Cos 301: Mini Project phase 2
Design document

Group 8

Group Members:

Melany Barnes (12030466)

Zane Bloom (12236722)

Mathys Ellis (12019837)

Zenadia Groenewald (12265676)

Alfred Ngako (12236731)

Gerhard Smit (12282945)

Version 1.0

Change Log

Change log			
Date	**Version**	**Description**	**Person**
11/03/2014	v 0.0	Document created	Mathys Ellis
11/03/2014	v 0.1	Added android application: UI design, Process Specifications and user work flow	Mathys Ellis
11/03/2014	v 0.2	Some database tables added	Melany Barnes
12/03/2014	v 0.3	Added android application: API. Modified Process Specifications	Mathys Ellis
12/03/2014	v 0.4	Added Protocols to be used	Zane Bloom
12/03/2014	v 0.5	Added Libraries to be used	Zane Bloom
13/03/2014	v 0.6	Added Technologies to be used	Gerhard Smit
13/03/2014	v 0.7	Added Back-end: Database tables, user work-flow	Melany Barnes
13/03/2014	v 0.8	Added Frameworks to be used	Gerhard Smit
13/03/2014	v 0.8	Added Web User Interface sections	Alfred Ngako
13/03/2014	v 0.9	Added Back-end: API, levels of granularity, System class diagrams	Zenadia Groenewald
13/03/2014	v 1.0	Finalised document	Mathys Ellis

GitHub repository: Mini project - Phase 2 - Group 8

Contents

List of Figures

1 Software architecture design

1.1 Technologies

1.1.1 Programming languages

The system will be coded in Django a high-level Python Web framework, as it is specified in the architecture constraints. Django allows the system to be built in phases, it allows different parts to be programmed and designed then merged at the end of the design.

1.1.2 Database technologies

MySQL will be used to code the database as it is part of the architecture constraints. Also the MySQL database is scalable as it can handle over fifty million rows of data at time. It has a high level of security which allows us to specify which user has privileges, as well as encrypting the passwords of each user. It is cost effective as the University is already running the Novel network on their infrastructure which includes the MySQL database package.

1.1.3 Application servers

The application server chosen will be the Apache server, as it is part of the architecture constraints. The Apache server is already installed on the University's server. Apache server is open source allowing modification if required, it is cost effective due to the nature of being open source, it supports Python which is what the system is to be built on. It is portable allowing it to be installed on any system.

1.1.4 Testing modules

The system will be tested using the Django unittest module which comes with the Django package it is part of the architecture constraints. The unittest module allows the sharing and integration of code between seperate tests such as the start up and shutdown of testing. It allows Command-Line interface as well as code imbedded testing. Unittest has functionality to re-use old test code instead of designing the test code over.

1.2 Frameworks

1.2.1 Object-Relational Mapper

The Django object-relational mapper allows us to integrate between the objects created with the users data, such as marks, names, surnames and so forth, to simple SQL statements to be stored in the MySQL database that will be used. Django allows multiple inheritance allwoing easier implementation and structure of objects. The object-relational mapper from Django is an architecture constraint.

1.2.2 Web services frameworks

The system will make use of Django-REST-framework as it allows easy and fast solutions to making REST APIs, it has serialization that is usable by both ORM and non-ORM data sources, which is useful for us as we require ORM sources to be used and integrated in the system. It has OAuth 1.0 and OAuth 2.0 integrated making the system easier to be accessed via client side to the server side through any platform.

1.3 Protocols

1.3.1 HTTP - HyperText Transfer Protocol

The system will make use of HTTP to receive requests and send responses through the different access channels. This protocol will define the communication between the sytem and web browsers as well the Android application.

1.3.2 HTTPS - HyperText Transfer Protocol Secure

The system will make use of HTTPS when communicating sensitive data through the different access channels.

1.3.3 REST - Representational State Transfer

Strictly speaking REST is not a protocol, but it makes use of HTTP to provide web services. REST is preferred over SOAP as it is lightweight, easy to use and will provide better performance in the case of this system. These advantages have made REST the chosen web services protocol. Making use of RESTful web services is an architectural constraint.

1.3.4 JSON - JavaScript Object Notation

The system will make use of the JSON protocol to marshal response objects that are requested by other systems. JSON is the preferred marshaling protocol due to it being efficient and lightweight. When compared to XML, the JSON marshaled objects are smaller in size and the time taken to process them is less. These performace gains have made JSON the chosen marshaling protocol.

1.3.5 LDAP - Lightweight Directory Access Protocol

The system will make use of the LDAP system used by the Department of Computer Science. This system contains the details of the users of the system. Thus the users will be authenticated through this LDAP system. LDAP is both efficient and low cost. Making use of the LDAP protocol is an architectural constraint.

1.4 Libraries

1.4.1 JSON Marshaling

The system will make use of the serialization library provided by Django. The term "serialize" is considered to be synonymous with "marshal" in the Python standard library. Therefore this library is suited for marshaling. This library supports many different formats but specifically supports JSON. This functionality is needed to by the web services.

1.4.2 LDAP Authentication Integration

The Django framework provides its own authentication system. To integrate LDAP with the Django framework, a new authentication backend must be written and added to the Django framework. This backend must realize the contract setup by the Django framework. This functionality is needed to authenticate users.

1.4.3 Importing CSV

The system will be making use of the django-csv-importer v0.1.3.5 library. This library provides the functionality to transform the data of a CSV file into Django model instance. This functionality is needed to import assessment entries from CSV files.

1.4.4 Exporting CSV

The system will be making use of the Python CSV library. This library provides the functionality to data to a CSV file. This functionality is needed to export mark sheets to CSV files.

1.4.5 Generating PDF's

The system will be making use of the ReportLab library. This library provides the functionality to dynamically generate reports in the PDF format. This functionality is needed to generate reports requested by the lecturer.

2 Application design

2.1 Back-end system

2.1.1 Lower Levels of Granularity

The levels of granularity are specific according to the various aspects of the program, meaning the different aspects are ordered by a hierarchy in the system. This order, from the lowest level to the highest, is as follows:

Per assessment:
There are many different assessments of each type that follow a single course, and there are several students who partake in each assessment.

Per assessment type:
While there are less assessment types such as tests, practicals, tutorials, exams, etc., it still follows that this aspect is highly granular.

Per semester/exam mark:
This encompasses the individual marks of students and the results they achieved throughout the semester in relation to their exam mark.

Per overall mark:
This determines the student's academic standing in the course; it consists of their various assessment marks as well as their results for examinations.

Per course:
This encompasses all students administered for that course as well as their assessment marks and average for that specific course.

These levels need to be considered in the subdivision of the system and accommodated in the program.

The application can be further subdivided by its architectural components:

The interface:
To provide an interface via an API to the system.

API:
To facilitate the interactions between the user and the system.

Client-Server communication:
Also provided by the API.

Back-end:

Receives information such as user details, performs database operations, validates information, collects data relevant to queries and commands given by the user through the API.

2.1.2 API specifications

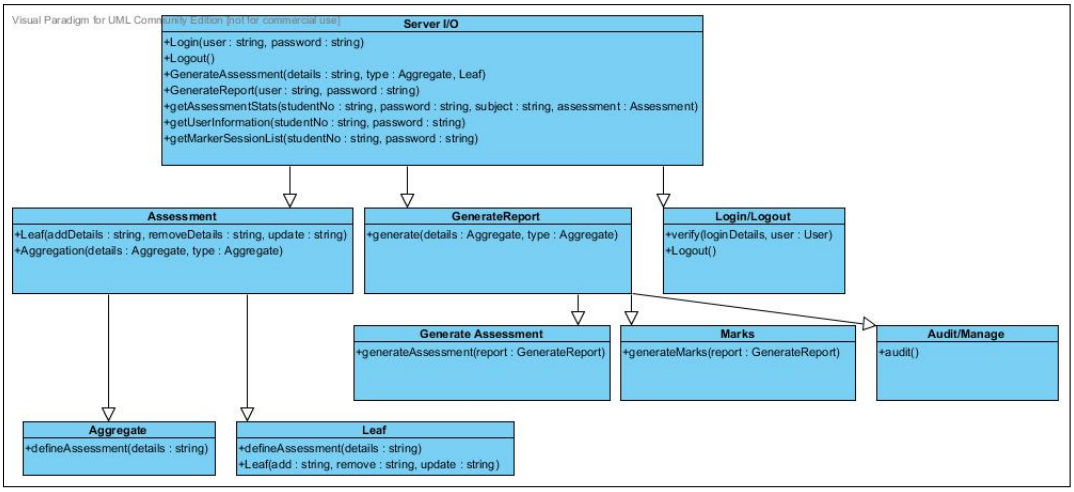

Figure 1: Class Diagram

2.1.3 System Class diagram

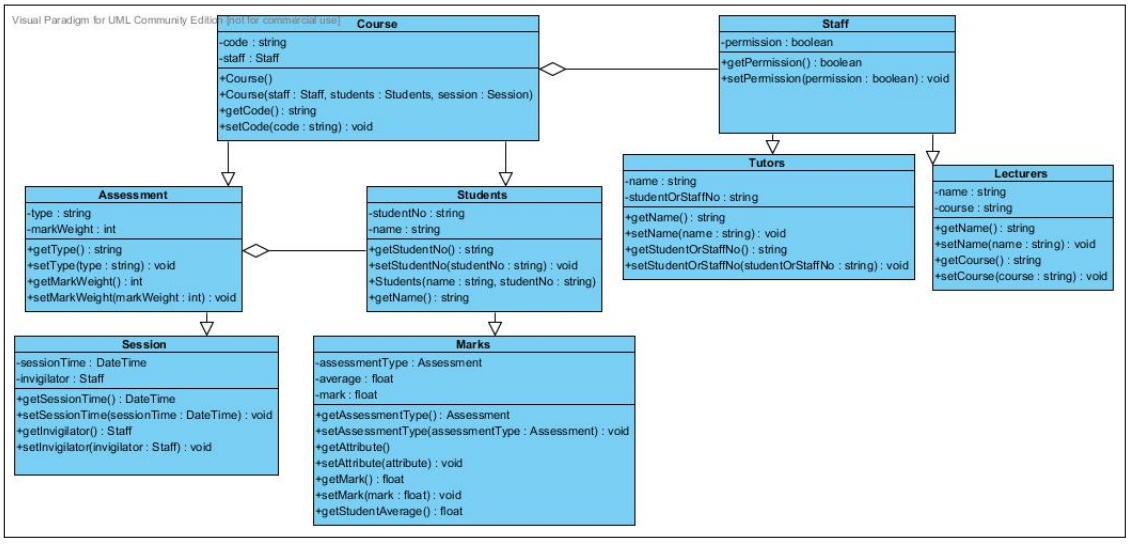

Figure 2: System Class diagram

2.1.4 Work-flow

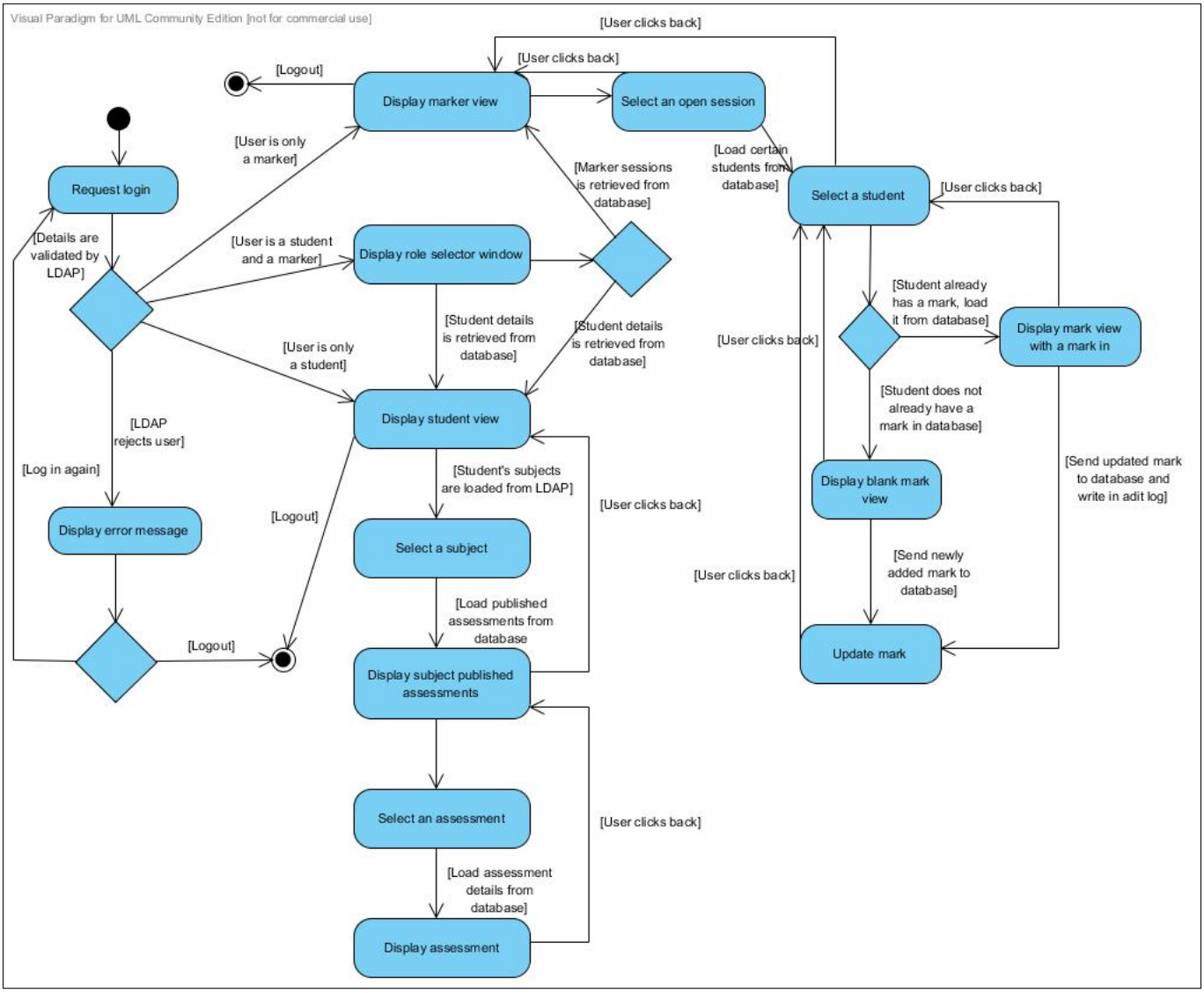

Figure 3: Activity diagram for user work-flow specifications

2.1.5 Database tables

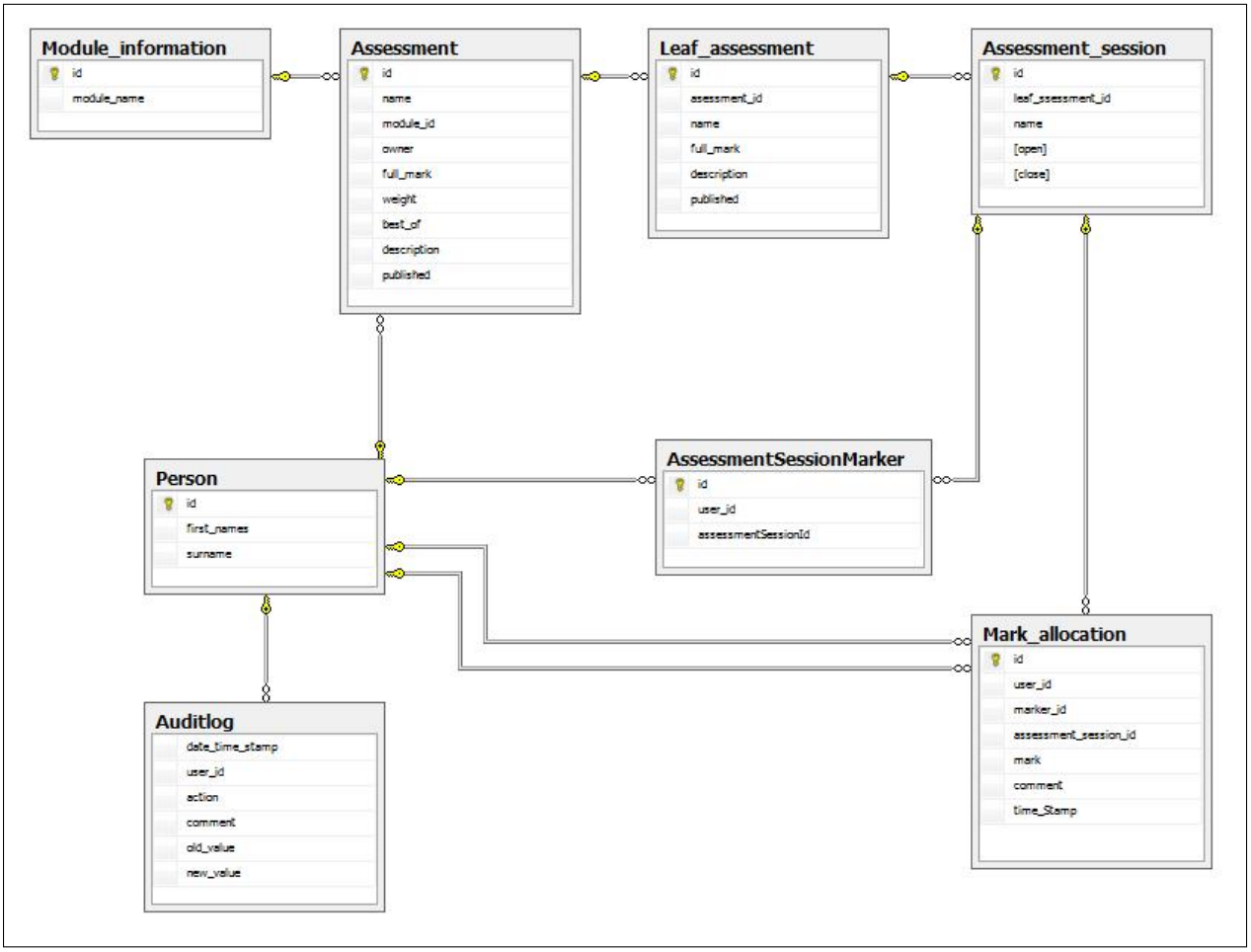

Figure 4: Database Diagram

Module_Information					
Field	Type	Null	Key	Default	Extra
id	int(11)	No	Pri	NULL	auto_increment
name	varchar(255)	No			

Person					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
first_names	varchar(255)	Yes		NULL	
surname	varchar(255)	No			

Person table: This table will store only the details of a user that will be used for searching because it is a fast reference, as these and other details about a user is already stored in the LDAP system.

Auditlog					
Field	Type	Null	Key	Default	Extra
date_time_stamp	datetime	No		0000_00_00 00:00:00	
id	int(8)	No	Foreign		References Person
action	varchar(20)	No			See reference 1
description	varchar(100)	Yes		NULL	
old_value	varchar(100)	Yes		NULL	
new_value	varchar(100)	No			

Reference 1: Action may have one of the following values: Assessment Created, Assessment Modified, Assessment Removed, Mark Submitted, Mark Modified, Mark Removed, Open Assessment, Close Assessment, Publish Marks, Assessment Report, Students Marks Report, Audit Report. Auditlog table: This table will store all changes made to the database except for a mark entered for the first time. The old value can be any type of value for example a session closing datetime which is changed or a mark which is updated, the new value will specify the value which it was changed to.

Assessment					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
name	varchar(100)	No			
module_id	int(8)	No	Foreign		References Module_information
owner	int(8)	No	Foreign		References Person
weight	int(3)	Yes		100	
best_of	int(5)	Yes		99999	
description	varchar(255)	Yes		NULL	
published	bit	No		False	

Assessment and Leaf_assessment table: For example Practical, Tutorial, Semester Test or Exam will be assessments, leaf assessments will be for example Practical 1, Practical 2, Tutorial 1, Tutorial 2, Semester Test 1, Semester Test 2.

Assessment_session table: This table will contain all the information needed per session of an assessment.

Assessment_Session_Marker table: This table will specify which marker is associated with which assessment session.

Leaf_assessment					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
name	varchar(100)	No			
assessment_id	int(8)	No	Foreign		References Assessment
full_mark	int(3)	No			
description	varchar(255)	Yes	NULL		
published	bit	No		False	

Assessment_session					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
leaf_assessment_id	int(8)	No	Foreign		References Leaf_assessment
name	varchar(100)	No			
open	datetime	No		Date/time now	
close	datetime	No		Date/time now	

Mark_allocation table: This table will store all the marks obtained per student per assessment session marked.

Assessment_Session_Marker					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
user_id	int(8)	No	Foreign		References Person
assessment_session_id	int(8)	No	Foreign		References Assessment_session

Mark_allocation					
Field	Type	Null	Key	Default	Extra
id	int(8)	No	Pri	NULL	auto_increment
user_id	int(8)	No	Foreign		References Person
marker_id	int(8)	No	Foreign		References Person
assessment_session_id	int(8)	No	Foreign		References Assessment_session
mark	int(3)	No			
comment	varchar(255)	Yes			
time_stamp	datetime	No			

2.2 Web interface

2.2.1 UI work flow

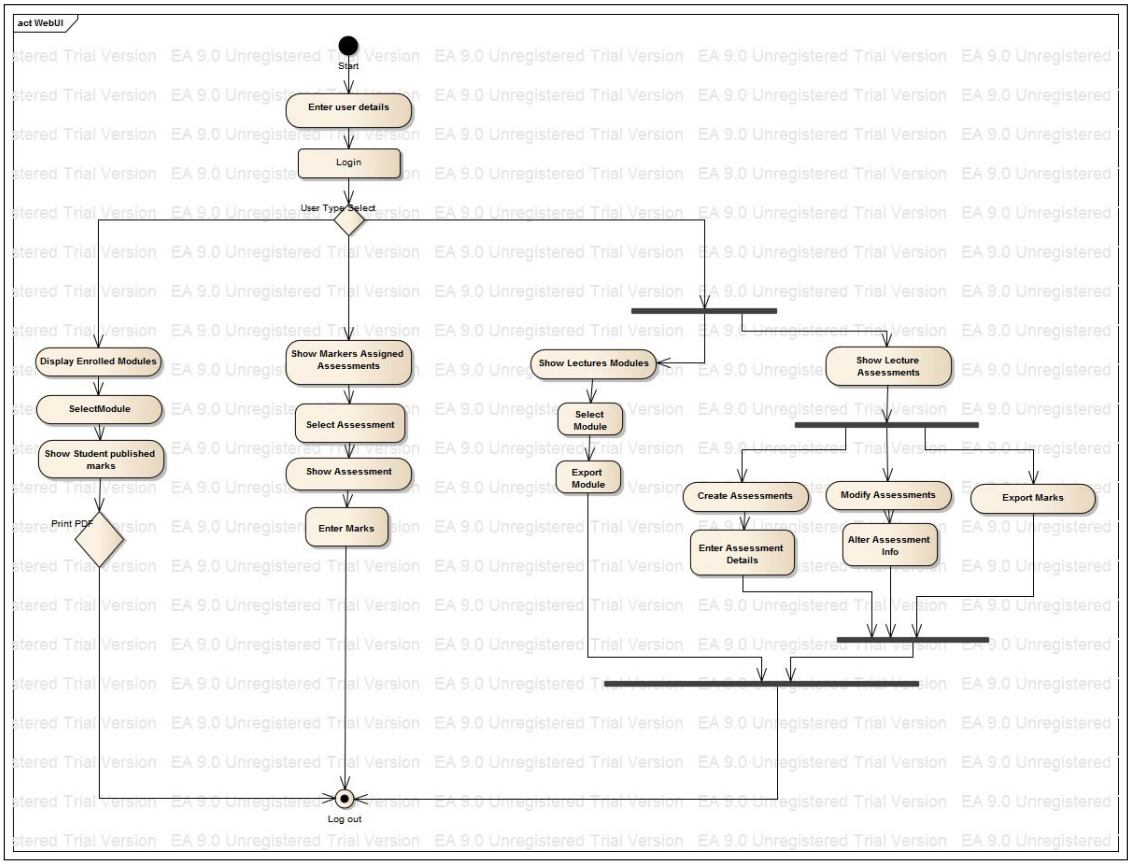

Figure 5: Web UI user work flow

2.2.2 UI Screen Design

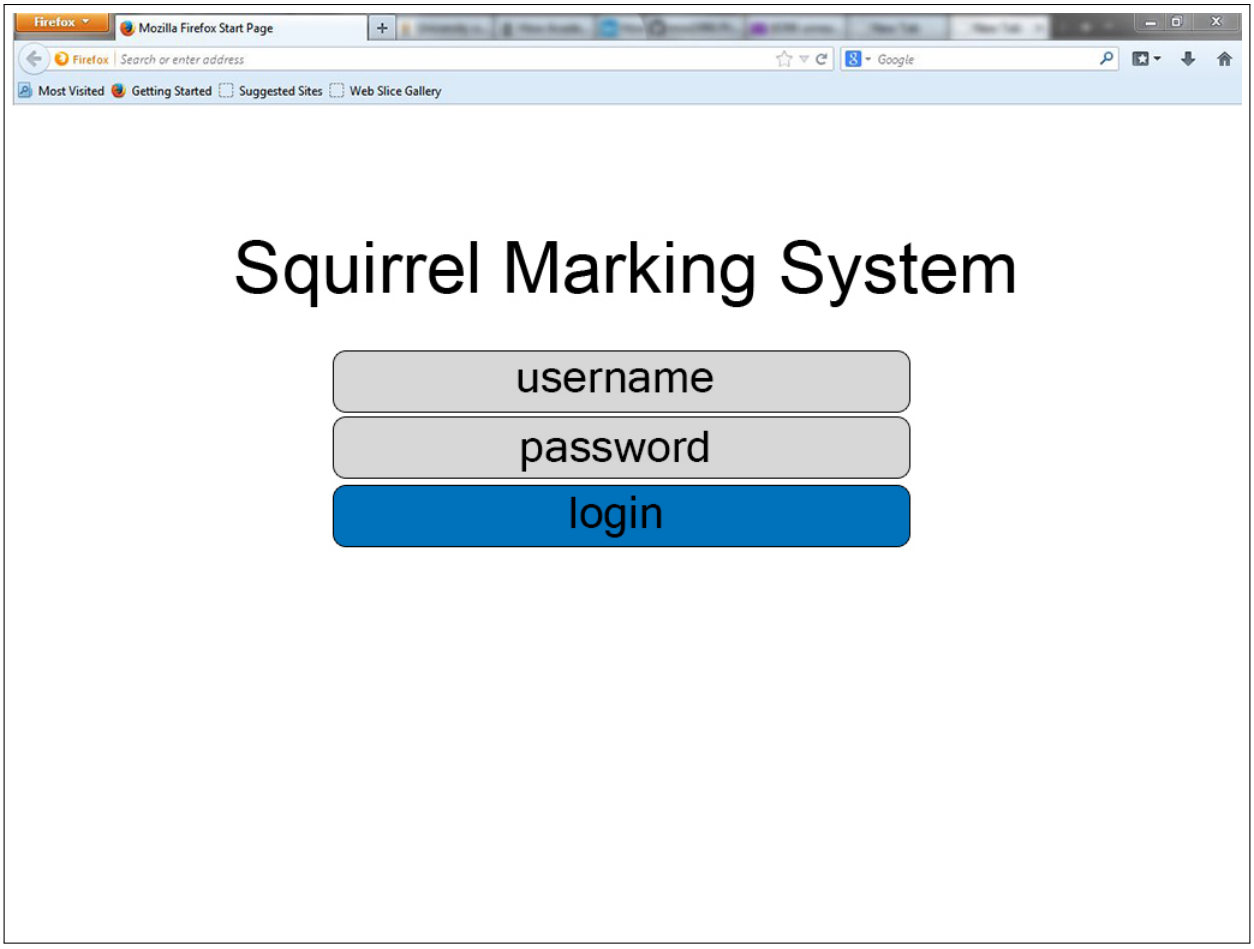

Figure 6: Web UI screen design: Login Screen

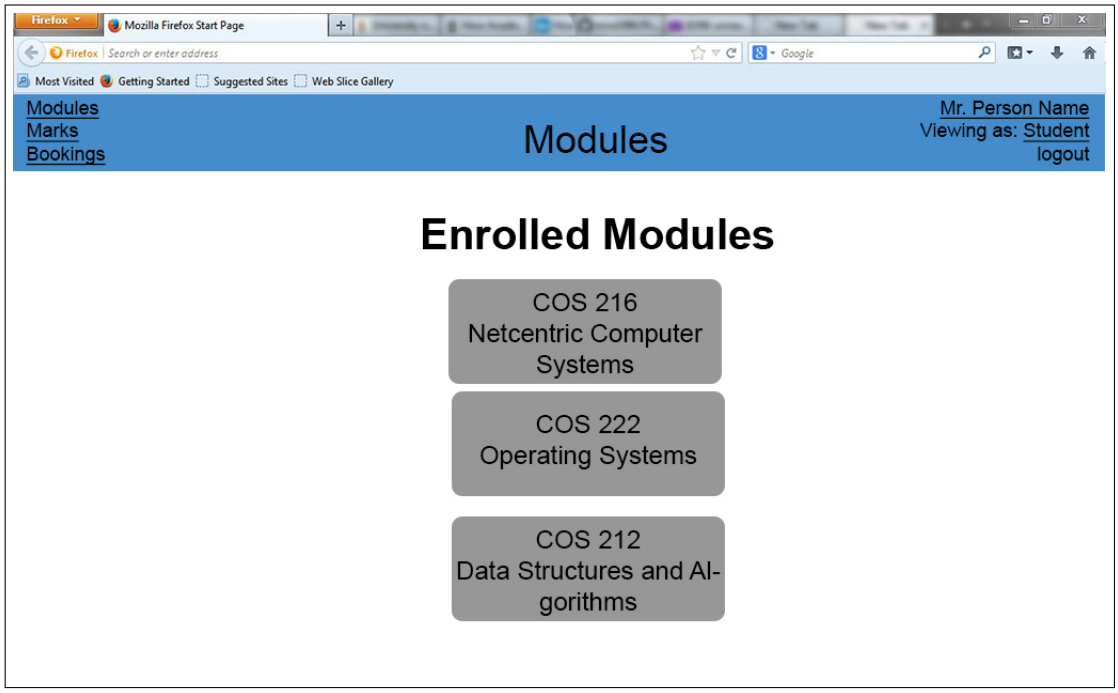

Figure 7: Web UI screen design: Student Landing Page

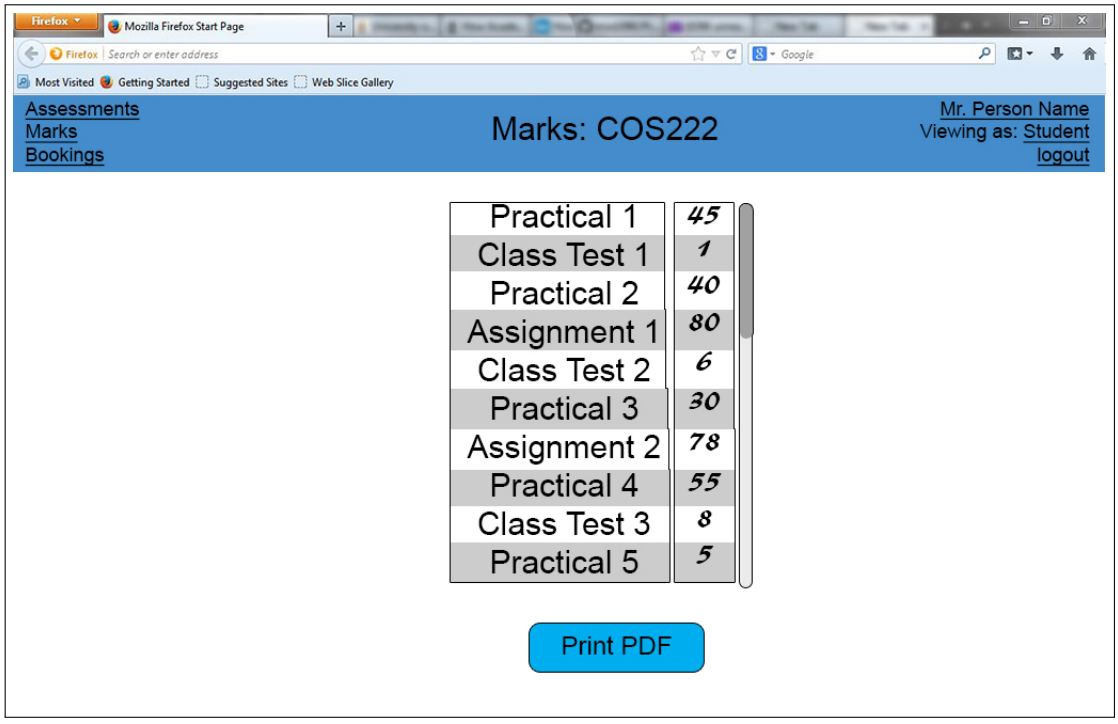

Figure 8: Web UI screen design: Student Marks

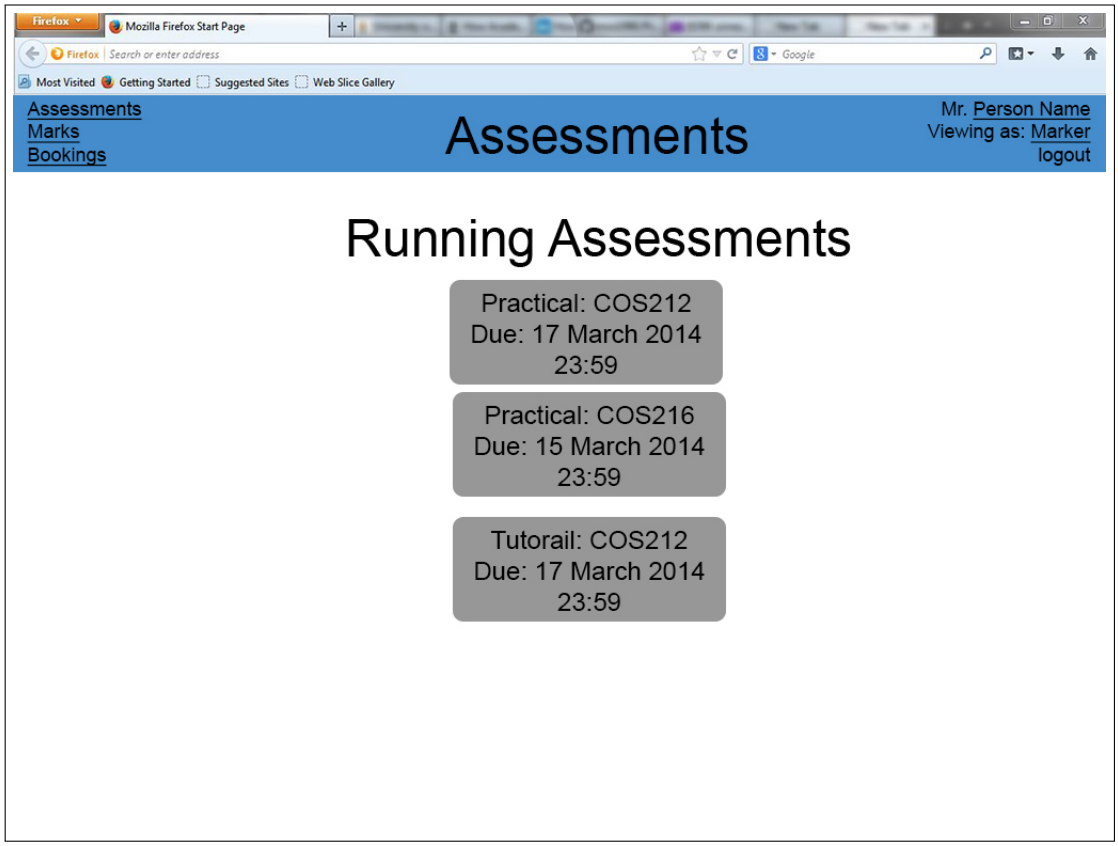

Figure 9: Web UI screen design: Marker Landing Page

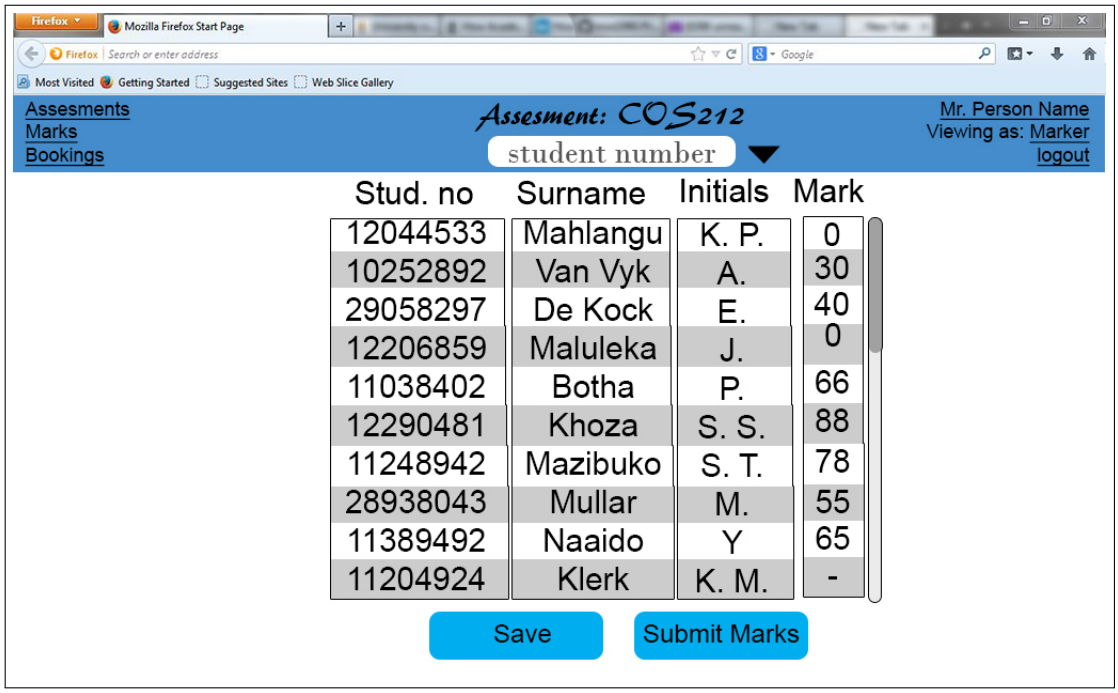

Figure 10: Web UI screen design: Marker Assessment

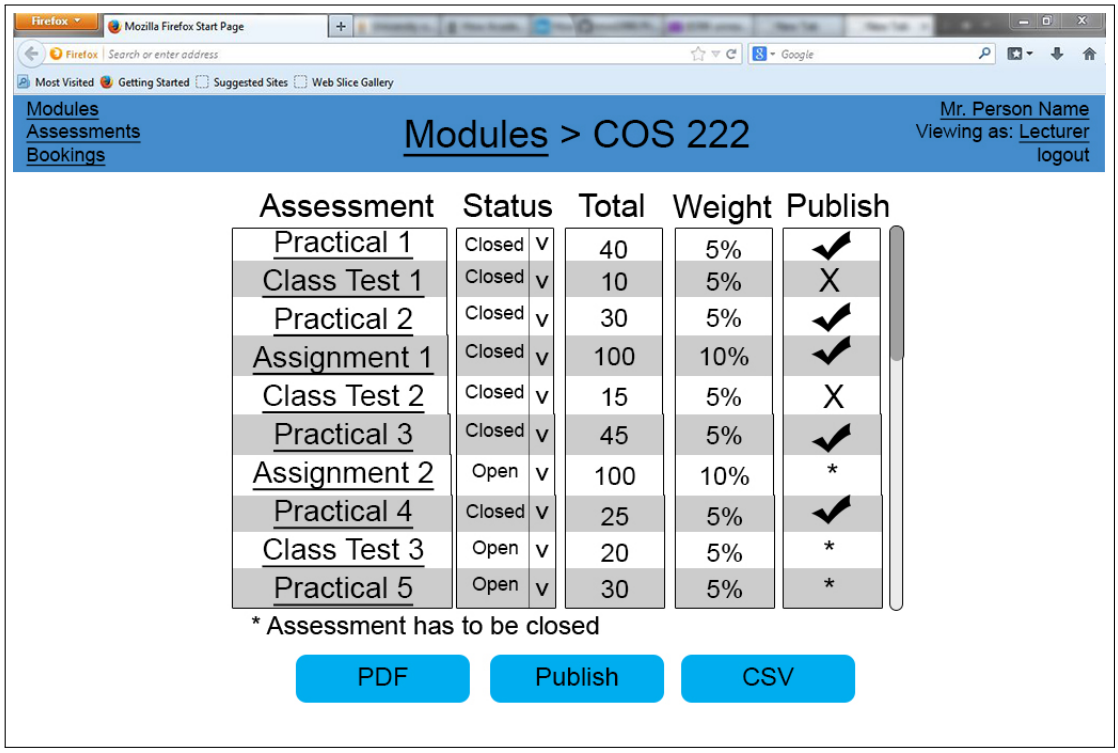

Figure 11: Web UI screen design: Lecturer Module

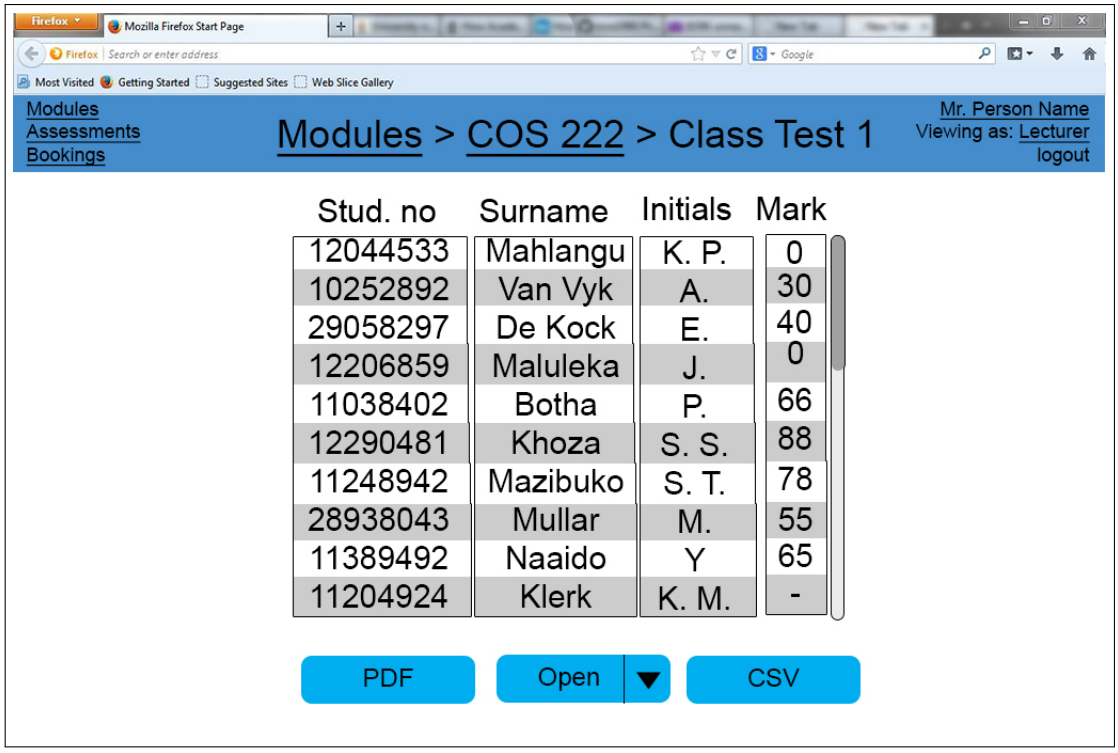

Figure 12: Web UI screen design: Lecturer Module Assessment

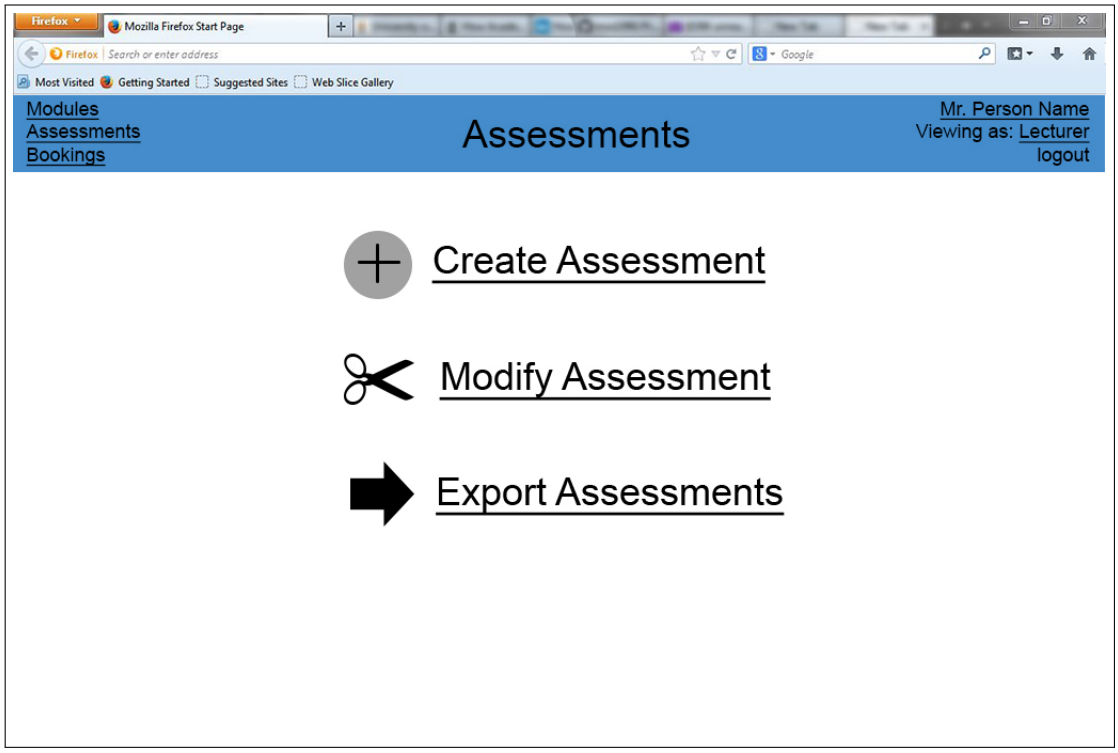

Figure 13: Web UI screen design: Lecturer Assessment

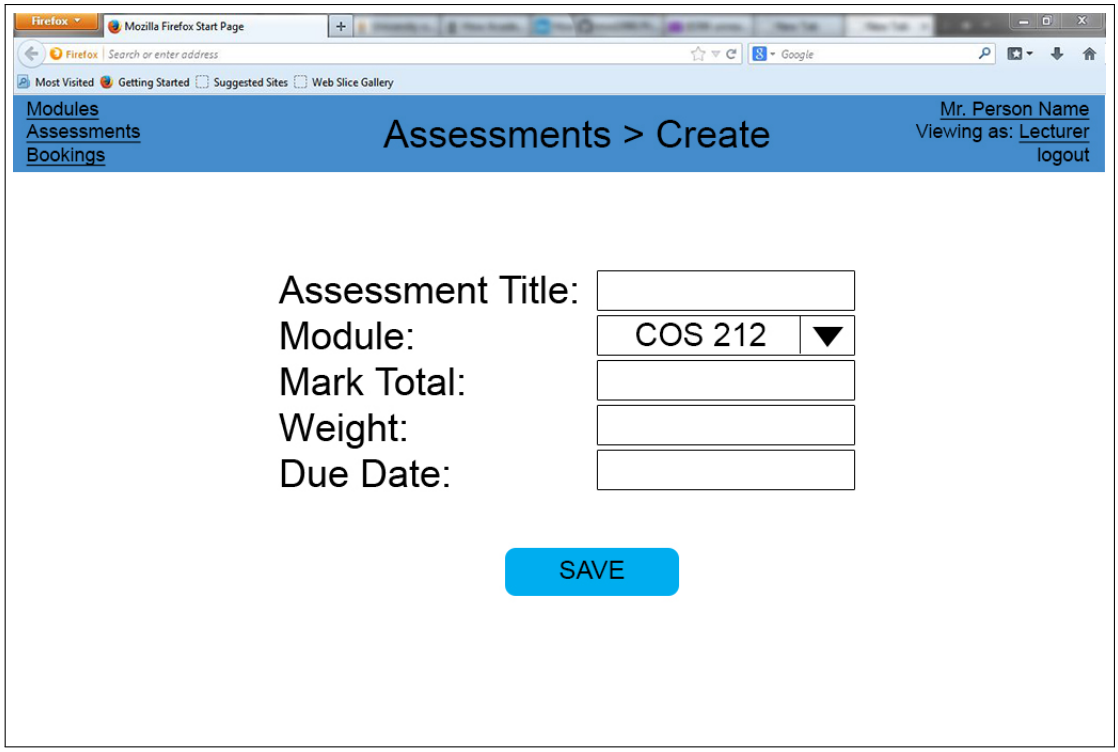

Figure 14: Web UI screen design: Lecturer Assessment Create

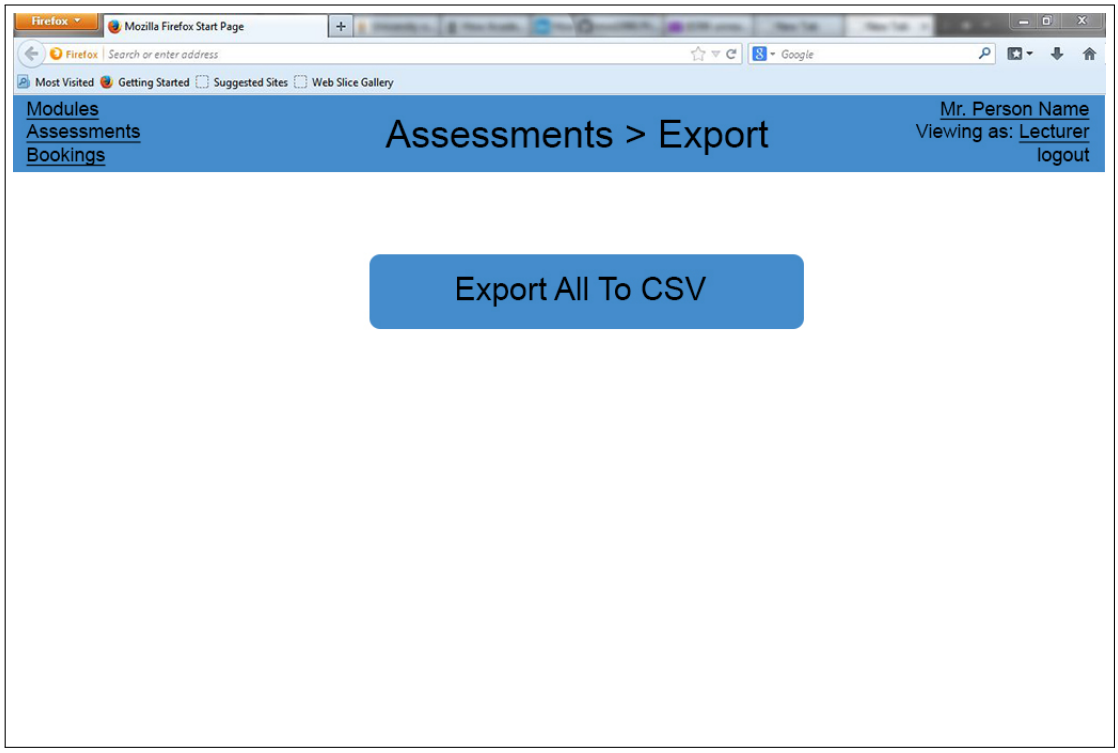

Figure 15: Web UI screen design: Lecturer Assessment Export

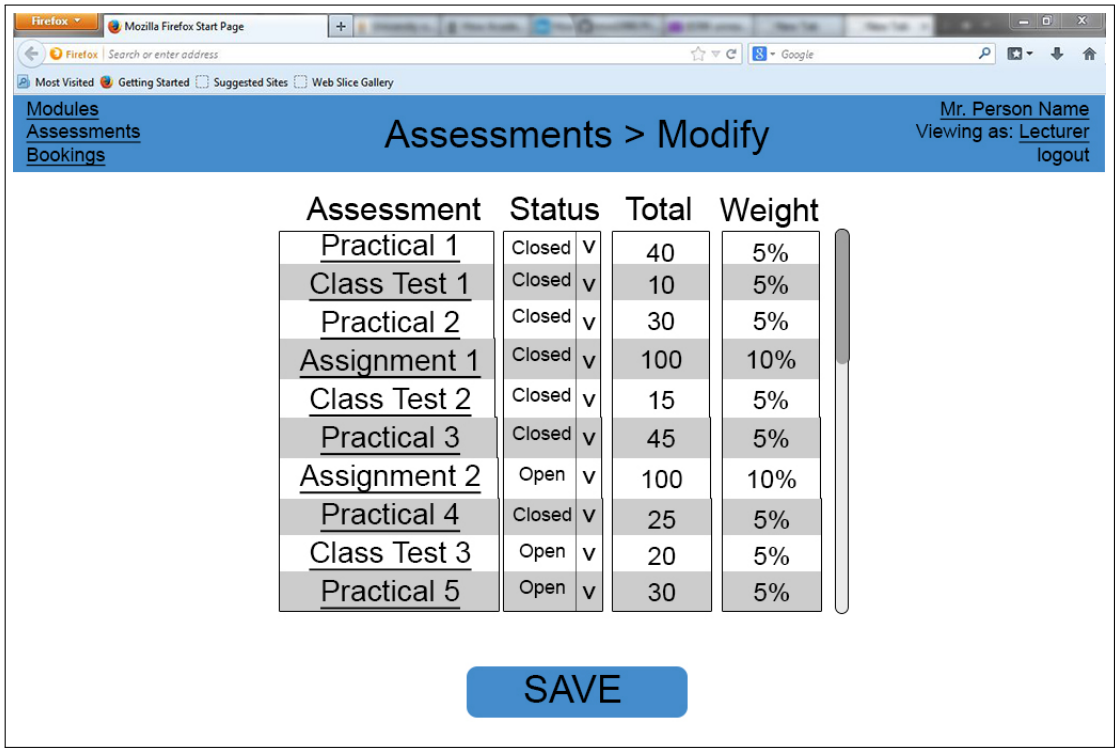

Figure 16: Web UI screen design: Lecturer Assessment Modify

2.3 Android application

2.3.1 API

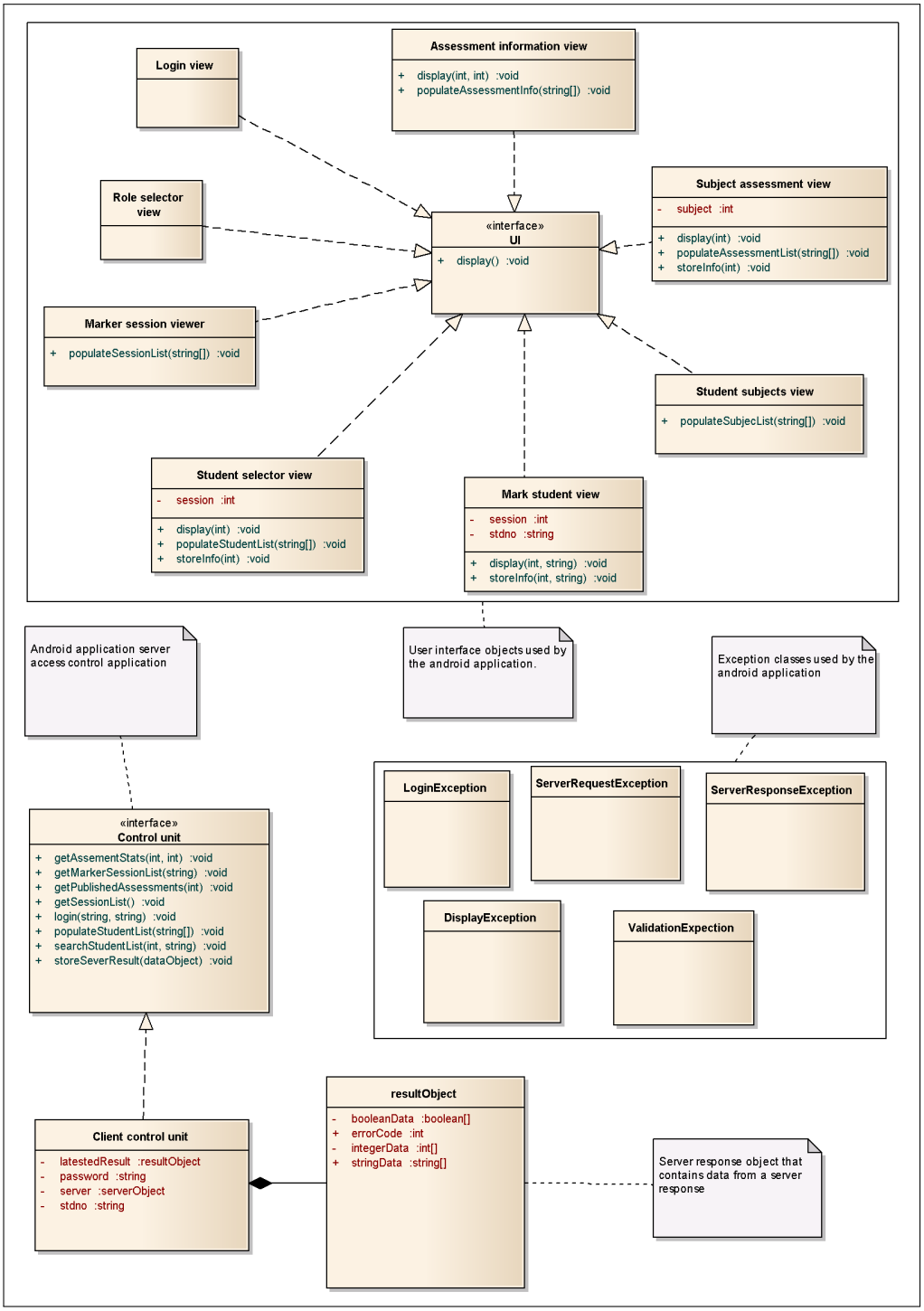

Figure 17: Android Application API diagram

2.3.2 Process specifications

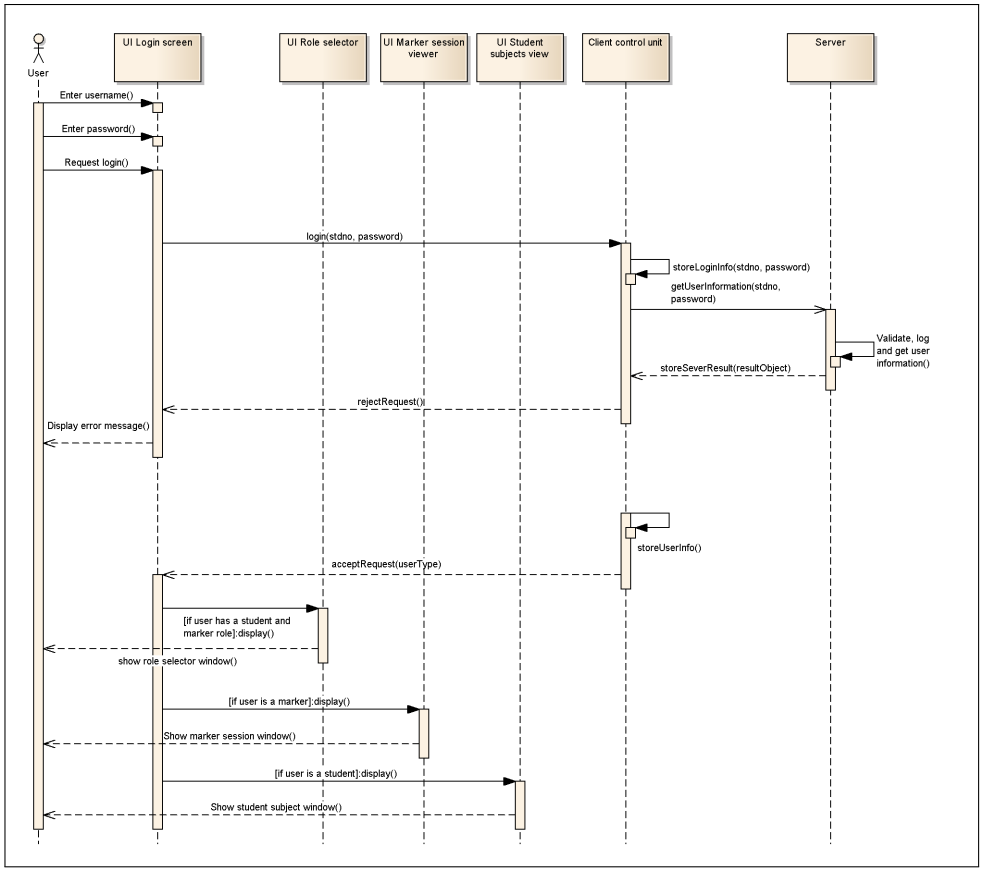

Figure 18: Android Application process specifications: Login UI processes

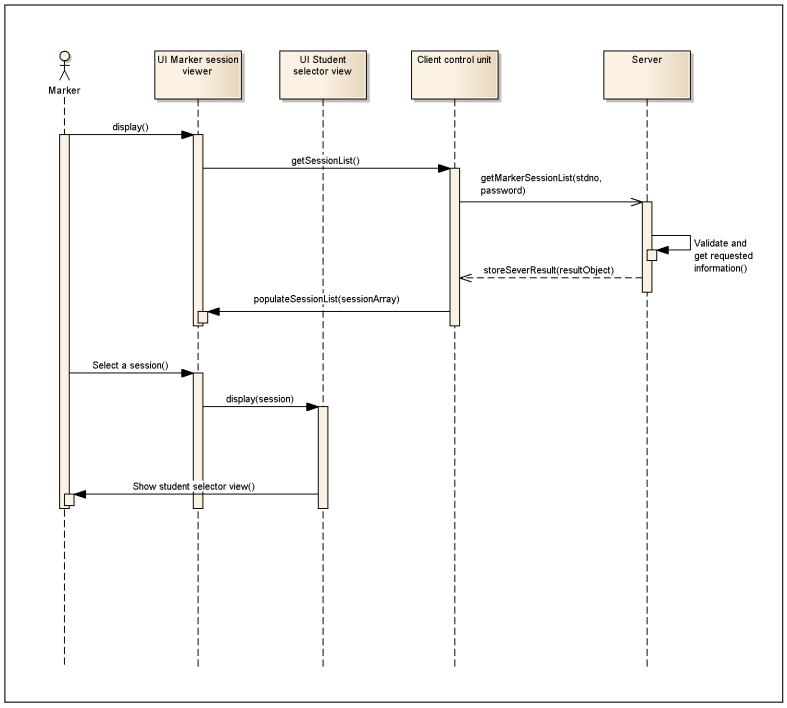

Figure 19: Android Application process specifications: Marker session view UI processes

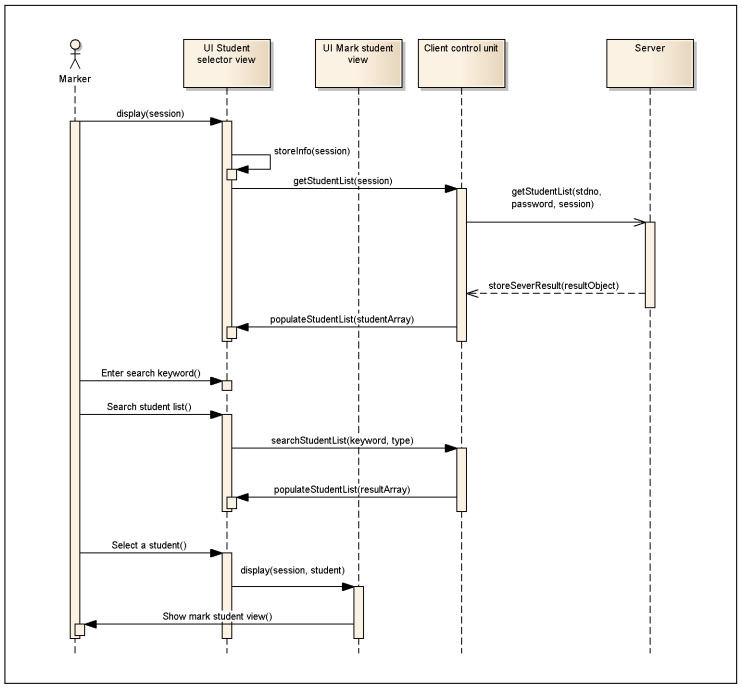

Figure 20: Android Application process specifications: Student selector view UI processes

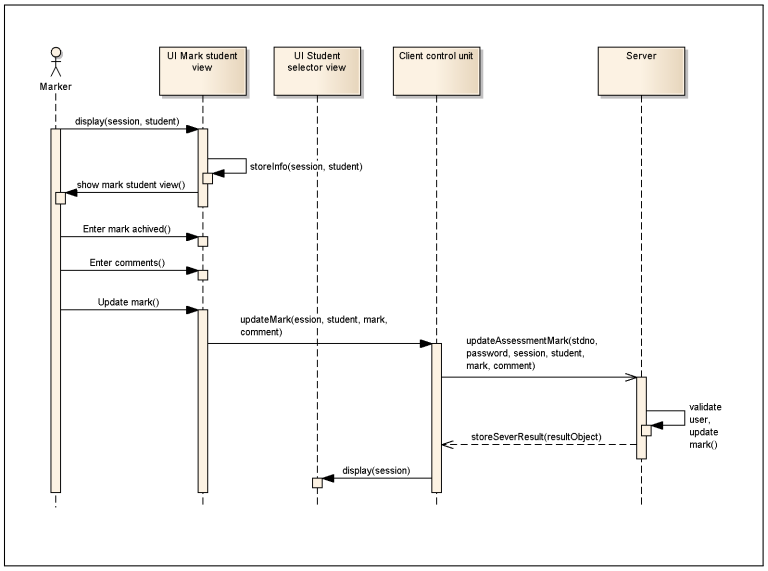

Figure 21: Android Application process specifications: Mark student view UI processes

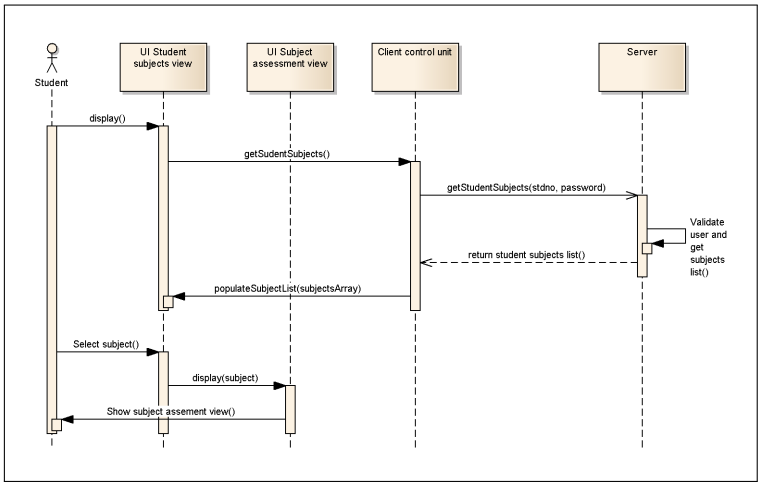

Figure 22: Android Application process specifications: Student subjects view UI processes

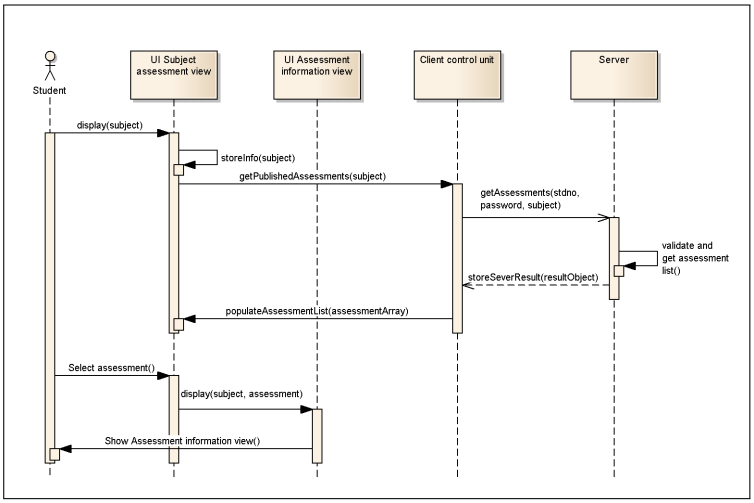

Figure 23: Android Application process specifications: Subject's assessments view UI processes

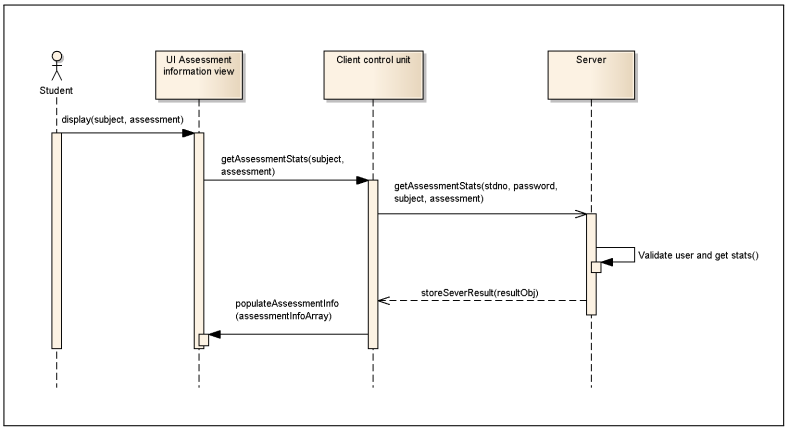

Figure 24: Android Application process specifications: Assessment information view UI processes

2.3.3 UI user work-flow

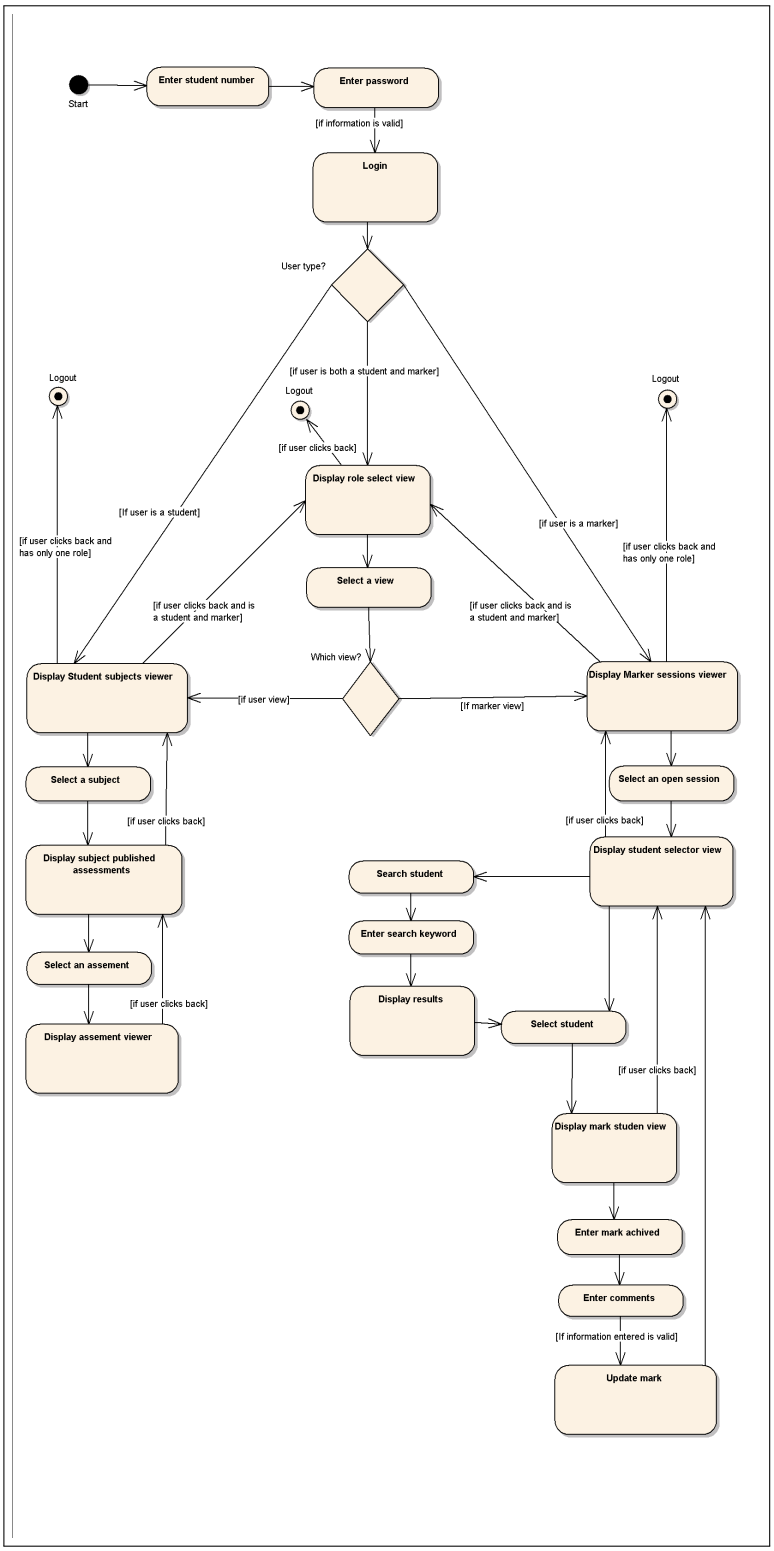

Figure 25: Android Application user work flow: UI

2.3.4 UI Screen design

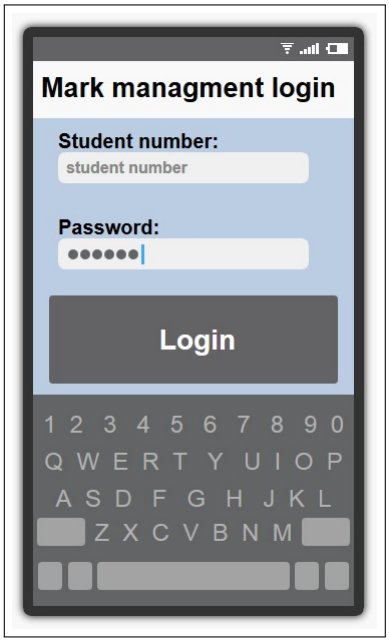

Figure 26: Android Application UI screen design: Login screen

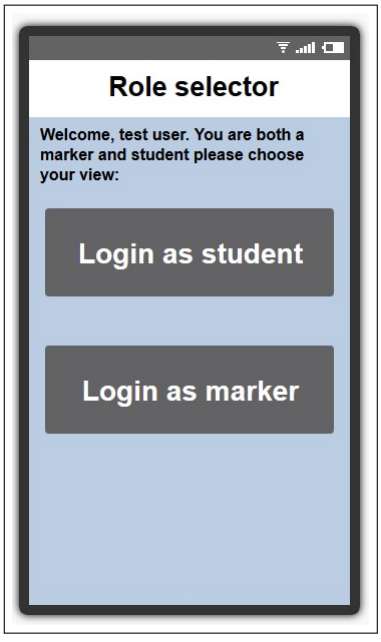

Figure 27: Android Application UI screen design: Role selector screen

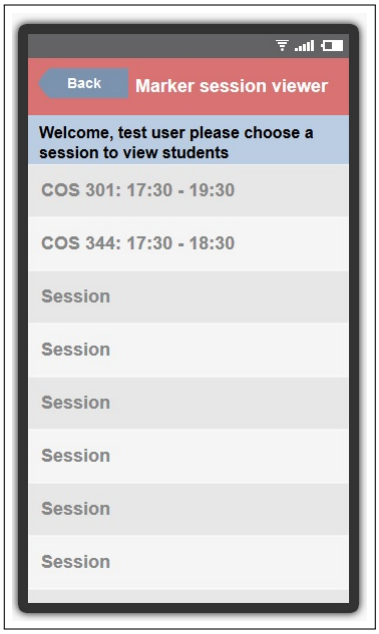

Figure 28: Android Application UI screen design: Marker session viewer screen

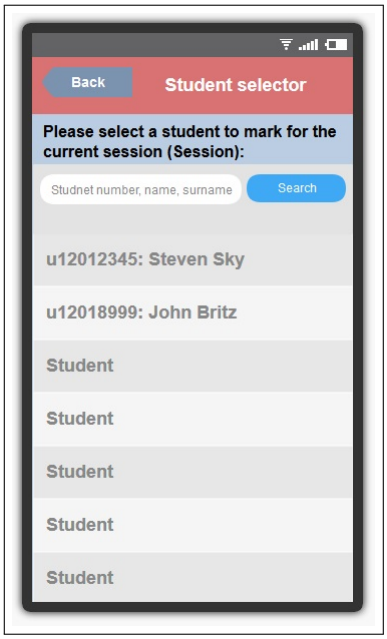

Figure 29: Android Application UI screen design: Student selector screen

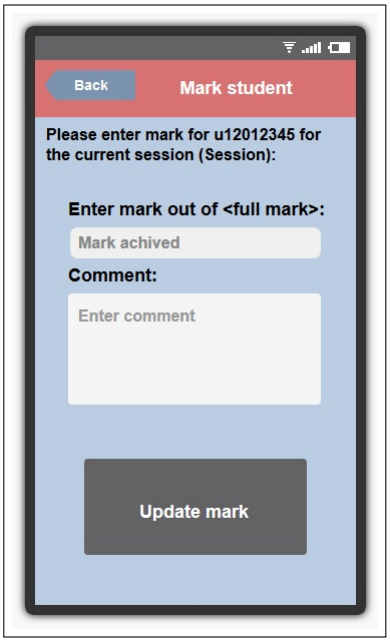

Figure 30: Android Application UI screen design: Mark student screen

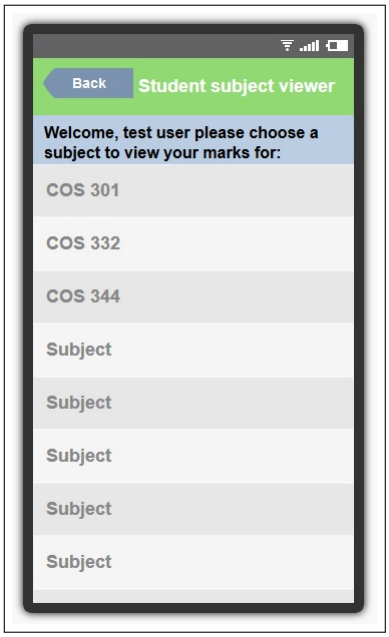

Figure 31: Android Application UI screen design: Student subject viewer screen

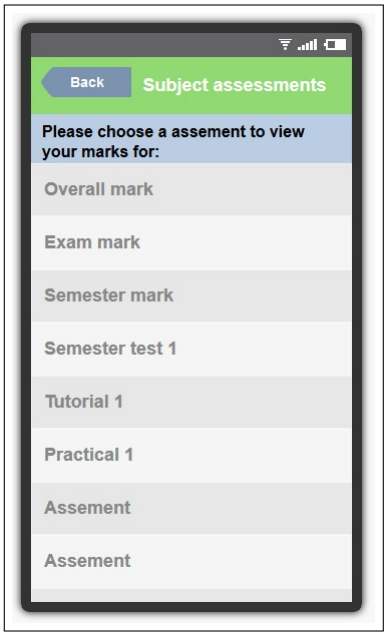

Figure 32: UI screen diagram: Android Application Subject assessments screen

Figure 33: UI screen diagram: Android Application Assessment viewer screen

3 Glossary

- Student - Entailing all students register at the university for specific modules.
- Marker - A grouping of including Teaching Assistance and Tutors, which have permission to assign marks.
- Lecturer - Co-ordinator and/or module presenter.
- Markable item - Including tests, class tests, assignments, practicals.
- Mark list - List consisting of all students registered for a specific module.
- Course - A module presented at the university.
- Web interface - Browser client.
- SSO - Single Sign On.
- LDAP - System used during SSO for authentication.
- API - Application Programming Interface.
- HTTPS - Secured HTTP connection.
- HTML 5 - Standardised version of HTML.
- PDF - The format used in statistics exports.
- CSV - Column Separated Values, used for import of marks and student information.
- SOAP - Simple Object Access Protocol
- WSDL - Web Service Definition Language
- Android - Mobile operating system used.
- Django - Web framework used for the systems back-end.
- Python - Programming language used in Django.
- Java - Used by Android to program application.
- MySQL - Language for database structure and queries.
- OAuth - Open standard for Authorization.
- ORM - Object Relational Mapper