]
Log out
[Login success]
Course Selection screen
Where course management and statistical reports can be generated
Course was selected
Marker screen will have search box searching for student mark that has to be entered, if that student number is not found or next student needs to be mark will return to marker screen.
[Lecturer in course selected]
[Marker in course selected]
[Student in course selected]
Return to course screen
Lecturer Screen
Student Screen
Students marks will be reported and displayed
Marker Screen

Figure 18: Android App Activity Diagram.

This is activity diagram of the flow of how the user interface of the android application should work. The rest of the android application layout will not be specified in detail like where the textboxes should be located since we want to keep the user interface flexible for implementation phase and don't want to tie down and limit the implementation.

Lower Levels of Granularity

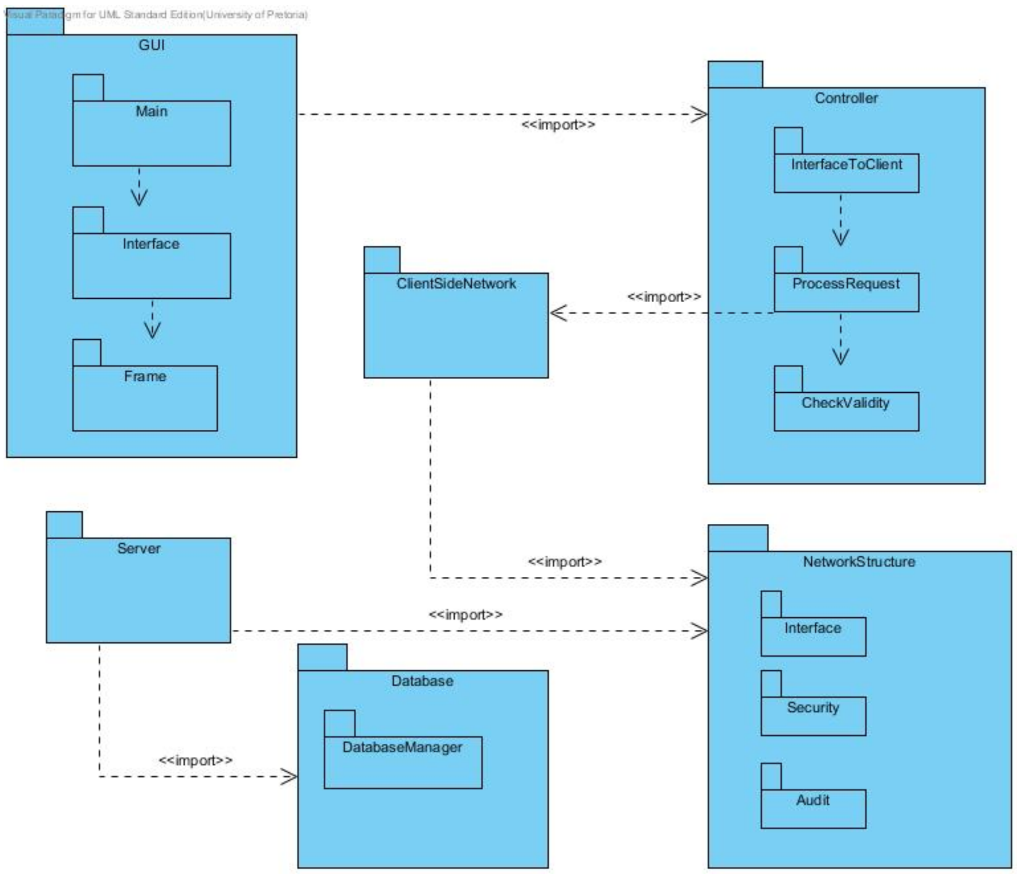
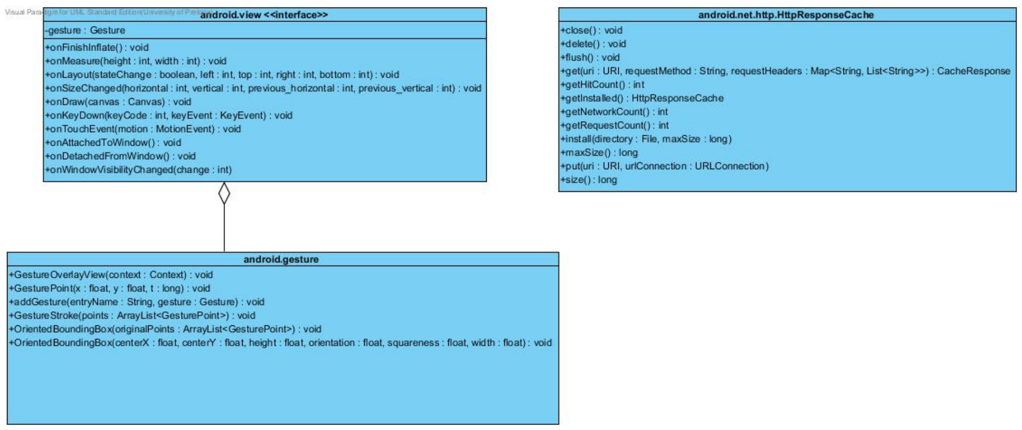


Figure 19: Android App Lower Levels of Granularity.

API Specifications


Visual Paradigm for UML Standard Edition(University of Pretoria)

```
android.view <<interface>>
-gesture : Gesture
+onFinishInflate() : void
+onMeasure(height : int, width : int) : void
+onLayout(stateChange : boolean, left : int, top : int, right : int, bottom : int) : void
+onSizeChanged(horizontal : int, vertical : int, previous_horizontal : int, previous_vertical : int) : void
+onDraw(canvas : Canvas) : void
+onKeyDown(keyCode : int, keyEvent : KeyEvent) : void
+onTouchEvent(motion : MotionEvent) : void
+onAttachedToWindow() : void
+onDetachedFromWindow() : void
+onWindowVisibilityChanged(change : int)
```

```
android.net.http.HttpResponseCache
+close() : void
+delete() : void
+flush() : void
+get(uri : URI, requestMethod : String, requestHeaders : Map<String, List<String>>) : CacheResponse
+getHitCount() : int
+getInstalled() : HttpResponseCache
+getNetworkCount() : int
+getRequestCount() : int
+install(directory : File, maxSize : long)
+maxSize() : long
+put(uri : URI, urlConnection : URLConnection)
+size() : long
```

```
android.gesture
+GestureOverlayView(context : Context) : void
+GesturePoint(x : float, y : float, t : long) : void
+addGesture(entryName : String, gesture : Gesture) : void
+GestureStroke(points : ArrayList<GesturePoint>) : void
+OrientedBoundingBox(originalPoints : ArrayList<GesturePoint>) : void
+OrientedBoundingBox(centerX : float, centerY : float, height : float, orientation : float, squareness : float, width : float) : void
```

Figure 20: Android App API.

Class Diagrams

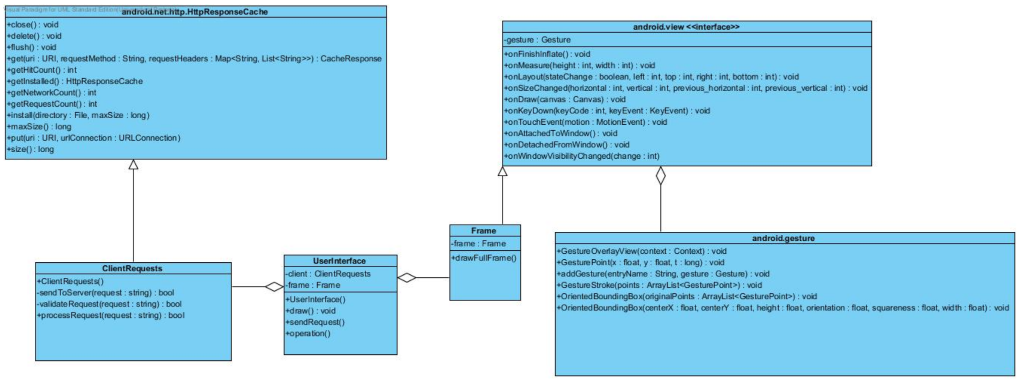

Visual Paradigm for UML Standard Edition(University of Pretoria)
android.net.http.HttpResponseCache
+close() : void
+delete() : void
+flush() : void
+get(uri : URI, requestMethod : String, requestHeaders : Map<String, List<String>>) : CacheResponse
+getHitCount() : int
+getInstalled() : HttpResponseCache
+getNetworkCount() : int
+getRequestCount() : int
+install(directory : File, maxSize : long)
+maxSize() : long
+put(uri : URI, urlConnection : URLConnection)
+size() : long
android.view <<interface>>
-gesture : Gesture
+onFinishInflate() : void
+onMeasure(height : int, width : int) : void
+onLayout(stateChange : boolean, left : int, top : int, right : int, bottom : int) : void
+onSizeChanged(horizontal : int, vertical : int, previous_horizontal : int, previous_vertical : int) : void
+onDraw(canvas : Canvas) : void
+onKeyDown(keyCode : int, keyEvent : KeyEvent) : void
+onTouchEvent(motion : MotionEvent) : void
+onAttachedToWindow() : void
+onDetachedFromWindow() : void
+onWindowVisibilityChanged(change : int)
ClientRequests
+ClientRequests()
-sendToServer(request : string) : bool
-validateRequest(request : string) : bool
+processRequest(request : string) : bool
UserInterface
-client : ClientRequests
-frame : Frame
+UserInterface()
+draw() : void
+sendRequest()
+operation()
Frame
-frame : Frame
+drawFullFrame()
android.gesture
+GestureOverlayView(context : Context) : void
+GesturePoint(x : float, y : float, t : long) : void
+addGesture(entryName : String, gesture : Gesture) : void
+GestureStroke(points : ArrayList<GesturePoint>) : void
+OrientedBoundingBox(originalPoints : ArrayList<GesturePoint>) : void
+OrientedBoundingBox(centerX : float, centerY : float, height : float, orientation : float, squareness : float, width : float) : void

Figure 21: Android App Class Diagrams.

Design of the Database Tables

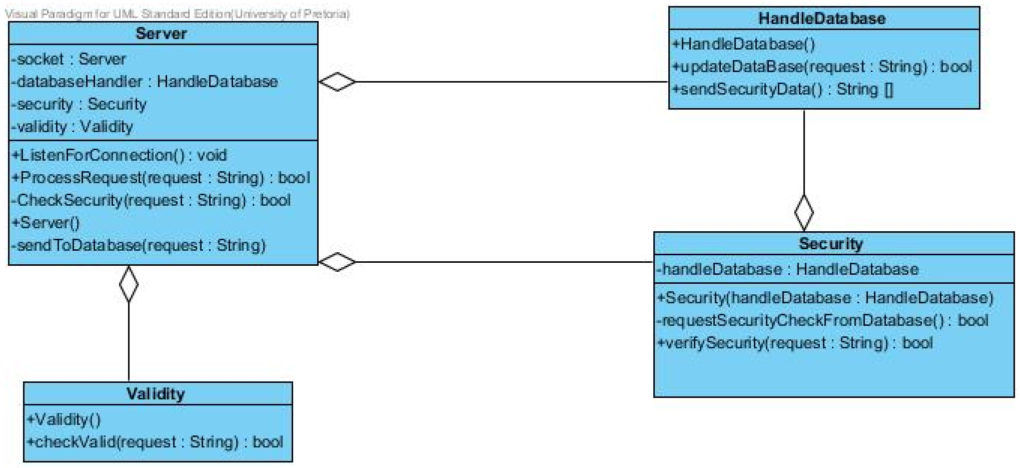

Visual Paradigm for UML Standard Edition(University of Pretoria)
Server
-socket : Server
-databaseHandler : HandleDatabase
-security : Security
-validity : Validity
+ListenForConnection() : void
+ProcessRequest(request : String) : bool
-CheckSecurity(request : String) : bool
+Server()
-sendToDatabase(request : String)
HandleDatabase
+HandleDatabase()
+updateDataBase(request : String) : bool
+sendSecurityData() : String []
Security
-handleDatabase : HandleDatabase
+Security(handleDatabase : HandleDatabase)
-requestSecurityCheckFromDatabase() : bool
+verifySecurity(request : String) : bool
Validity
+Validity()
+checkValid(request : String) : bool

Figure 22: Android App Class Diagrams for the Server.