Contents

[1. Introduction 5](#_Toc82790)

[1.1 Purpose 5](#_Toc82791)

[1.2 Scope 5](#_Toc82792)

[1.3 Definitions, Acronyms, and Abbreviations 5](#_Toc82793)

[1.4 Overview 6](#_Toc82794)

[2. Architectural Representation 6](#_Toc82795)

[2.1 Scenarios 6](#_Toc82796)

[2.2 Logical view 6](#_Toc82797)

[2.3 Process view 6](#_Toc82798)

[2.4 Development view 7](#_Toc82799)

[2.5 Physical view 7](#_Toc82800)

[3. Architectural Goals and Constraints 7](#_Toc82801)

[3.1 Server side 7](#_Toc82802)

[3.2 Client Side 7](#_Toc82803)

[3.3 Security 7](#_Toc82804)

[3.4 Persistence 8](#_Toc82805)

[3.5 Reliability / Availability 8](#_Toc82806)

[3.6 Performance 8](#_Toc82807)

[3.7 Portability and reuse 8](#_Toc82808)

[3.8 Development tools 8](#_Toc82809)

[3.9 Schedule 9](#_Toc82810)

[4. Use-Case View 11](#_Toc82811)

[4.1 Use case diagrams 11](#_Toc82812)

[4.2 Use-Case Realizations 14](#_Toc82813)

[4.2.1 User Management related use cases 14](#_Toc82814)

[4.2.2 Lecturer specific use cases 18](#_Toc82815)

[4.2.3 Student specific use cases 24](#_Toc82816)

[4.2.4 Admin specific use cases 27](#_Toc82817)

[5. Logical View 29](#_Toc82818)

[5.1 Overview 29](#_Toc82819)

[5.1.1 Subsystems 29](#_Toc82820)

[5.1.2 Layering 30](#_Toc82821)

[5.2 Architecturally Significant Design Packages 31](#_Toc82822)

[5.2.1 Class diagram 31](#_Toc82823)

[5.2.2 ER diagram 32](#_Toc82824)

[6. Process View 33](#_Toc82825)

[6.1 System sequence diagrams 33](#_Toc82826)

[6.1.1 User logs into the system 33](#_Toc82827)

[6.1.2 User invokes “forget password” 34](#_Toc82828)

[6.1.3 User changes the password 34](#_Toc82829)

[6.1.4 User logs out from the system 35](#_Toc82830)

[6.1.5 Admin creates a new user 35](#_Toc82831)

[6.1.6 User edits the profile 36](#_Toc82832)

[6.1.7 Lecturer creates a new MCQ 36](#_Toc82833)

[6.1.8 Lecturer creates a new essay question 37](#_Toc82834)

[6.1.9 Lecturer generates a MCQ paper using the MCQs that are already saved in the database 37](#_Toc82835)

[6.1.10 Lecturer schedules a paper 38](#_Toc82836)

[6.1.11 Lecturer marks a paper 38](#_Toc82837)

[6.1.12 Lecture views results 39](#_Toc82838)

[6.1.13 Student attempts a MCQ paper 39](#_Toc82839)

[6.1.14 Student enrolls to a new subject 40](#_Toc82840)

[6.1.15 Admin adds a new subject 41](#_Toc82841)

[6.1.16 Admin adds a lecturer for a subject 41](#_Toc82842)

[6.1.17 Admin adds a batch of new students 41](#_Toc82843)

[6.2 Activity Diagrams 42](#_Toc82844)

[6.2.1 Student enrolls to a subject 42](#_Toc82845)

[6.2.2 Student attempts MCQ quiz 43](#_Toc82846)

[6.2.3 Student attends essay quiz 44](#_Toc82847)

[6.2.4 User Views grades and performance 45](#_Toc82848)

[6.2.5 Admin adds a new batch 46](#_Toc82849)

[6.2.6 Admin adds a new subject 47](#_Toc82850)

[6.2.7 Admin maps lecturers with subject 48](#_Toc82851)

[6.2.8 Lecturer Submits new MCQ 49](#_Toc82852)

[6.2.9 Lecturer Submits new essay 50](#_Toc82853)

[6.2.10 Lecturer generates MCQ paper 51](#_Toc82854)

[6.2.11 Lecturer generates essay paper 52](#_Toc82855)

[6.2.12 Lecturer schedules a paper 53](#_Toc82856)

[6.2.13 Lecturer marks essay paper 54](#_Toc82857)

[7. Deployment View 54](#_Toc82858)

[8. Implementation View 55](#_Toc82859)

[8.1 Overview 55](#_Toc82860)

[8.2 Layers 56](#_Toc82861)

[9. Data View 57](#_Toc82862)

[9.1 ER diagram 57](#_Toc82863)

[9.2 MYSQL implementation 57](#_Toc82864)

[10. Size and Performance 58](#_Toc82865)

[11. Quality 58](#_Toc82866)

**Software Architecture Document**

# Introduction

“Online examination system (OES)” is a platform to hold online examinations. It caters to many requirements of holding online examinations. The system can generate statistical data for the records. The system makes it possible to maintain a repository of questions, and then generates papers at a later stage, such that the lecturer has more flexibility over holding online quizzes. Furthermore it provides the functionality to mark the papers automatically.

This document elaborates the software architecture document for the system “Online Examination System (OES)”. The system architecture is abstracted into many views and components which are explained in detail. The document follows the 4+1 view model as the reference model for this document.

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

This document elaborates the architecture of the system in 5 different views. (4+1 view model). Both static and dynamic behavior of the system is described in this document. All the required diagrams and their descriptions are available in this document.

Using 4+1 view model makes it possible to depict the software as accurately as possible. It allows a wide range of stakeholders to find what they require in the architecture document.

## Scope

The software architecture document applies to each static and dynamic aspect of the system. Since 4+1 view model is used as the reference model, it incorporates many view of the system, thus makes the document complete and consistent.

Under the static behavior of the system, the document discusses the class diagrams, package diagrams and other static architecture designs. Dynamic aspects of the system are elaborated using use case realizations and system sequence diagrams.

## Definitions, Acronyms, and Abbreviations

OES – Online examination system (project name)

OOP – Object oriented programming

MVC – Model view control architecture

## Overview

The report will present a detailed analysis of the architecture of Online Examination system. Further sections cover the architectural representation of the project including architectural representation, architectural goals and constraints and use case realizations. The later sections cover the detailed specific details of the 4 main views (logical view, process view, deployment view and implementation view) of the system. Furthermore, the document describes the data view and performance, quality views.

# Architectural Representation

This section details the architecture using the views defined in the “4+1” model. The views used to document the OES (online examination system) application are:

## Scenarios

**Audience**:

All the stakeholders of the system, including the end-users.

**Area**:

Describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system. Other than the basic work flow the documents addresses the exception cases, exception outputs, and other related use cases.

## Logical view

**Audience**:

Designers, Programmers, Testing staff **Area**:

Functional requirements, object hierarchy, system layers

Describes the design of object model. Also describes the subsystems of the system and their relationships.

## Process view

**Audience**:

Integrators, Programmers

**Area**: Non-functional requirements, describes the design's concurrency and synchronization aspects. Elaborates the run time behavior of the system.

## Development view

**Audience**:

Programmers, Code testers **Area**:

Software components: describes the modules and subsystem divisions of the system.

## Physical view

**Audience**:

Database administrators, System engineers, Deployment managers **Area**:

Persistence: describes the architecturally significant persistent elements in the data model. Describes the mapping of the software onto the hardware and shows the system's distributed aspects

# Architectural Goals and Constraints

## Server side

OES (Online examination system) will be hosted at “Hoerku” JSP server. Being a web based application, this underlying client OS can be any PC operating system. (Windows, Linux, Apple). MySQL will be used as the central database server. All communication with client has to comply with public HTTPS, TCP/IP communication protocol standards.

## Client Side

Users will be able to access OES only online. At the initial stage this will be hosted as a standalone application, but with modifications, it can be integrated with a system like university MOODLE system, so that users may use the features using the major system (like MOODLE). Clients/users are expected to use a modern web browser such as Mozilla Firefox 10, Internet Explorer 9, Google Chrome or Safari to get full user experience. .

## Security

Central security features are handled by the institute officials. They’ll be given full access features both in the application and database levels. Creating user accounts for lecturers, students are done by the admin staff. After the users are added to the system, they’ll be provided with a default password, which can be changed by the user. All the passwords are encrypted in order to ensure higher security. Responses and commits made by the students can be seen only by the relevant lecturer and admin.

## Persistence

All the data will be saved in the central server. This is a rational database that implements the 3rd normal form. (MySQL). In order to maintain ACID (Atomicity, Confidentially, Integrity, Durability) some measures have been taken such as encrypting passwords, using transactions for all database commits and etc.

## Reliability / Availability

The system will be subjected to several testing operations (Unit testing, integration testing, system testing) before being deployed in order to make sure that the system is reliable. The MYSQL database server can respond to many number of clients at a given moment without losing consistency and data integrity. .

## Performance

The system responds to any request under standard database and web server script timeouts (30 seconds), also system performance can depend on available hardware, network and internet connection capabilities. Especially the statistical information generation tasks may take comparatively high time. Therefore, actual performance can be determined only after the system is deployed and tested.

## Portability and reuse

Even though OES is designed as a complete and standalone product, it’s possible to extend the product to integrate in a system like MOODLE. In order to maintain reusability, all the functionalities are very well structured and layered. Best practices of RUP are followed throughout the project and the project strictly adheres to OOP standards.

## Development tools

The project incorporates many development tools.

Programming: NetBeans IDE

Database: XAMPP

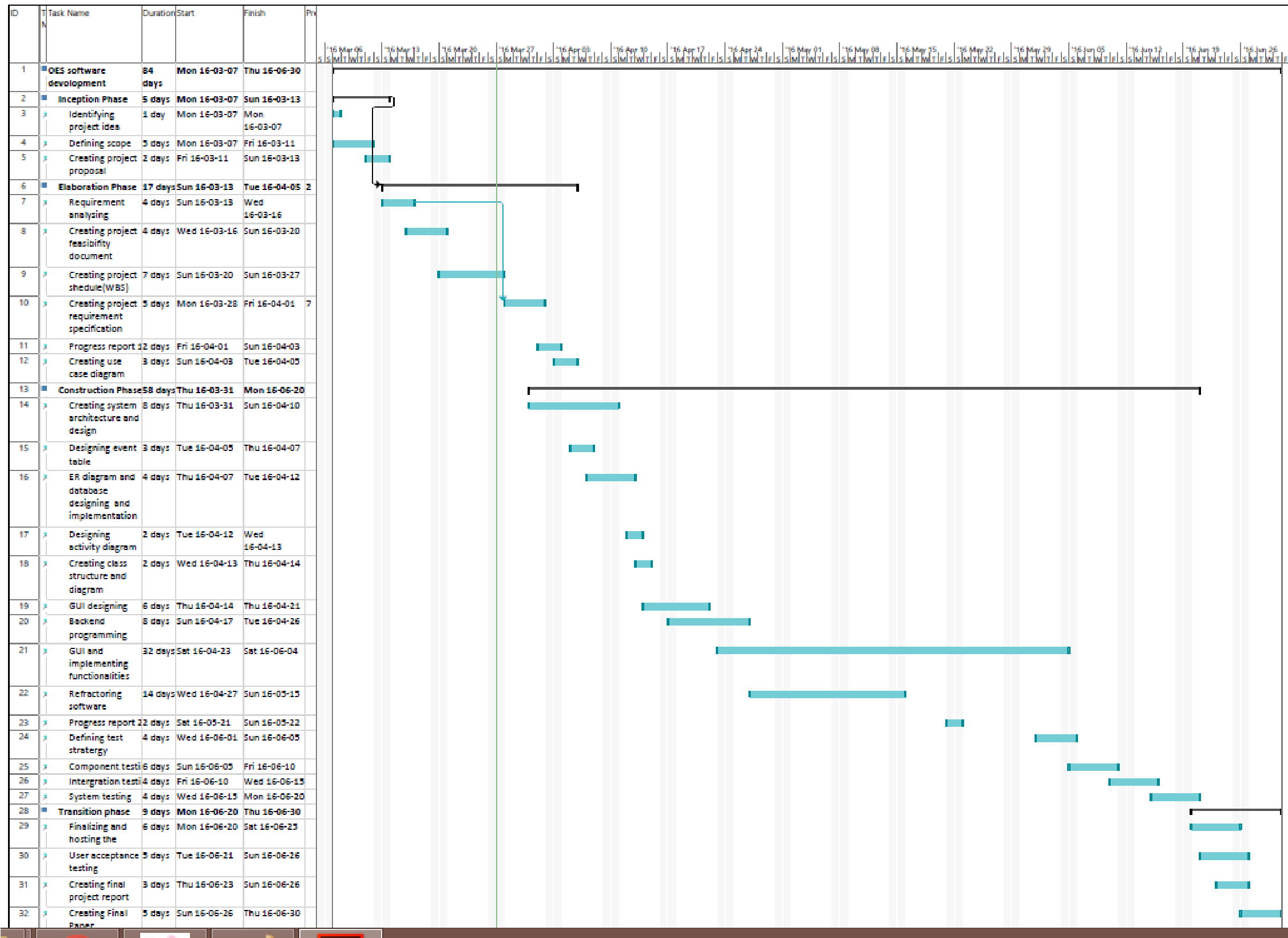
Diagrams: Draw.IO, CREATLY, GENMYMODEL

Database connection: MYSQL connector

Schedule: Microsoft Project

## Schedule

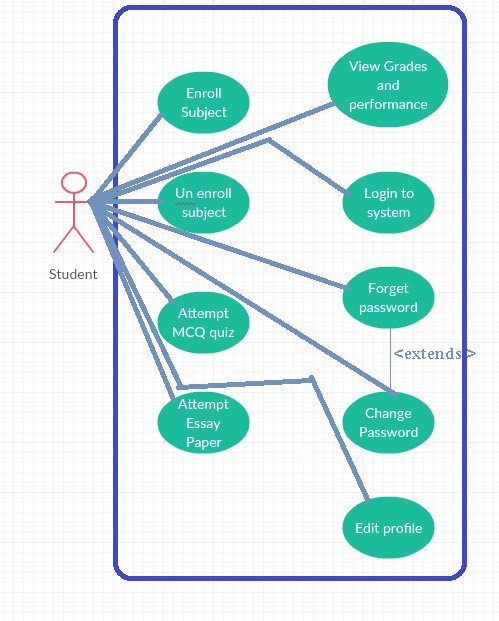
The project is designed using RUP workflows. All the activities come under one of 4 main phases, Inception phase, Elaboration phase, construction phase and transition phase. All the deliverables and relevant deadlines are considered and satisfied in this work break down structure. The following diagram shows the GAANT chart of the OES.

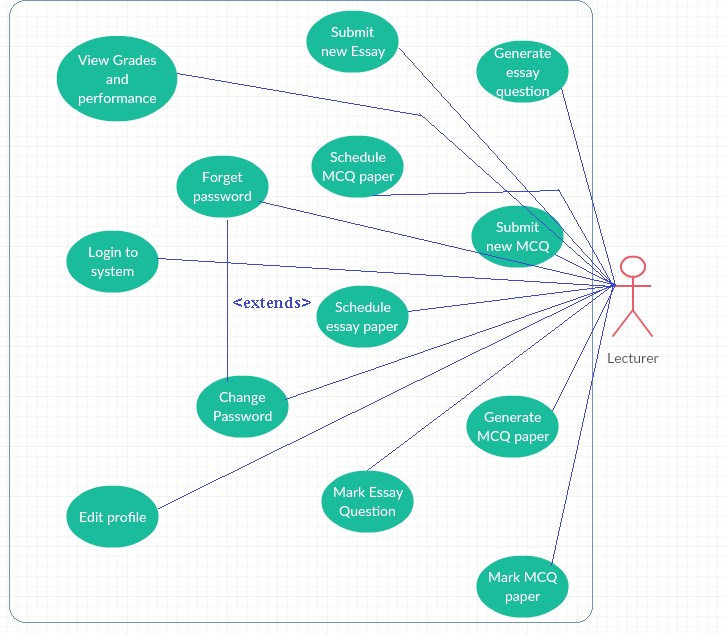


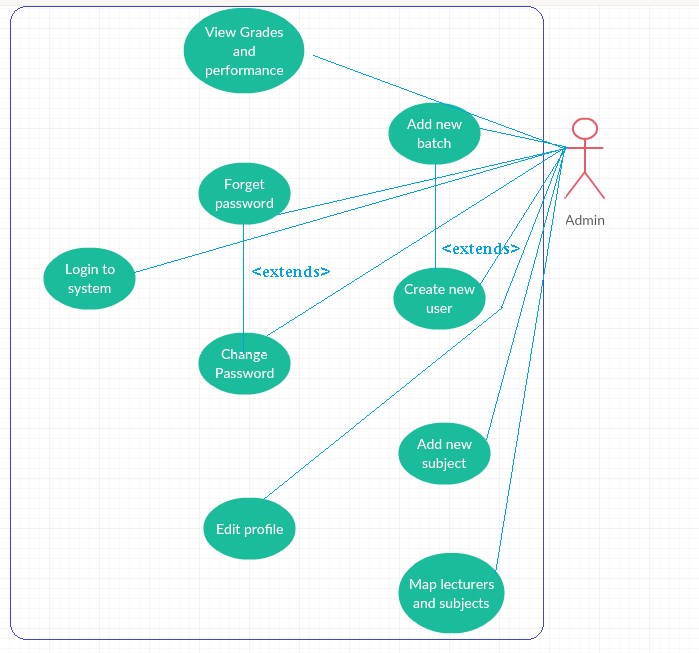


# Use-Case View

## Use case diagrams







## Use-Case Realizations

### User Management related use cases

4.2.1.1 User logs into the system

|  |  |
| --- | --- |
| Use case name | Login to the system. |
| Scenario | Login to the system using username and password. |
| Triggering event | User enters the username and password and submit. |
| Brief description | When user inputs username and password, system checks for the validity and the matching of input values and redirects to the home page of user. (the home page depends on the user and user type) |
| Actors | Student, Lecturer, Admin |
| Related use cases | Null. |
| Preconditions | The user should be a valid user (should possess a username and password). |
| Post conditions | If the user is a valid user, he or she should be redirected to the correct home page. |
| Flow of events | User loads the login page.  User inputs the username and password and then submit. System check the validity of the entered password and username.  If this is a valid user, the user will get redirected to respective home page. |
| Exception conditions | If the entered username and password is invalid, the user will be asked to enter them again.  If the user has forgotten the password, he can click  “forget password”. |

4.2.1.2 User invokes “forget password”

|  |  |
| --- | --- |
| Use case name | User invokes “forget password” |
| Scenario | When the user forgets the password, he or she can invoke forget password and go through a recovery process. |
| Triggering event | User clicks “forget password” button. |
| Brief description | When user clicks “forget password“, he can ask the |
|  | system to send a mail to his email address, from which he can change the password to a new one. |
| Actors | Student, Lecturer, Admin |
| Related use cases | Login to the system. |
| Preconditions | The user should be a valid user, and should have an email address that is saved in the system. |
| Post conditions | User will be able to change the password. |
| Flow of events | User clicks the forget password.  Then using the window, user can ask for a mail which contents a security key.  User types that security key in the following interface. Then if the code is correct, user will be redirected to change password page. |
| Exception conditions | If the user doesn’t have a pre-set email address, the user can’t continue this process and the user should contact the system admin and reset the password.  If the user enters the wrong code, he can re attempt and get a new code. |

4.2.1.3 User changes password

|  |  |
| --- | --- |
| Use case name | User changes the password. |
| Scenario | The user changes his or her password. |
| Triggering event | User enters previous password, new password and the confirmations and submit. |
| Brief description | When the user goes to the “change password” section, he or she can change the password. For that the user should input the previous password again. The new password and confirm new password fields should be identical. The password should adhere to the security standards that are enforced by the application policy  (more than 8 characters long, should contain alphanumeric and symbolic characters).  Or else when a user forgets the password and using the code that is sent via an email the user can access the  “change password” section. |
| Actors | Student, Lecturer, Admin |
| Related use cases | Login to the system, User invokes “forget password”. |
| Preconditions | The user should be a valid user (should possess a username and password) and should have a working account.  User should remember the previous password (or else should go through forget password section). |
| Post conditions | User gets a new password. |
| Flow of events | User clicks change password control.  User inputs previous password.  User inputs new password and the “confirm new password” section.  System checks the equality of two entered passwords, and whether the new password adheres to the password standards that are enforced.  User submits the form.  If the user has forgotten the previous password, he should follow the forget password scenario. |
| Exception conditions | If the password doesn’t adhere to the rules of the system, it would ask for another password.  If the new password filed and “confirm new password” fields doesn’t match, it’ll ask to re-enter the passwords. |

4.2.1.4 User log outs from the system

|  |  |
| --- | --- |
| Use case name | User logs out from the system. |
| Scenario | User who has currently logged into the system logs out. |
| Triggering event | User clicks logout button. |
| Brief description | When a user who has already logged in to the system clicks “logout” he’ll get log out from the system, so that the session variables will get reset. |
| Actors | Student, Lecturer, Admin |
| Related use cases | Login to the system. |
| Preconditions | The user have logged in to the system already. |
| Post conditions | User should get redirected to the login page. |
| Flow of events | User clicks “logout”.  System removes the session values for this user User gets redirected to the login page. |
| Exception conditions | None. |

4.2.1.5 Create a new user

|  |  |
| --- | --- |
| Use case name | Create a new user. |
| Scenario | The admin creates a new user account. |
| Triggering event | The admin submit a new user account creation activity. |
| Brief description | When a new user has to be added to the new system, the admin adds him using “create new user” work flow. If the new user is a lecturer or an admin, the user will get added separately, whereas students will get added as a batch.  The new user gets a username and a password. The password can be reset later. |
| Actors | Admin |
| Related use cases | None |
| Preconditions | The new user should possess all the required attributes of the user entity (NIC, email, etc.) |
| Post conditions | New user should get access to the system. |
| Flow of events | New user provides the essential details to an admin.  Admin performs “create new user” activity. Admin submits the form.  System saves a new user account.  A new username and a password is given to the new user. |
| Exception conditions | None |

4.2.1.6 User edits his own profile

|  |  |
| --- | --- |
| Use case name | User edits the profile. |
| Scenario | The user edits his own user details. |
| Triggering event | User invokes the “edit user” configuration. |
| Brief description | The user who has already logged in to the system, edits his details. The username and the password can’t be changed at this point. Some other parameters such as  NIC can’t be changed as well. |
| Actors | Student, Lecturer, Admin |
| Related use cases | None |
| Preconditions | The user should have logged in to the system already. |
| Post conditions | User details get updated. |
| Flow of events | User clicks “edit profile” tab.  User submits the “edit” form.  System checks for the validity of new values entered. System saves the new information. |
| Exception conditions | New information doesn’t adhere to the system rules: System asks the user to resubmit. |

### Lecturer specific use cases

4.2.2.1 Submit new MCQ

|  |  |
| --- | --- |
| Use case name | Lecturer submits a new MCQ. |
| Scenario | A new MCQ is added to the system. |
| Triggering event | Lecturer adds a new MCQ. |
| Brief description | The lecturer adds a new MCQ to one of his enrolled subjects. The system checks the completeness of the question and saves it. |
| Actors | Lecturer, System |
| Related use cases | None |
| Preconditions | The new MCQ should possess all the required fields. The lecturer should be logged in to the system. The lecturer should be enrolled to the subject to which the new MCQ belongs. |
| Post conditions | New MCQ added to the specified subject. |
| Flow of events | Lecturer adds all the fields of “add a new MCQ” option. Lecturer submits the form.  System checks for the validity of entered values.  System saves the MCQ. |
|  |  |
| Exception conditions | None. |

4.2.2.2 Submit new essay questions

|  |  |
| --- | --- |
| Use case name | Lecturer submits a new essay question. |
| Scenario | A new essay question is added to the system. |
| Triggering event | Lecturer adds a new essay question. |
| Brief description | The lecturer adds a new essay question to one of his enrolled subjects. The system checks the completeness of the question and saves it. |
| Actors | Lecturer, System |
| Related use cases | None |
| Preconditions | The new essay question should possess all the required fields. The lecturer should be logged in to the system. The lecturer should be enrolled to the subject to which the new essay question belongs. |
| Post conditions | New essay question added to the specified subject. |
| Flow of events | Lecturer adds all the fields of “add a new essay question” option.  Lecturer submits the form.  System checks for the validity of entered values. System saves the essay question. |
| Exception conditions | None. |

4.2.2.3 Generate papers using MCQ

|  |  |
| --- | --- |
| Use case name | Generate MCQ paper. |
| Scenario | Lecturer generates a new MCQ paper. |
| Triggering event | Lecturer invokes create new paper option. |
| Brief description | The lecturer can create a new MCQ paper for a given subject. Lecturer can use the MCQs that are already saved in the database for that particular subject. This |
|  | paper will be saved in the system. |
| Actors | Lecturer |
| Related use cases | Create new MCQ |
| Preconditions | There should be at least one MCQ for the specified subject of interest. |
| Post conditions | A new MCQ paper is generated for the specified subject. |
| Flow of events | Lecturer selects a subject out of the subjects he has enrolled in.  Lecturer invokes create new MCQ paper option. Lecturer selects n number of MCQs that are there in the database.  Lecturer adds a description and submit action.  System saves the new paper. |
| Exception conditions | No MCQ is available for that subject: System sends a warning saying that no MCQ is found for that specific subject. |

4.2.2.4 Schedule a MCQ paper

|  |  |
| --- | --- |
| Use case name | Schedule a new MCQ paper. |
| Scenario | The lecturer schedules a new MCQ paper. |
| Triggering event | A new paper is scheduled. |
| Brief description | Using the MCQ papers that are already generated previously, the lecturer can schedule them. Lecturer specifies the attributes such as starting date, time and then submit.  The system validates the input values and send a notification to enrolled students of that subject. |
| Actors | Lecturer, Student |
| Related use cases | Create new MCQ paper. |
| Preconditions | At least one MCQ paper should exist for the specified subject. |
| Post conditions | A notification is sent to all enrolled students. |
| Flow of events | Lecturer clicks the “schedule new MCQ paper” option. System displays all the available papers.  Lecturer selects one paper and specifies the related attributes such as due date, time.  Lecturer submits the values.  System validates the entered attributes.  System sends a notification to all the enrolled students of the particular subject. |
| Exception conditions | No paper is available: System displays a warning message. |

4.2.2.5 Schedule an essay question

|  |  |
| --- | --- |
| Use case name | Schedule a new essay paper. |
| Scenario | The lecturer schedules a new essay paper. |
| Triggering event | A new paper is scheduled. |
| Brief description | Using the essay papers that are already generated previously, the lecturer can schedule them. Lecturer specifies the attributes such as starting date, time and then submit.  The system validates the input values and send a notification to enrolled students of that subject. |
| Actors | Lecturer, Student |
| Related use cases | Create new essay paper |
| Preconditions | At least one essay paper should exist for the specified subject. |
| Post conditions | A notification is sent to all enrolled students. |
| Flow of events | Lecturer clicks the “schedule new essay paper” option. System displays all the available papers.  Lecturer selects one paper and specifies the related attributes such as due date, time.  Lecturer submits the values.  System validates the entered attributes.  System sends a notification to all the enrolled students of the particular subject. |
| Exception conditions | No paper is available: System displays a warning message. |

4.2.2.6 Mark MCQ paper

|  |  |
| --- | --- |
| Use case name | Mark MCQ paper. |
| Scenario | The system marks the MCQ paper |
| Triggering event | The student submits a MCQ paper.  Allocated time finishes for a MCQ paper. |
| Brief description | When a student finishes a MCQ paper, the system automatically marks the answers. The prompts for each question is saved in the system. |
| Actors | Student, System |
| Related use cases | Attempt MCQ quiz - student |
| Preconditions | Student should submit the completed MCQ quiz. |
| Post conditions | System records the prompts of students for each question, and generate the marks. |
| Flow of events | Student finishes MCQ quiz.  System locks the quiz.  System compares student answers with the exact answers.  System displays the results to the student immediately. |
| Exception conditions | None. |

4.2.2.7 Mark essay question

|  |  |
| --- | --- |
| Use case name | Mark essay question. |
| Scenario | The system marks the essay question |
| Triggering event | The student submits an essay question.  Allocated time finishes for an essay question. |
| Brief description | When a student finishes an essay question, the system automatically marks the answers based on keywords. |
| Actors | Student, System |
| Related use cases | Attempt essay - student |
| Preconditions | Student should submit the completed essay question. |
| Post conditions | System records the answers of students, and generate marks. |
| Flow of events | Student finishes essay question.  System locks the quiz.  System marks the student answer based on available keywords.  System displays the results to the student immediately. |
| Exception conditions | None. |

4.2.2.8 View results

|  |  |
| --- | --- |
| Use case name | View results |
| Scenario | Lecturer view results of students |
| Triggering event | Lecturer asks for statistical information about a subject he’s enrolled |
| Brief description | The lecturer can get statistical information about the subjects that he is offering. This includes performance of students for a particular quiz, performance of a batch and etc. |
| Actors | Lecturer, System |
| Related use cases | None |
| Preconditions | None |
| Post conditions | None |
| Flow of events | Lecturer invokes “view statistical info” option. System displays all the supported forms.  Lecturer selects one category.  System displays statistical information. |
| Exception conditions | None. |

### Student specific use cases

4.2.3.1 Attempt MCQ quiz

|  |  |
| --- | --- |
| Use case name | Attempt MCQ paper. |
| Scenario | Student attempts an online MCQ paper. |
| Triggering event | Student clicks “start quiz option”. |
| Brief description | Student start the MCQ quiz. The timer is set. Student can mark an answer from the 5 available answers. Student then submits the paper. |
| Actors | Student |
| Related use cases | None |
| Preconditions | The MCQ paper should exist.  The student should have enrolled to the subject. |
| Post conditions | The responses are recorded in the system. |
| Flow of events | The student invokes “start attempt” option under the specific MCQ quiz.  System displays the paper.  Student selects the answers from the given 5 prompts.  Student submits answers.  System records the prompts. |
| Exception conditions | None |

4.2.3.2 Attempt essay question

|  |  |
| --- | --- |
| Use case name | Attempt essay question. |
| Scenario | Student attempts an online essay question. |
| Triggering event | Student clicks “start quiz option” |
| Brief description | Student start the essay question. The timer is set.  Student enters the answer.  Student then submits the paper. |
| Actors | Student |
| Related use cases | None |
| Preconditions | The essay question should exist.  The student should have enrolled to the subject. |
| Post conditions | The responses are recorded in the system. |
| Flow of events | The student invokes “start attempt” option under the specific essay question.  System displays the paper.  Student enters the answer.  Student submits paper.  System records the prompt. |
| Exception conditions | None |

4.2.3.3 View grades and performance

|  |  |
| --- | --- |
| Use case name | View grades and performance. |
| Scenario | Student views his own performance |
| Triggering event | Student asks for statistical information |
| Brief description | The student can get statistical information about the subjects that he has enrolled in. This performance history of papers. |
| Actors | Student, System |
| Related use cases | None |
| Preconditions | None |
| Post conditions | None |
| Flow of events | Student invokes “view statistical info” option. System displays all the supported forms.  Student selects one category.  System displays statistical information. |
| Exception conditions | None. |

4.2.3.4 Enroll for a subject

|  |  |
| --- | --- |
| Use case name | Enroll for a new subject. |
| Scenario | Student enrolls to a new subject. |
| Triggering event | Student clicks “enroll new subject” option. |
| Brief description | The student can enroll for a subject that he is permitted to. All the subjects that the student can enroll are displayed in the window. The student can select one of them and submit. |
| Actors | Student |
| Related use cases | None. |
| Preconditions | The subject should exists and it should be permitted to be enrolled by the particular student. |
| Post conditions | Student enrolls to a new subject. |
| Flow of events | Student invokes “enroll new subject option”.  System shows all the available and permitted subjects.  Student selects a particular subject and click “enroll”. System saves the response. |
| Exception conditions | None. |

4.2.3.5 Un enroll for a subject

|  |  |
| --- | --- |
| Use case name | Un enroll from a subject. |
| Scenario | Student un enrolls from a subject. |
| Triggering event | Student clicks “un enroll subject” option. |
| Brief description | The student can un enroll from a subject that he has already enrolled to. |
| Actors | Student |
| Related use cases | Enroll new subject. |
| Preconditions | The student should have enrolled to that subject previously. |
| Post conditions | Student un enrolls from the particular subject. |
| Flow of events | Student invokes “un enroll subject option”. System shows all the enrolled subjects.  Student selects a particular subject and click “un enroll”. System saves the response. |
| Exception conditions | None. |

### Admin specific use cases

4.2.4.1 Add new subject

|  |  |
| --- | --- |
| Use case name | Add a new subject. |
| Scenario | Admin adds a new subject. |
| Triggering event | Admin selects “add a new subject” prompt. |
| Brief description | The admin can add a new subject to the system. Initially there will be no assigned lecturers or students. |
| Actors | Admin |
| Related use cases | None |
| Preconditions | None |
| Post conditions | A new subject is added to the system. |
| Flow of events | Admin clicks “add new subject” option. System displays the form.  Admin inputs all the necessary attributes and submit. System saves the response. |
| Exception conditions | None |

4.2.4.2 Map lecturers with subject

|  |  |
| --- | --- |
| Use case name | Map lecturers with subject. |
| Scenario | Admin maps a lecturer with a subject. |
| Triggering event | Admin maps a lecturer with a subject. |
| Brief description | Admin maps a subject with a lecturer. A lecturer can have many number of subjects as well as a subject can have many number of lecturers. |
| Actors | Admin |
| Related use cases | None |
| Preconditions | Subject and the lecturer should exist. |
| Post conditions | Lecturer and a subject get mapped. |
| Flow of events | Admin selects a “Map subject” option.  System shows a list of subjects and a list of lecturers.  Admin selects a subject and a lecturer and click submit.  The system saves the response. |
| Exception conditions | None. |

4.2.4.3 Add a new batch of students

|  |  |
| --- | --- |
| Use case name | Add a new batch of students. |
| Scenario | Admin adds a new batch. |
| Triggering event | Admin clicks “add new batch option”. |
| Brief description | When a new batch of students are to be added to the system, the admin adds them as a batch rather than individual students. |
| Actors | Admin |
| Related use cases | None |
| Preconditions | None |
| Post conditions | A new batch gets initiated. |
| Flow of events | The admin clicks “add a new batch option”. System shows the input form.  Admin inputs the parameters.  Admin adds the details of all new students.  Admin submits the form.  System saves the response. |
| Exception conditions | None. |

# Logical View

## Overview

### Subsystems

System OES (Online examination system) can be divided into 3 main sub systems.

1. User management subsystem
2. Subject management subsystem
3. Statistical data management system

5.1.1.1 User management sub system

This sub system provides the facilities that covers all the user management functionalities. Main use cases that comes under this subsystem includes

* + User login
  + New user creation
  + New Batch of students creation
  + Edit profile and change password scenarios

Each user that interacts with the system is an actor of these use cases. Depending on the access levels, some actions will not be permitted.

5.1.1.2 Subject management subsystem

This subsystem involves the main business login of the system. Main use cases that come under this subsystem includes

* + Addition of a new subject
  + Assignment of lecturers to subjects
  + Student enrolments to subjects
  + Submitting MCQs and essays to the system
  + Generating papers
  + Schedule papers
  + Marking papers

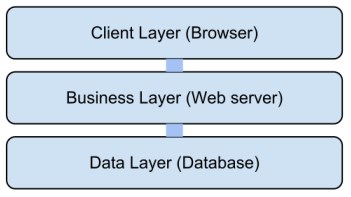
5.1.1.3 Statistical Data management subsystem

This is the subsystem that incorporates most of the algorithms and processing power. Previously mentioned subsystems involve data access operations, but this subsystem includes front end processing. Only the required parameters are fetched from the database, and all the processing and diagram drawing happens in the front end. The system is capable of generating many number of statistical reports including

* + Student performance reports
  + Batch performance reports
  + Performance for a single question

### Layering

OES system is divided into 3 layers. The layering model of the OES application is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development, reusability and maintenance.



(MVC architecture model)

## Architecturally Significant Design Packages

### Class diagram

Since the diagram is not very clear in this document it’s available online.

<https://drive.google.com/file/d/0B_key68-co6Id2NpUW9QV1JOY2c/view?usp=sharing>

Especial design decisions

* All users are derived from the super class called “user”. It’s because each user of the system has common attributes and behaviours such as Id, password, name, NIC etc.
* Each subject can have many number of lecturers and each lecturer can have many number of subjects. That the reason behind having a many to many relationship between lecturer and subject.
* Both MCQ paper and the essay paper have the same characteristics except for few attributes. Hence there exists a super class “Question” which contains the common attributes and functionalities.
* MCQ paper consists of many number of MCQs. Hence an aggregation relationship exists between MCQ paper and MCQ.
* Each student’s marks for any question paper is stored in a Result object.
* Login manager and Login classes are used to keep track of login times and dates of each user.

### ER diagram

Since the diagram is not very clear it’s available in the following link

<https://drive.google.com/file/d/0B_key68-co6IdWJpQk02cDE2TWs/view?usp=sharing>

Especial design decisions

* Like in the class diagram each user has a super entity called ‘user’ that holds all the major functionality.
* Each student has a batch parameter. Entity “batch” contains the details of a particular batch (started date, batch id).
* Entity keywords is used to store the keywords of an essay question. These words are used to get rough statistical information about student’s behaviour towards an essay question.
* The relationship between student, MCQ paper and MCQ is a trinary relationship since the system has to keep track of answers for each question from each student.
* Each MCQ, paper, essay question has an associated “subject code” parameter.
* Active parameter in both user, subject represents whether the content is valid. For an example if a lecturer leave the institute, the entry for that lecturer is not deleted, but marked DEACTIVATED.
* Relationship “Student Batch Eligible” between batch and subjects represents the relationship that contains all the subjects that a particular batch of students can enrol.
* Last login entity keeps track of user login details on order to make sure the security.

# Process View

## System sequence diagrams

When user inputs username and password, system checks for the validity and the matching of input values and redirects to the home page of user. (The home page depends on the user and user type). If the entered username and password is invalid, the user will be asked to enter them again. If the user has forgotten the password, he can click “forget password”.

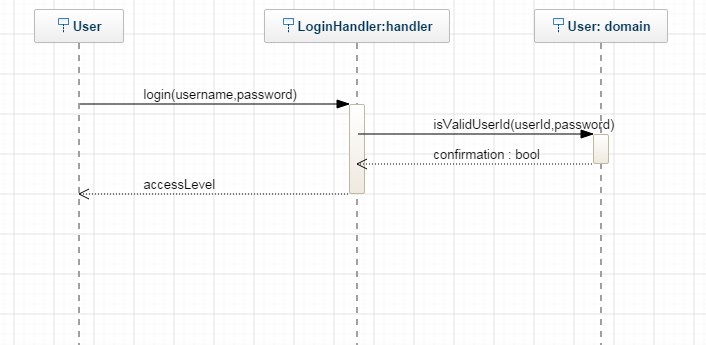
When user clicks “forget password“, he can ask the system to send a mail to his email address, from which he can change the password to a new one. The user should be a valid user, and should have an email address that is saved in the system. If the user doesn’t have a pre-set email address, the user can’t continue this process and the user should contact the system admin and reset the password. If the user enters the wrong code, he can re attempt and get a new code.

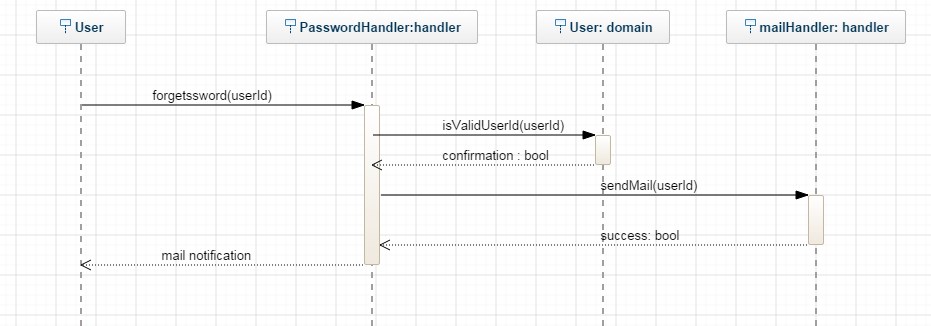
### User logs into the system

When user inputs username and password, system checks for the validity and the matching of input values and redirects to the home page of user. (The home page depends on the user and user type). If the entered username and password is invalid, the user will be asked to enter them again. If the user has forgotten the password, he can click “forget password”.

When user clicks “forget password“, he can ask the system to send a mail to his email address, from which he can change the password to a new one. The user should be a valid user, and should have an email address that is saved in the system. If the user doesn’t have a pre-set email address, the user can’t continue this process and the user should contact the system admin and reset the password. If the user enters the wrong code, he can re attempt and get a new code.

### User invokes “forget password”





### User changes the password

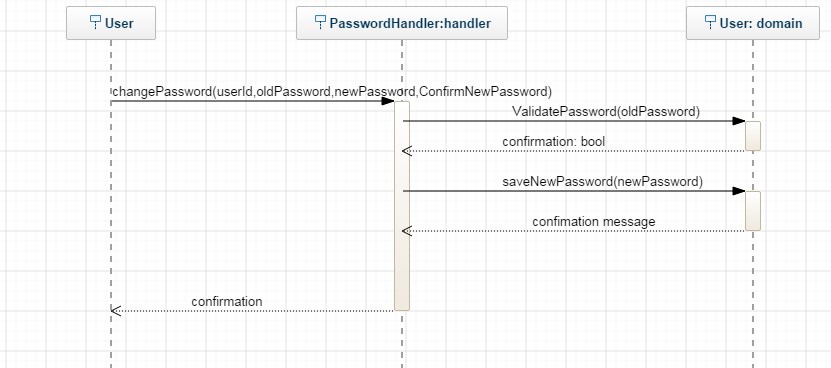
When the user goes to the “change password” section, he or she can change the password. For that the user should input the previous password again. The new password and confirm new password fields should be identical. The password should adhere to the security standards that are enforced by the application policy (more than 8 characters long, should contain alphanumeric and symbolic characters).

Or else when a user forgets the password and using the code that is sent via an email the user can access the

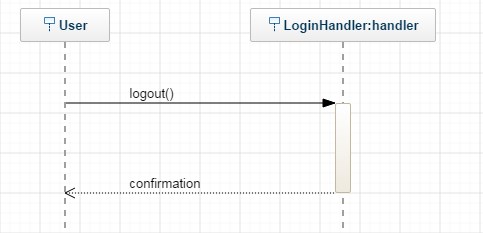
“change password” section.

If the password doesn’t adhere to the rules of the system, it would ask for another password.

If the new password filed and “confirm new password” fields doesn’t match, it’ll ask to re-enter the passwords.



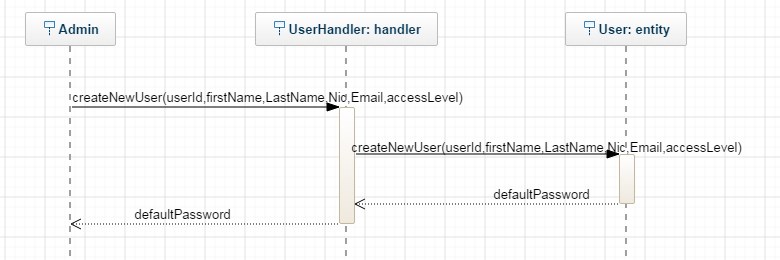
### User logs out from the system



### Admin creates a new user

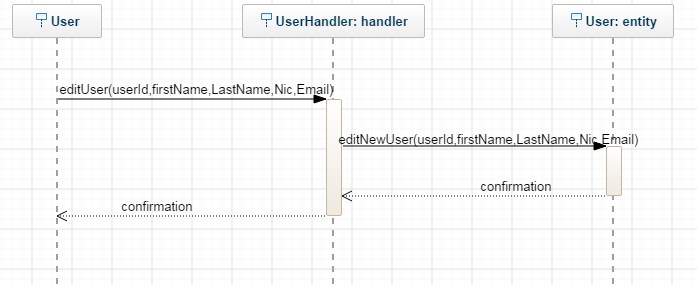
When a new user has to be added to the new system, the admin adds him using “create new user” work flow. If the new user is a lecturer or an admin, the user will get added separately, whereas students will get added as a batch.

The new user gets a username and a password. The password can be reset later



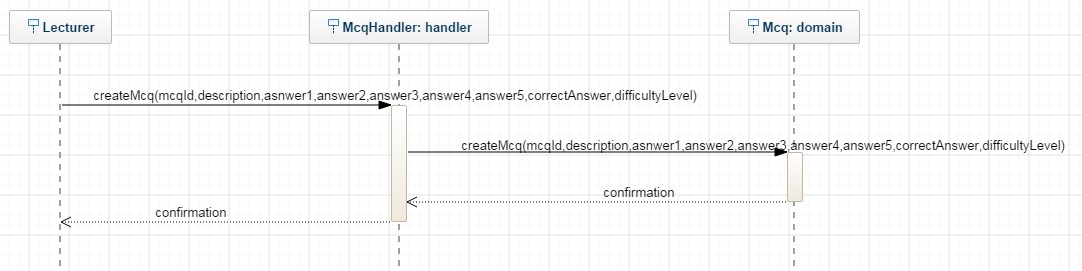
### User edits the profile

The user who has already logged in to the system, edits his details. The username and the password can’t be changed at this point. Some other parameters such as NIC can’t be changed as well.



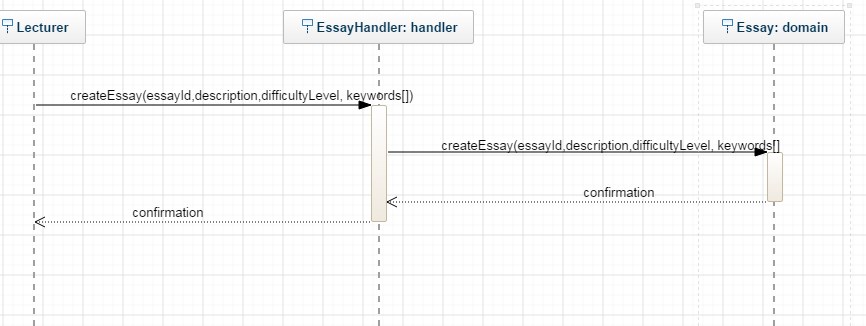
### Lecturer creates a new MCQ

The lecturer adds a new MCQ to one of his enrolled subjects. The system checks the completeness of the question and saves it. The new MCQ should possess all the required fields. The lecturer should be logged in to the system. The lecturer should be enrolled to the subject to which the new MCQ belongs.



### Lecturer creates a new essay question

The lecturer adds a new essay question to one of his enrolled subjects. The system checks the completeness of the question and saves it. The new essay question should possess all the required fields. The lecturer should be logged in to the system. The lecturer should be enrolled to the subject to which the new essay question belongs.



### Lecturer generates a MCQ paper using the MCQs that are already saved in the database.

The lecturer can create a new MCQ paper for a given subject. Lecturer can use the MCQs that are already saved in the database for that particular subject. This paper will be saved in the system.

There should be at least one MCQ for the specified subject of interest.

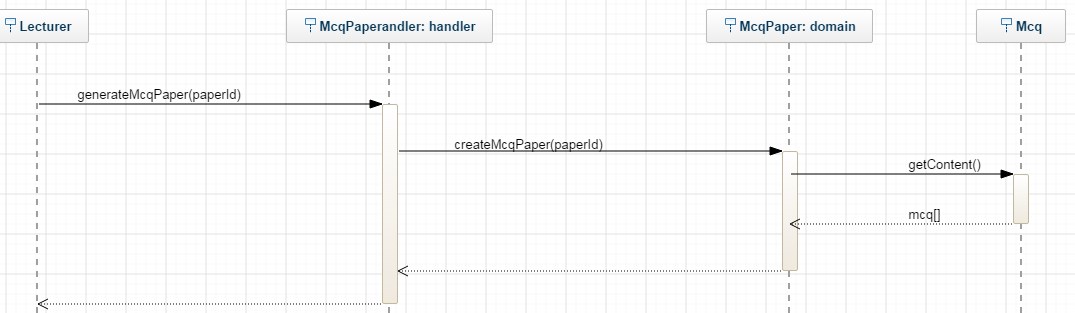
If no MCQ is available for that subject the System sends a warning saying that no MCQ is found for that specific subject.

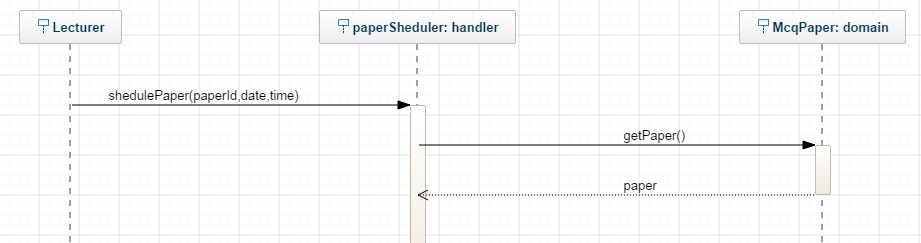
### Lecturer schedules a paper

Using the MCQ papers that are already generated previously, the lecturer can schedule them. Lecturer specifies the attributes such as starting date, time and then submit. The system validates the input values and send a notification to enrolled students of that subject.

At least one MCQ paper should exist for the specified subject.

If no paper is available then the System displays a warning message.



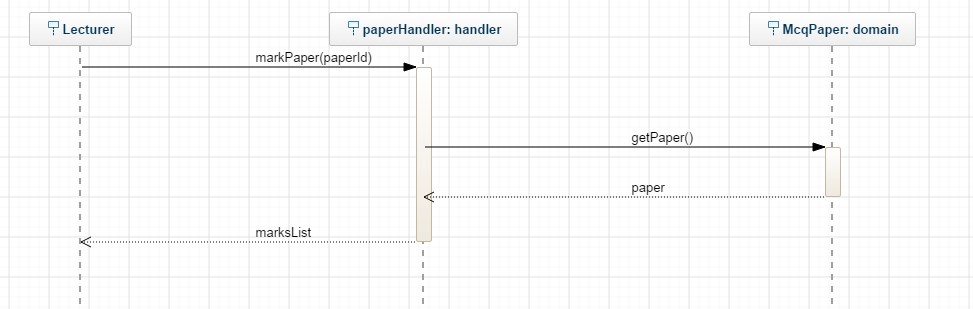


### Lecturer marks a paper

When a student finishes a MCQ paper, the system automatically marks the answers. The prompts for each question is saved in the system.

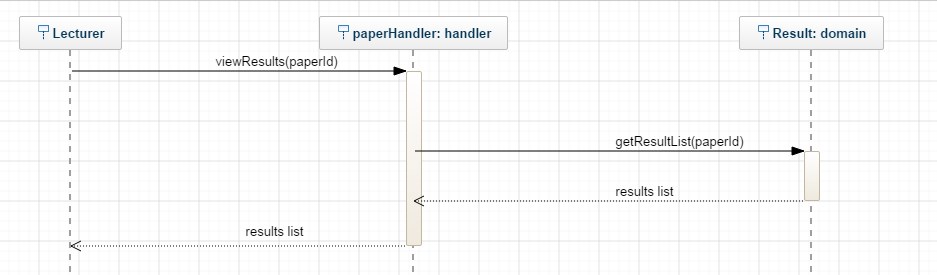
When a student finishes an essay question, the system automatically marks the answers based on keywords.

The keywords are pre saved words that contains the core of the essay content. Unlike in MCQ papers, it’s not possible to have a system which marks the essay questions completely since there is no straight forward answer. This functionality will give a basic idea about the content and it will be use full when generating reports.



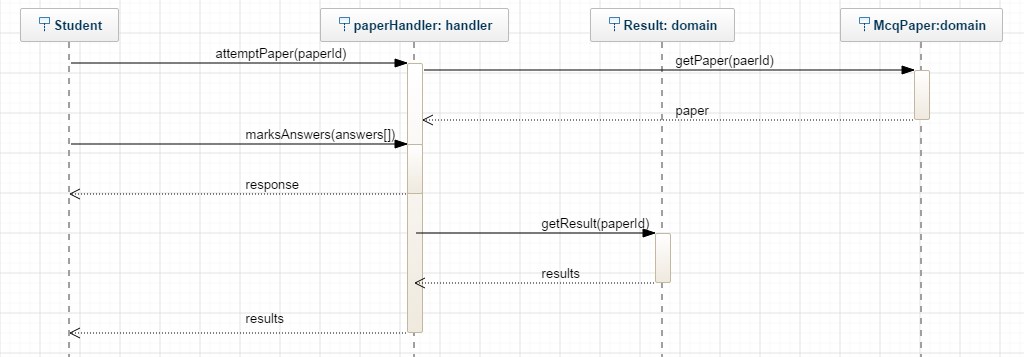
### Lecture views results

The lecturer can get statistical information about the subjects that he is offering. This includes performance of students for a particular quiz, performance of a batch and etc. These details are derived from the values that are stored in the database. The statistical information that a particular user can request depends on the access level of the user.



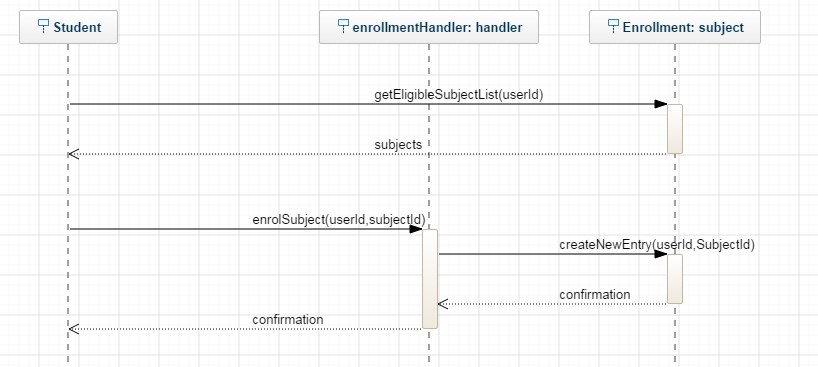
### Student attempts a MCQ paper

Student start the MCQ quiz. The timer is set. Student can mark an answer from the 5 available answers. Student then submits the paper. All the responses are saved and evaluated immediately. The student should have enrolled to the subject in order to access the quiz.



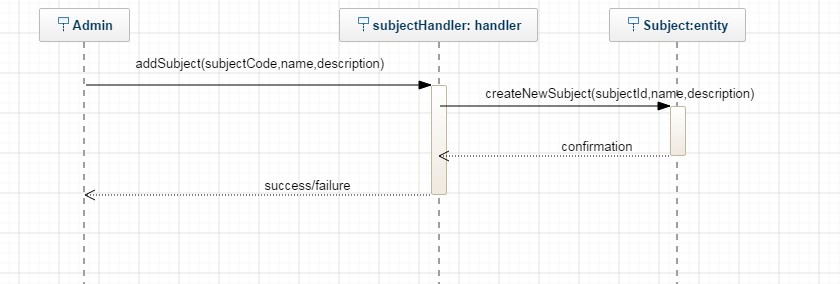
### Student enrolls to a new subject

The student can enroll for a subject that he is permitted to. All the subjects that the student can enroll are displayed in the window. The student can select one of them and submit. Using the “batch id” parameter of the student, the eligible subject list gets generated.



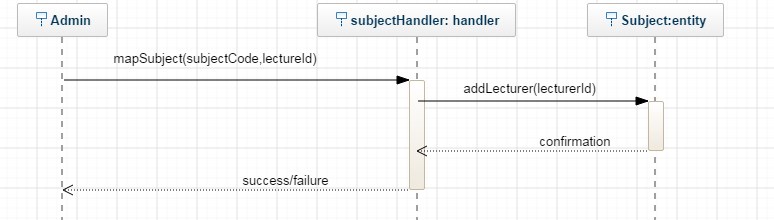
### Admin adds a new subject

The admin can add a new subject to the system. Initially there will be no assigned lecturers or students. Later on the lecturers can be matched with the subject.



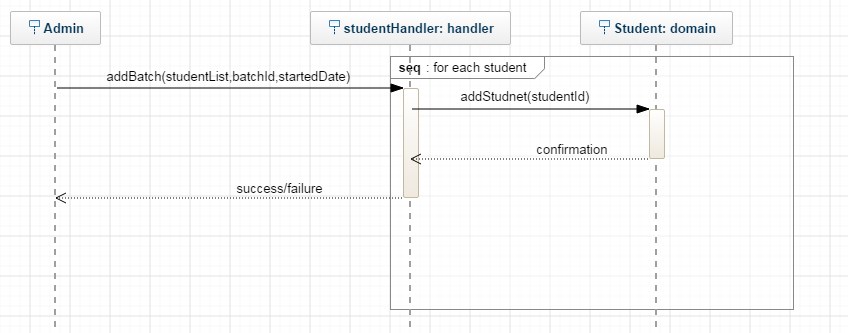
### Admin adds a lecturer for a subject

Admin maps a subject with a lecturer. A lecturer can have many number of subjects as well as a subject can have many number of lecturers. Once a lecturer is assigned to a subject, he gets access to the course page, and the subject appears under the lecturer’s subject list.



### Admin adds a batch of new students

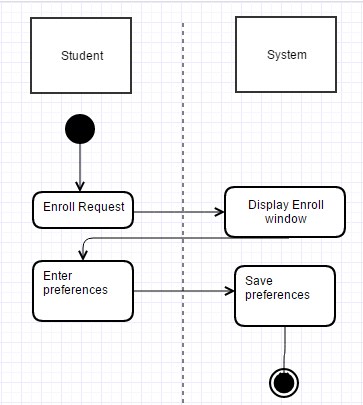
When a new batch of students are to be added to the system, the admin adds them as a batch rather than individual students. In the internal process, the “add a new user” use case will be executed several times.



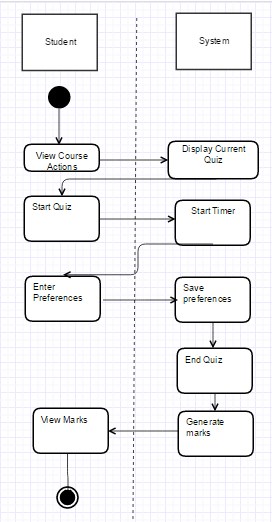
## Activity Diagrams

The activity diagrams for the main business functionalities (use cases) are as bellows.

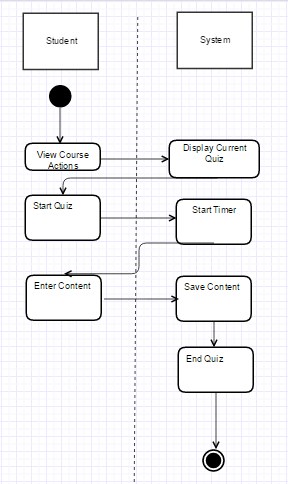
### Student enrolls to a subject



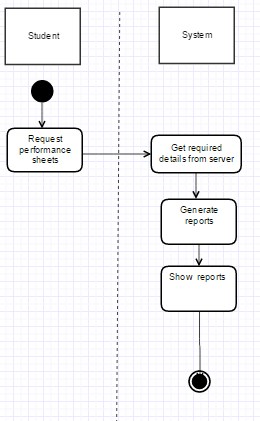
### Student attempts MCQ quiz



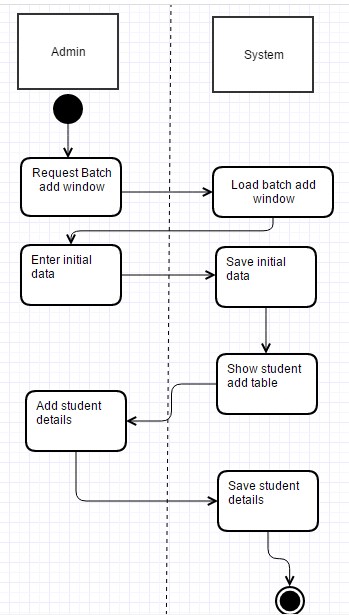
### Student attends essay quiz



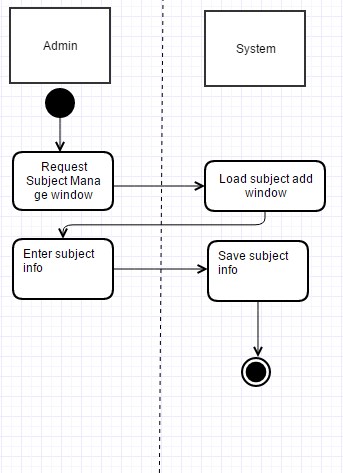
### User Views grades and performance



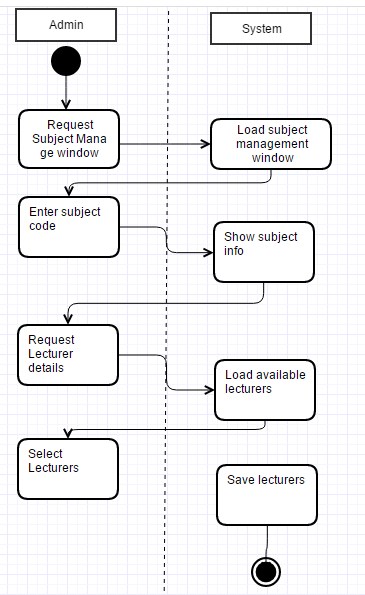
### Admin adds a new batch



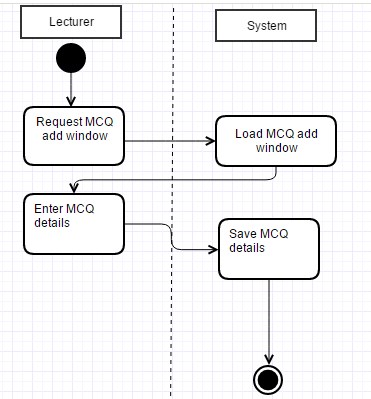
### Admin adds a new subject



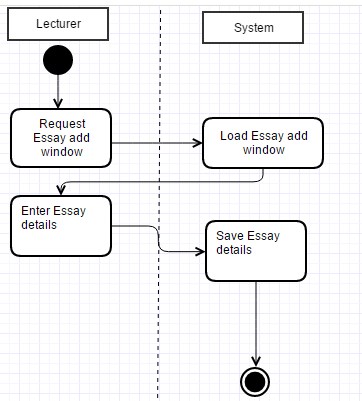
### Admin maps lecturers with subject



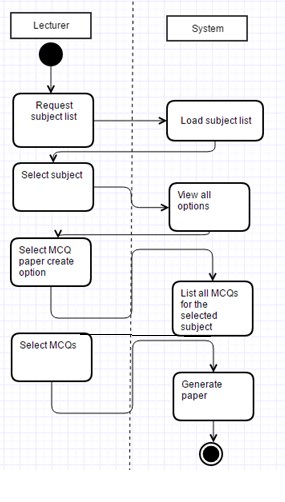
### Lecturer Submits new MCQ



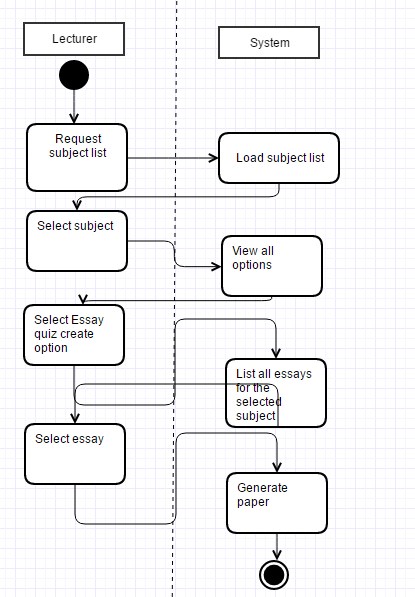
### Lecturer Submits new essay



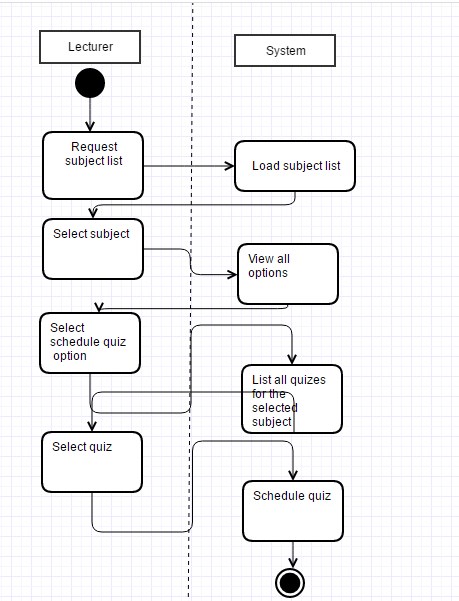
### Lecturer generates MCQ paper



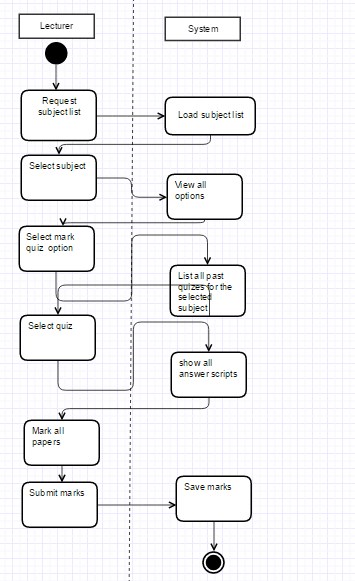
### Lecturer generates essay paper



### Lecturer schedules a paper



### Lecturer marks essay paper

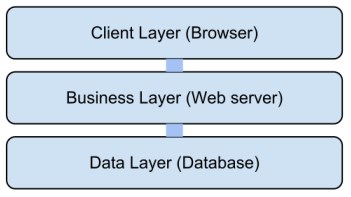


# Deployment View

Being a web application this system is hosted in a remote server, (initially HEOKU hosting space). The database will be hosted in some other hosting space. Except for the statistical graph generation, all the processing is done in the backend, so that the client computer doesn’t have to spend much of CPU power. In case of statistical graphs generation, all the data processing will be done in the front end, so that the client computer will need a particular level of performance.

# Implementation View

## Overview



OES (online examination system) is a web application that follows the MVC architecture pattern. Main reason to use this pattern is to separate functions into layers thus improve the maintainability and reusability.

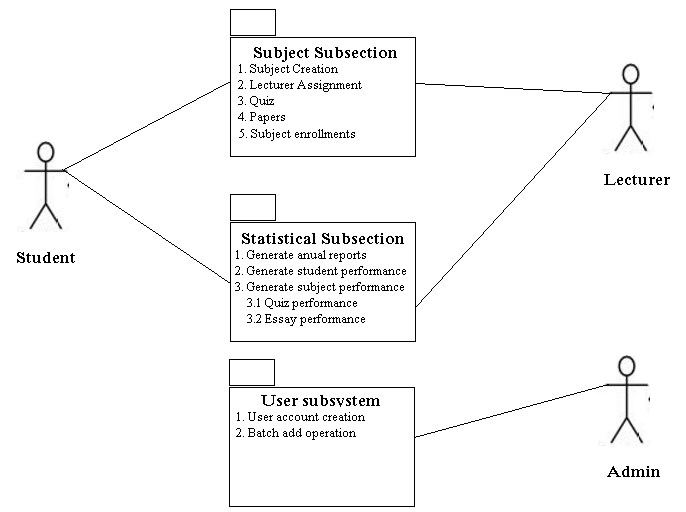
View layer (AKA client layer) contains the graphical user interfaces (web pages). The actions of these web pages are handled by controller classes. Controller classes invokes and instantiates objects of model classes that contains the business login. Separating software packages in this manner reduces the complexity.

The model classes can be subdivided in to two layers. Business layer and the data access layer. Data access layer is manipulated using HIBERNATE framework. Business layer contains the main entity classes such as Student, Lecturer and etc.

The object relational mapping is done using the HIBERNATE framework. For each entity in the database there exists a class in the software that provides data access operations to that entity.

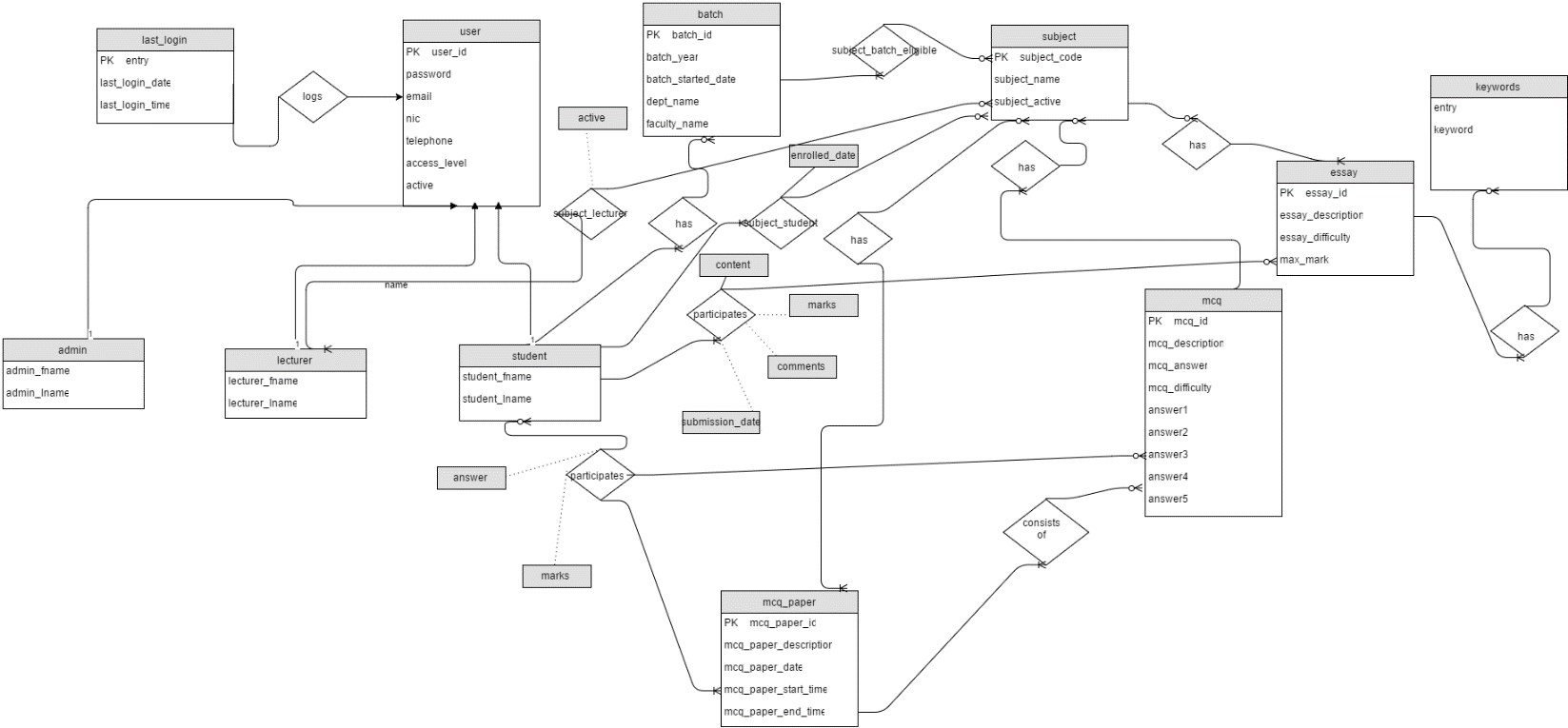
## Layers

As mentioned above the software will be developed following the MVC pattern. For each sub layer, these 3 layers apply. Following is the representation of main subsystems of the system and their relevant main use cases.



# Data View

## ER diagram



Since the diagram is not very clear it’s available in the following link

<https://drive.google.com/file/d/0B_key68-co6IdWJpQk02cDE2TWs/view?usp=sharing>

## MYSQL implementation

Main design concentrations of this ER diagram is mentioned above under the section “Logical view”. The database implementation for this ER diagram is done using MYSQL. The main tables are

1. Admin
2. Batch
3. Essay
4. Essay-Student
5. Keywords
6. Last-Login
7. Lecturer
8. MCQ
9. MCQ-paper
10. MCQ-paper-map
11. MCQ-paper-student
12. Student
13. Subject
14. Subject-batch-eligible
15. Subject-lecturer
16. Subject-student
17. User

Hibernate framework is used to automate the object relational mapping. From that a class is made for each table in the database, that makes is possible to access the database.

# Size and Performance

Being a web application, OES will be hosted in the free web hosting space HEROKU.com. Users will not have to install any component to use this software system.

The size of the software is still not calculated, but it is in the range 100-200 MB. The MYSQL database will be hosted in another hosting space.

The client computers need to have a web browser in order to access the system. All the functionalities will be processed at the backend, except for the statistical report generation. Statistical reports will be generated in the front end, thus the process requires some computational power from the client computer. Performance characteristics are mentioned in the section 3.6.

# Quality

Architectural goals and constrains topic covers the quality parameters for the OES (online examination system).