3/19/2017

SOFTWARE DESIGN SPECIFICATION

FOR A Learning Management System with social functionality

Abdelrahman Mamdouh

Mark Wafik

Abdelrahman Mohamed

Mohamed Magdy

Moatsem Salah

VERSION: REVISION – 1.0

## CONTENTS

1. [INTRODUCTION 6](#_bookmark0)
   1. [Purpose 6](#_bookmark0)
   2. [Scope 6](#_bookmark0)
   3. [Objective 6](#_bookmark0)
2. [SYSTEM OVERVIEW 7](#_TOC_250007)
   1. [Product Perspective 7](#_TOC_250006)
      1. Design Method 7
      2. [User Interfaces 7](#_bookmark1)
      3. [Hardware Interfaces 7](#_bookmark1)
      4. [Software Interfaces 7](#_bookmark1)
      5. [Memory Constraints 7](#_bookmark1)
      6. [Operations 7](#_bookmark1)
      7. [Site Adaptation Requirements 7](#_bookmark1)
   2. [Product Functions 8](#_TOC_250005)
   3. [User Characteristics 8](#_TOC_250004)
   4. [Constraints 8](#_bookmark2)
   5. [Assumptions and Dependencies 8](#_bookmark2)
   6. [Apportioning of Requirements 8](#_bookmark2)
3. [DESIGN CONSIDERATIONS 9](#_TOC_250003)
   1. [Operating Environment 9](#_TOC_250002)
   2. [Fault Tolerant Design 9](#_TOC_250001)
   3. [Design Conventions 9](#_TOC_250000)
   4. [Architectural Design 9](#_bookmark3)
   5. [User Interface: 9](#_bookmark3)
      1. [Expected Input: 9](#_bookmark3)

|  |  |
| --- | --- |
| 3.5.2 Output: | 10 |
| [**4. SYSTEM ARCHITECTURE**](#_bookmark4) | [**111**](#_bookmark4) |
| [**4.1 View of Product Classes (Figure 1)**](#_bookmark4) | [**111**](#_bookmark4) |
| [**4.2 Individual Classes of System**](#_bookmark5) | [**122**](#_bookmark5) |
| [4.2.1 Airport:](#_bookmark5) | [12](#_bookmark5) |
| [4.2.1.1 Attributes of Airport class](#_bookmark5) | [122](#_bookmark5) |
| [4.2.1.1.1 takeoffQueue](#_bookmark5) | [122](#_bookmark5) |
| [4.2.1.1.2 landingQueue](#_bookmark5) | [122](#_bookmark5) |
| [4.2.1.2 Functions available in Airport Class](#_bookmark5) | [122](#_bookmark5) |
| [4.2.1.2.1 Averager ( )](#_bookmark5) | [122](#_bookmark5) |
| [4.2.1.2.2 newArrival ( )](#_bookmark5) | [122](#_bookmark5) |
| [4.2.2 Airplane](#_bookmark5) | [122](#_bookmark5) |
| 4.2.2.1 Functions available in Airplane Class | 12 |
| 4.2.2.2.1 takeoff ( ) | 12 |
| [4.2.2.2.2 land ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.2.2.3 crash ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.3 Runway](#_bookmark6) | [133](#_bookmark6) |
| [4.2.3.1 Functions available in Runway Class](#_bookmark6) | [133](#_bookmark6) |
| [4.2.3.1.1 isBusy ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.3.1.2 moveToRunway ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.3.1.3 timeleftonRunway ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.4 Queue](#_bookmark6) | [133](#_bookmark6) |
| [4.2.4.1 Functions available in Queue class](#_bookmark6) | [133](#_bookmark6) |
| [4.2.4.1.1 size ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.4.1.2 Capacity ( )](#_bookmark6) | [133](#_bookmark6) |
| [4.2.4.1.3 insert ( )](#_bookmark7) | [144](#_bookmark7) |
| [4.2.4.1.4 front ( )](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5 Averager](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.1 Attributes of Averager Class](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.1.1 count](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.1.2 sum](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.1 Functions available in Averager Class](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.2.1 next\_number ( )](#_bookmark7) | [144](#_bookmark7) |
| [4.2.5.2.2 average ( )](#_bookmark7) | [144](#_bookmark7) |
| [4.2.6 NextArrival](#_bookmark7) | [144](#_bookmark7) |
| [4.2.6.1 Attributes of NextArrival Class](#_bookmark7) | [144](#_bookmark7) |
| [4.2.6.1.1 aver\_wait\_time](#_bookmark7) | [144](#_bookmark7) |
| [4.2.6.2 Functions available in NextArrival Class](#_bookmark7) | [144](#_bookmark7) |
| [4.2.6.2.1 wait\_time ( )](#_bookmark7) | [144](#_bookmark7) |

|  |  |
| --- | --- |
| [**5. FIGURES**](#_bookmark8) | [**155**](#_bookmark8) |
| [**5.1 Use Cases (Figure 2)**](#_bookmark8) | [**155**](#_bookmark8) |
| [**5.2 Successful Takeoff**](#_bookmark9) | [**166**](#_bookmark9) |
| [5.2.1 Successful Takeoff Sequence (Figure 3)](#_bookmark9) | [166](#_bookmark9) |
| [5.2.2 Successful Takeoff Collaboration (Figure 4)](#_bookmark10) | [177](#_bookmark10) |
| [**5.3 Successful Landing**](#_bookmark11) | [**188**](#_bookmark11) |
| [5.3.1 Successful Landing Sequence (Figure 5)](#_bookmark11) | [188](#_bookmark11) |
| [5.3.2 Successful Landing Collaboration (Figure 6)](#_bookmark12) | [199](#_bookmark12) |
| **5.4 Runway Busy** | **20** |
| [5.4.1 Runway Busy Sequence (Figure 7)](#_bookmark13) | [20](#_bookmark13) |
| [5.4.2 Runway Busy Collaboration (Figure 8)](#_bookmark14) | [211](#_bookmark14) |
| [**5.5 Crash**](#_bookmark15) | [**222**](#_bookmark15) |
| [5.5.1 Crash Sequence (Figure 9)](#_bookmark15) | [222](#_bookmark15) |
| [5.5.2 Crash Collaboration (Figure 10)](#_bookmark16) | [233](#_bookmark16) |
| [**5.6 Print Report**](#_bookmark17) | [**244**](#_bookmark17) |
| [5.6.1 Print Report Sequence (Figure 11)](#_bookmark17) | [244](#_bookmark17) |
| [5.6.3 Print Report Collaboration (Figure 12)](#_bookmark18) | [255](#_bookmark18) |
| [**6. REFERENCES**](#_bookmark19) | [**266**](#_bookmark19) |
| [**6.1 References**](#_bookmark19) | [**266**](#_bookmark19) |

## INTRODUCTION

* 1. **Purpose**

This document will define the design of the Learning Management System with social features . It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

.

# Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement the Learning Management System with social features

# Objective

The “My University” is a LMS mobile application which helps student to stay connect with every thing related to their university . started from their courses and social media groups to their university buses locations .The application should be free to download from either a mobile phone application store or

similar services.

Teacher assistant can provides grades , assignments and materials using the web-portal and check for assignment plagiarism online between other student submissions and over the internet. the students deliver their assignments . view their grades and download their materials and track their buses .student

can register their courses Online .and receive notification for every important announcement.

## SYSTEM OVERVIEW

.

# Product Perspective

This system will consist of two parts: one mobile application and one web portal. The mobile application will be used to make the student able to connect easily to their grades ,assignements and even their buses locations while the web portal will be used for managing

the information about the restaurants and the system as a whole.

The product is supposed to be an open source, under the GNU general Public License. It is a cross platform mobile application with client-server model. The "My University"

will provide the proper methods to make all the possible communication methods for all students with their colleagues and the university staff.

.

**2.1.1 Design Method**

The design of this product utilizes an object-oriented approach.

* + 1. **User Interfaces**

The user of this software product will be interfacing with the simulation system to help predict the behavior an actual air traffic control system. The product allows the user to get familiar with the software without actually having the responsibility of controlling the air traffic

* + 1. **Hardware Interfaces**

Since neither the mobile application nor the web portal have any designated hardware, it does not have any direct hardware interfaces. and the hardware connection to the database server is managed by the underlying operating system on the mobile phone and the web server.

* + 1. **Software Interfaces**

The mobile application communicates with the database in order to get the information about the students, see Figure 1. The communication between the database and the web portal consists of operation concerning both reading and modifying the data, while the communication between the database and the mobile application consists of only reading operations.

.

* + 1. **Memory Constraints**

The amount of Operate System memory occupied by the application. It must not to exceed 20 MB during the testing but we think that it will be around 16 MB but our most wishes is 10 MB

* + 1. **Site Adaptation Requirements**

This software is intended to execute on any platform with no modifications needed to support different sites.

# Product Functions

a Learning Management System with web portal for the university staff and mobile application for the students . the staff will be able to set assignments , quizzes and grades they can send announcements to all students . beside every assignments submitted from a student there will be 2 percentages for plagiarism the first is a compare between all the assignments that is submitted from all the students and the other comparing the submitted assignments will the resources on the Internet the percentage is a percentage of similarity and they can post public and anonymous surveys for the students . the student get an announcement for the quizzes , assignments and grades . they can download materials and submit assignments only before deadlines . they can chat with each others and with the staff with private message but with permission from the receiver first . and as public message as a post . all the students is divided automatically by courses , groups , sections and individuals so every action on the system can be done according to this groups .

# User Characteristics

our user characteristics is kind of dynamic it's build on permissions so every set of permissions make a new user role but basically they will come around two roles

the most of our users will be two types of users that interact with the system: users of the mobile application : students, and university staff. Each of these two types of users has different use of the system so each of them has their own requirements.

The mobile application users (students) can use the application to check the quizzes , assignments and grades . they can download materials and submit assignments only before deadlines . they can chat with each other’s and with the staff with private message but with permission from the receiver first . and as public message as a post .

the staff will be able to set assignments , quizzes and grades they can send announcements to all students . beside every assignments submitted from a student there will be 2 percentages for plagiarism the first is a compare between all the assignments that is submitted from all the students and the other comparing the submitted assignments will the resources on the Internet the percentage is a percentage of similarity and

they can post public and anonymous surveys for the students .

# Constraints

The system is constrained by the system of every university because different universities have different routines even the students have different experiences so the sys will have to be customized for universities

The Internet connection is also a constraint for the application. Since the application fetches data from the database over the Internet, it is crucial that there is an Internet connection for the application to function.

Both the web portal and the mobile application will be constrained by the capacity of the database. Since the database is shared between both application it may be forced to queue incoming requests and therefor increase the time it takes to fetch data.

# Assumptions and Dependencies

One assumption about the product is that it will always be used on mobile phones that have enough performance. If the phone does not have enough hardware resources available for the application, for example the users might have allocated them with other applications, there may be scenarios where the

application does not work as intended or even at all.

# Apportioning of Requirements

In the case that the project is delayed, there are some requirements that could be transferred to the next version of the application. Those requirements are to be developed in the third release

## DESIGN CONSIDERATIONS

* 1. **Operating Environment**

The system server side will work on windows or any Linux distribution.

# Fault Tolerant Design

Application errors will be handled by common fault detection services ( e.g. common php exception handling, and error checking on task processing).

# Design Conventions

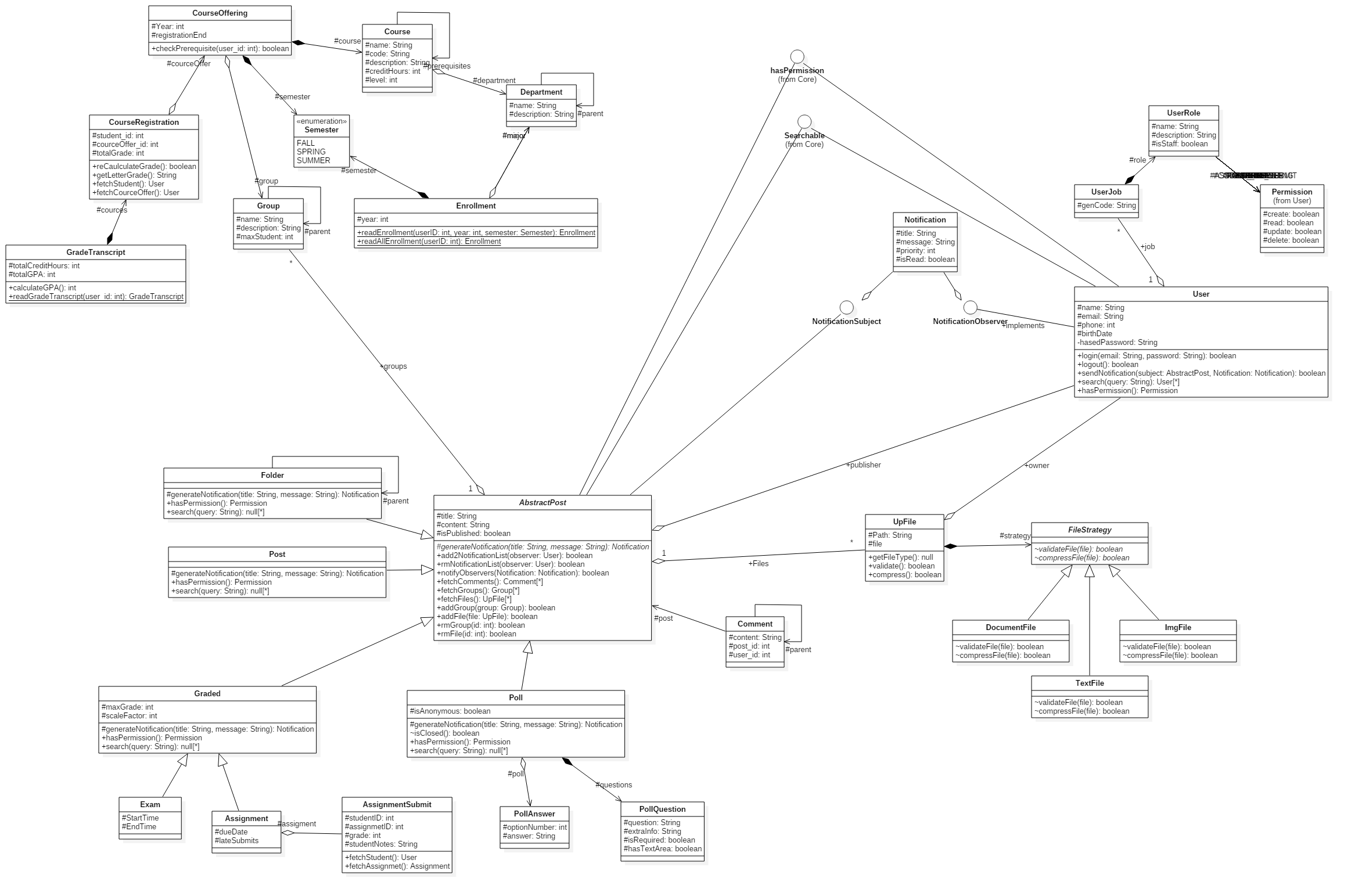
the Learning Management System with social features uses the Object Oriented methodology described in "The Unified Software Development Process" by Ivar Jacobsen, Grady Booch and James Rumbaugh. (Booch, 1999)

# Architectural Design

The software capabilities and requirements specified in the Learning Management System with social features Software Requirements Specification are transformed into programs that will execute on a Linux server. Software items are partitioned into classes and packages using Object Oriented methodology to maximize encapsulation and minimize interfaces. Packages.

## SYSTEM ARCHITECTURE

* 1. **View of Product Classes (Figure 1)**



# Individual Classes of System

* + 1. **AssignmentController:**

Can edit anything related to assignments .

#### Functions available in AssignmentController Class

* + - * 1. **createAssigment()**

Function: start new assignment.

* + - * 1. **updateAssigment()**

Function: change assignment info.

* + - * 1. **deleteAssigment()**

Function: delete assignment info.

* + - * 1. **submitAssigmentSolution()**

Function: submit assignment from student.

* + - * 1. **updateAssigmentSolution()**

Function: change assignment submission.

* + 1. **CourseController**

Can edit anything related to courses .

#### Functions available in CourseController Class

Function: start new course.

* + 1. **PollConstroller**

Can crud anything related to polls .

#### Functions available in PollController Class

* + - * 1. **submitAnswers()**

Function: vote in polls

* + - * 1. **modifyAnswers()**

Function: change vote in polls

* + - * 1. **createPoll()**

Function: add new poll

* + - * 1. **updatePoll()**

Function: update poll question or answer .

* + - * 1. **deletePoll()**

Function: delete poll

* + 1. **PostController**

Can crud anything related to polls .

#### Functions available in PostController class

* + - * 1. **deletePost()**

Function: delete posts .

* + - * 1. **createPost()**

Function: add new post.

* + - * 1. **updatePost()**

Function: change post details.

* + 1. **UserController**

Can crud anything related to users .

#### 4.2.5.1 Functions available in UserController Class

* + - * 1. **login()**

Function: check user info.

* + - * 1. **logout()**

Function: end user session

* + - * 1. **signup()**

Function: add new user.

* + 1. **DocumentFile**

* + - 1. **Functions available in** **DocumentFile** Class
         1. **validateFile(File file)**

Function: call the strategy file to check each file if it belongs to the proper category

* + - * 1. **compressFile**

Function: compress the file to save disk space and improve the performance

* + 1. **FileStrategy**

abstract file strategy class to handle different files types

* + - 1. **Functions available in** **FileStrategy** Class
         1. **validateFile(File file)**

Function: *check if file is valid*

* + - * 1. **compressFile(File file)**

Function: compress the file to save disk space and improve the performance

* + 1. **ImgFile**
       1. **Functions available in** **ImgFile** Class
          1. **validateFile(File file)**

Function: call the strategy file to check each file if it belongs to the proper category

* + - * 1. **compressFile**

Function: compress the file to save disk space and improve the performance

* + - 1. **Functions available in TextFile** Class
         1. **validateFile(File file)**

Function: call the strategy file to check each file if it belongs to the proper category

* + - * 1. **compressFile**

Function: compress the file to save disk space and improve the performance

* + 1. **UpFile**
       1. **Attributes available in UpFile** Class

String Path;

File location.

File file;

File object

User UploaderBy;

The user that uploaded the file

Type;

The file type (picture – compressed - …………)

* + - 1. **Functions available in UpFile** Class
         1. **validateFile(File file)**

Function: call the strategy file to check each file if it belongs to the proper category

* + - * 1. **compressFile**

Function: compress the file to save disk space and improve the performance

* + - * 1. **getFileType()**

Function: return the file type.

* + 1. **Notification**

The notification that will be sent to user to alert him if there is an important news

* + - 1. **Attributes available in** **Notification** Class

String title, message;

The notification message and the title of it.

User sender, recipient;

The news sender and the that the news are sent to .

int priority;

the priority of the news to be sent or not

boolean isRead;

is the user opened the notification or not.

* + 1. **Assignment**

The assignment object will be made from this class

* + - 1. **Attributes available in** **Assignment** Class

Date dueDate, lateSubmits;

The assignments due dates and late submit date .

* + 1. **AssignmentSubmit**

a solution for an assignment submitted by a student

* + - 1. **Attributes available in** **AssignmentSubmit** Class

int studentID, assignmetID;

the student id that submitted the assignment and which assignment he is submitting

int grade;

the grade that the student got on the assignment.

String studentNotes;

The notes on the assignment that a student have .

UpFile[] files;

Array of files that will be uploaded

* + - 1. **Functions available in** **AssignmentSubmit** Class
         1. **fetchStudent()**

*get the student that will submit the student*

* + - * 1. **fetchAssignmet()**

Function: *get the assignment that will be submitted*

* + 1. **Comment**

a comment for a post

it also support hierarchy ( different comment levels)

* + - 1. **Attributes available in** **Comment** Class

Comment parent;

The parent comment

String content;

The content of the comment

User user;

The user that made the comment

* + 1. **Exam**

An exam for students

* + - 1. **Attributes available in** **Exam** Class

Date StartTime, EndTime;

The start date and the end date of an exam

* + 1. **Folder**

Folder to contain the materials

* + - 1. **Attributes available in** **Folder** Class

CourseOffering course;

The course that the material will be for.

* + - 1. **Functions available in** **Folder** Class
         1. **generateNotification(String title, String message)**

Function: **get notification if the folder is upatede.**

* + - * 1. **hasPermission()**

Function: **check the permisions.**

* + - * 1. **search(String query)**

Function: **search in the folder**

* + 1. **Graded**

used to record student grades for example an exam, quiz .. etc

* + - 1. **Attributes available in** **Graded** Class

CourseOffering cource;

The course that the grades will be for.

int maxGrade, scaleFactor;

the maximum grade and if there are scaling

* + - 1. **Functions available in** **Folder** Class
         1. **generateNotification(String title, String message)**

Function: **get notification if the folder is upatede.**

* + - * 1. **hasPermission()**

*Function: check the permisions.*

* + - * 1. **search(String query)**

*Function: search in the folder*

* + 1. **Poll**
       1. **Functions available in** **Poll** Class

boolean isAnonymous;

check if the poll is anonymous or not

PollQuestion[] questions;

Array of the questions in the poll

* + - 1. **Functions available in** **poll** Class
         1. **generateNotification(String title, String message)**

Function: make notification when poll is set

* + - * 1. **isClosed()**

Function: know of the poll is ended or not

* + - * 1. **hasPermission()**

Function: check if the user have a permission or not

* + - * 1. **search(String query)**

Function: search in the polls

* + 1. **AbstractPost**
       1. **Attributes available in** **AbstractPost** Class

User creator;

String title, content;

UpFile[] files;

boolean isPublished;

* + - 1. **Functions available in** **AbstractPost** Class
         1. **generateNotification(String title, String message)**

Function: generate notification to be send

* + - * 1. **add2NotificationList(User observer)**

Function: add the generated notification to the notification list for a specific user.

* + - * 1. **rmNotificationList(User observer)**

Function: remove the generated notification to the notification list for a specific user.

* + - * 1. **notifyObservers(Notification Notification)**

Function: send the notification.

* + - * 1. **fetchComments()**

Function: get the comments for a post.

* + - * 1. **fetchGroups()**

Function: get the list of groups.

* + - * 1. **fetchFiles()**

Function: get list of files.

* + - * 1. **addGroup(Group group)**

Function: add new group.

* + - * 1. **addFile(UpFile file)**

Function: add new file.

* + - * 1. **rmGroup(int id)**

Function: remove group.

* + - * 1. **rmFile(int id)**

Function: remove files.

* + 1. **PollAnswer**
       1. **Attributes available in** **PollAnswer** Class

Poll poll;

Poll object.

User user;

User object.

int optionNumber;

the number of answers of a poll.

String answer;

The answer of a poll.

* + 1. **PollQuestion**
       1. **Functions available in** **PollQuestion** Class

String question, extraInfo;

The poll questions and if the question have extra information

Choice[] choices;

Array of choices

boolean isRequired, hasTextArea;

if the poll have an essay question.

* + 1. **Post**
       1. **Functions available in** **Post** Class
          1. **generateNotification(String title, String message)**

Function: **get notification if the folder is upatede.**

* + - * 1. **hasPermission()**

Function: **check the permisions.**

* + - * 1. **search(String query)**

Function: **search in the folder**

* + 1. **Course**

**a passive representation of a course**

* + - 1. **Attributes available in** **Course** Class

String name, code, description;

The name of the course and the course code the description of the course

Course[] prerequisites;

Array of prerequisites of every course

int creditHours;

the number of credit hours of every course

int level;

which level is the course in

Department department;

In which department the course in

* + 1. **CourseOffering**

an instance of a course for each semester/year

* + - 1. **Attributes available in** **CourseOffering** Class

Course course;

Which course will be opened .

int Year;

the year which the course will be opened

Semester semester;

the semester which the course will be opened

User[] staff;

The array of staff that working on the course

Date registrationEnd;

The time that the registration will end .

Group group;

The group that the course will be opened for.

* + - 1. **Functions available in** **CourseOffering** Class
         1. **checkPrerequisite(int user\_id)**

Function: check the prerequisite of a course

* + 1. **CourseRegistration**
    2. **Attributes available in** **CourseRegistration** Class

int student\_id, courceOffer\_id, totalGrade;

the student id and if the course is opened and the total grade of the course .

* + - 1. **Functions available in** **CourseRegistration** Class
         1. **reCaulculateGrade()**

Function: recalculate the GPA from the graded items

* + - * 1. **getLetterGrade()**

Function: get the letter grade representing the grade

* + - * 1. **fetchStudent()**

Function: fetch the student from the db by id

* + - * 1. **fetchCourceOffer()**

Function: fetch the course offer the db by id

* + 1. **Department**

Department can be Faculty of CS or English Department ...etc

* + - 1. **Attributes available in** **Department**Class

Department parent;

The parent department of the department

String name;

The name of the department

String description;

The description of the departement

UpFile LogoImg;

The logo of the course

* + 1. **Enrollment**

hold enrollment data for each term

* + - 1. **Attributes available in** **Enrollment** Class

Department major, minor;

If the the departement is a minor or a major

int year;

the year of the enrollment

Semester semester;

The semester of the enrollment

* + - 1. **Functions available in** **Enrollment** Class
         1. **readEnrollment(int userID, int year, Semester semester)**

Function: fetch enrollment

* + - * 1. **Enrollment readAllEnrollment(int userID)**

Function: fetch all the enrollment related to this user

* + 1. **GradeTranscript**

calculate the user gpa & completed credit hours from this Courses Registration

* + - 1. **Attributes available in** **GradeTranscript** Class

CourseRegistration[] cources;

All the courses that a student has registered .

int totalCreditHours, totalGPA;

the total credit hour that a student registered . and the total gpa for this hours

* + - 1. **Functions available in** **GradeTranscript** Class
         1. **calculateGPA()**

Function: calculate the user GPA

* + - * 1. **readGradeTranscript(int user\_id)**

Function: fetch the Grade transcript from DB by user ID

* + 1. **Group**

sections or groups ex: Group A , Section A3 would have A as parent

* + - 1. **Attributes available in** **Group** Class

String name, description;

The name and the description of a group .

int maxStudent;

maximum number of students in agroup

User[] students;

List of students in the group

* + 1. **Semester**

enum for semester names

* + - 1. **Attributes available in** **Semester** Class

FALL, SPRING, SUMMER

Which semester is this .

* + 1. **User**

Containing the user object

* + - 1. **Attributes available in** **User** Class

*String name, email;*

*The name and the email of the student*

*int universityID, phone;*

*the university id and the phone number of the students*

*Date birthDate;*

*The birth date of student*

*Enrollment[] enrollments;*

*The semester that he enrolled in*

*private String hasedPassword;*

*the password of the tudent*

* + - 1. **Functions available in** **User** Class
         1. **login(String email, String password)**

Function: check user email and password

if correct start a new session and set this object data to the users

* + - * 1. **logout()**

Function: reset the session

* + 1. **UserJob**

*The job of the user either a staff member or student*

* + - 1. **Attributes available in** **UserJob** Class

Department department;

The departement of the user

UserRole role;

The role of the user

* + 1. **UserRole**

Have the roles of the user

* + - 1. **Attributes available in** **UserRole** Class

String name,

The name of the role

description;

the description of the role

boolean isStaff;

if the user a staff or not

* + 1. **Permission**

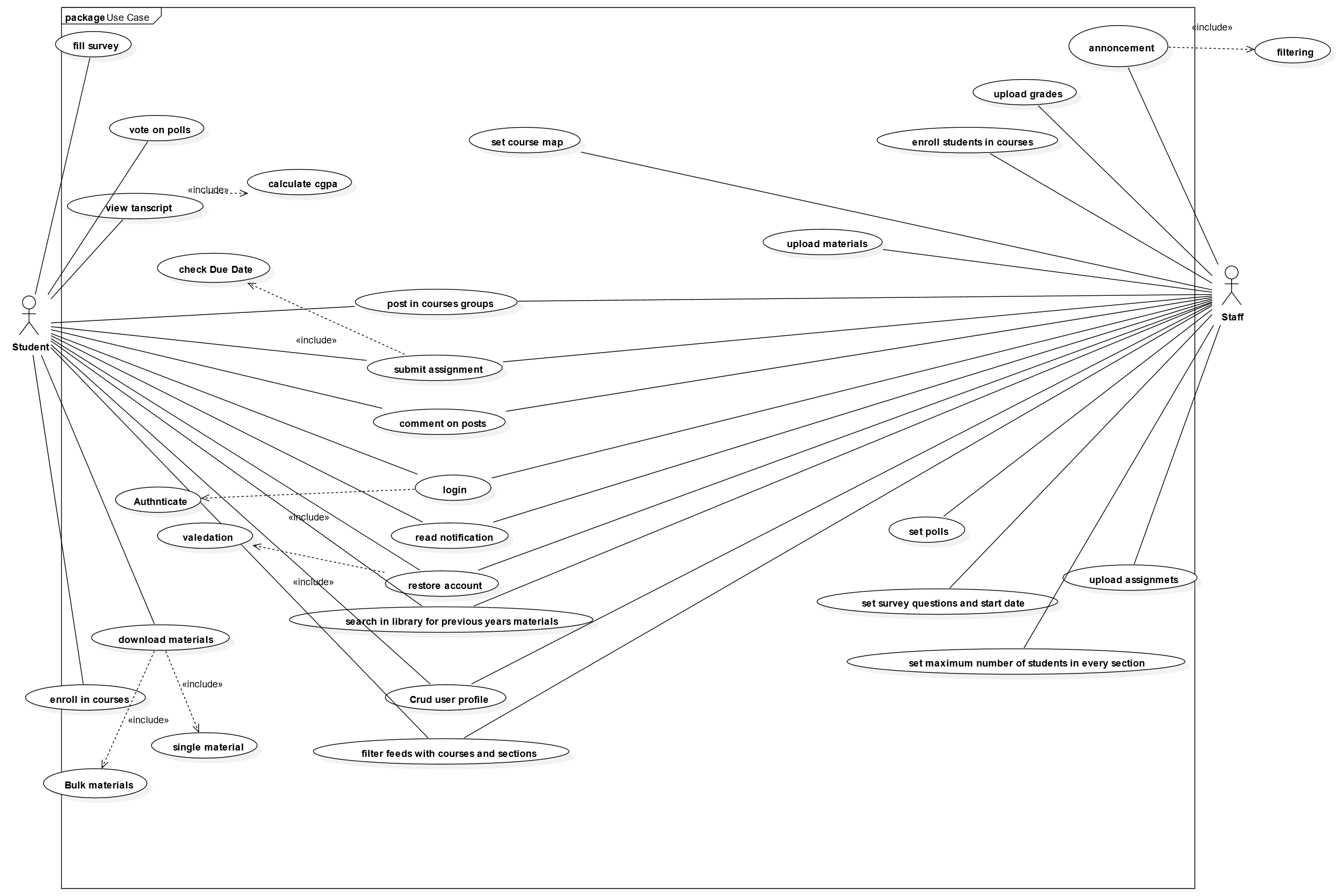
The permissions of every user

* + - 1. **Attributes available in** **Permission** Class

boolean create, read, update, delete;

check of a user has the permission of create , read , update or delete

1. **FIGURES**
   1. **Use Cases (Figure 2)**

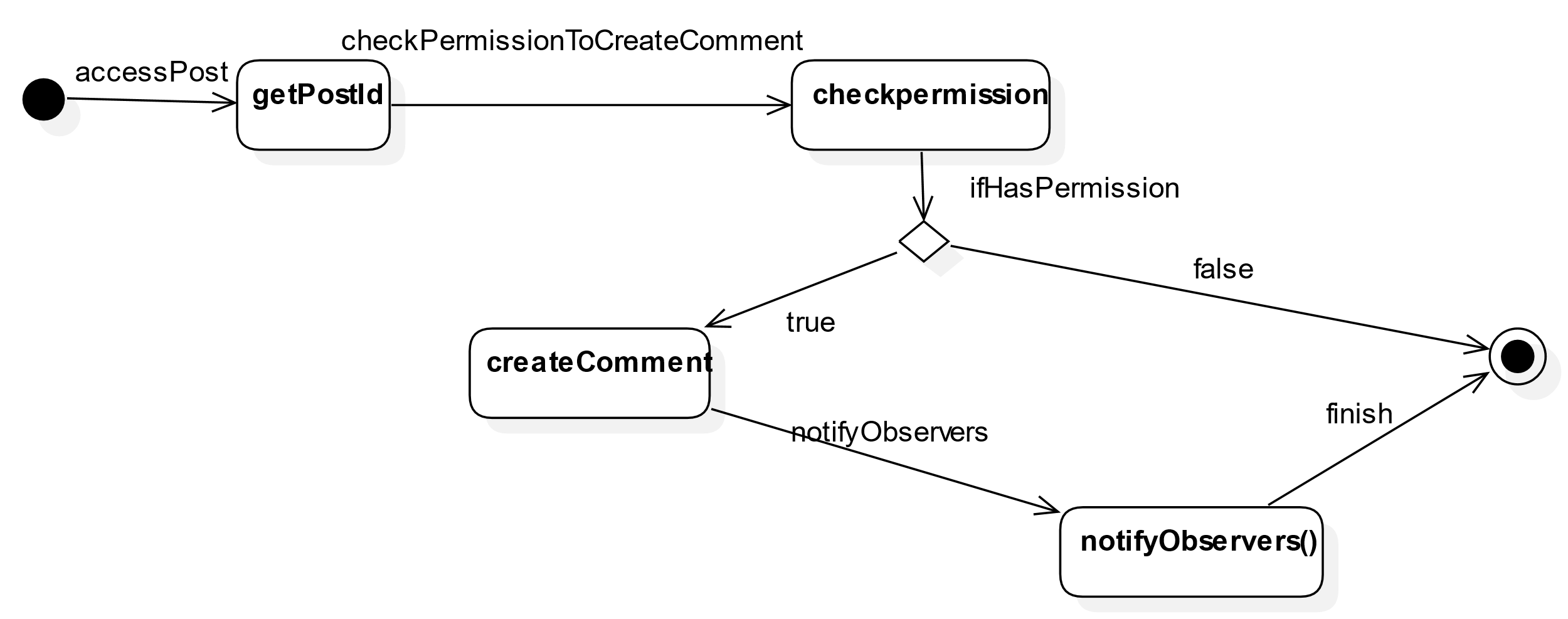
––+

* 1. **comments on posts**

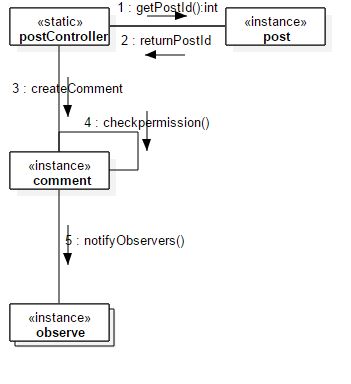
TakeoffRequest : NextArrival

|  |  |  |
| --- | --- | --- |
| TakeoffQ : Queue | : Runway | recordTakeoff : Averager |

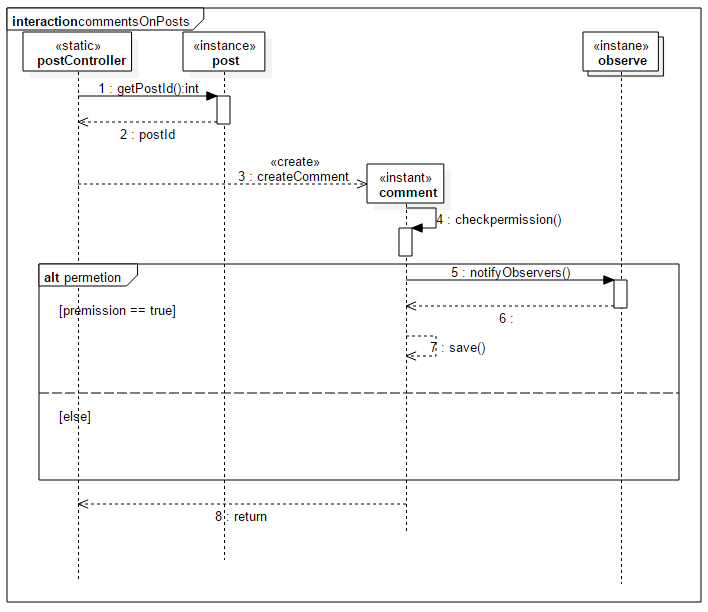
**5.2.1 activity comments on posts**

****

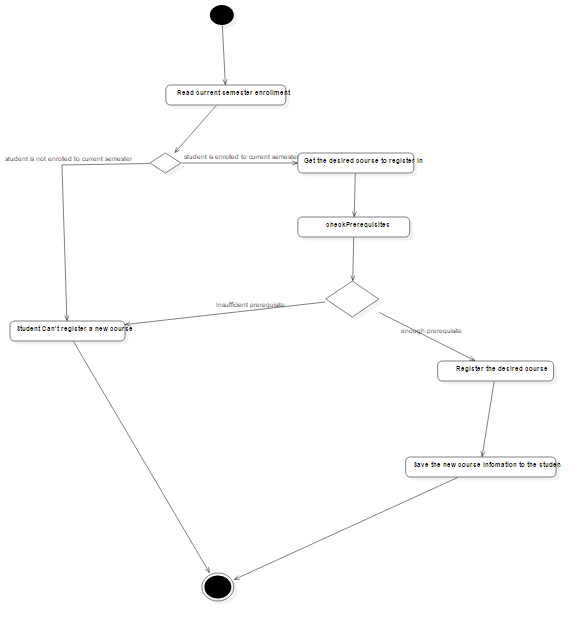
**5.2.2 collaboration comments on posts**

****

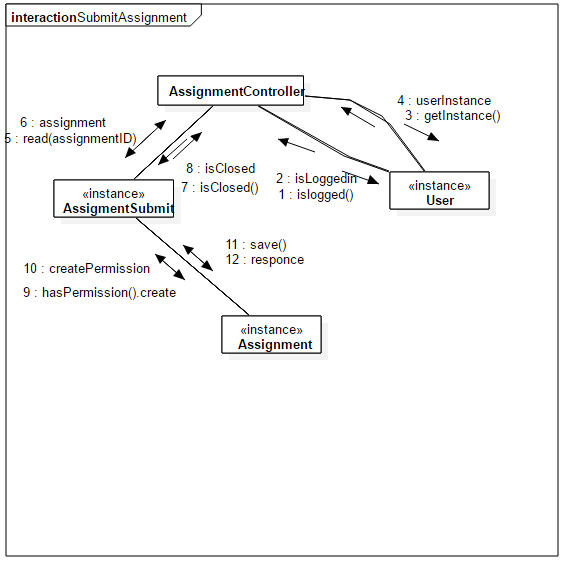
5.2.3 sequence comment on posts



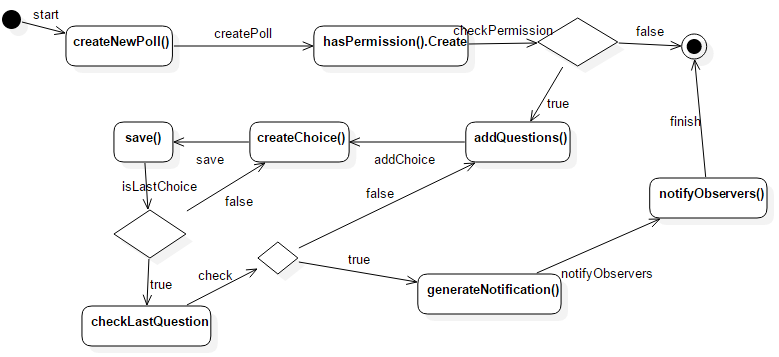
* 1. enroll in courses
     1. activity enroll in courses



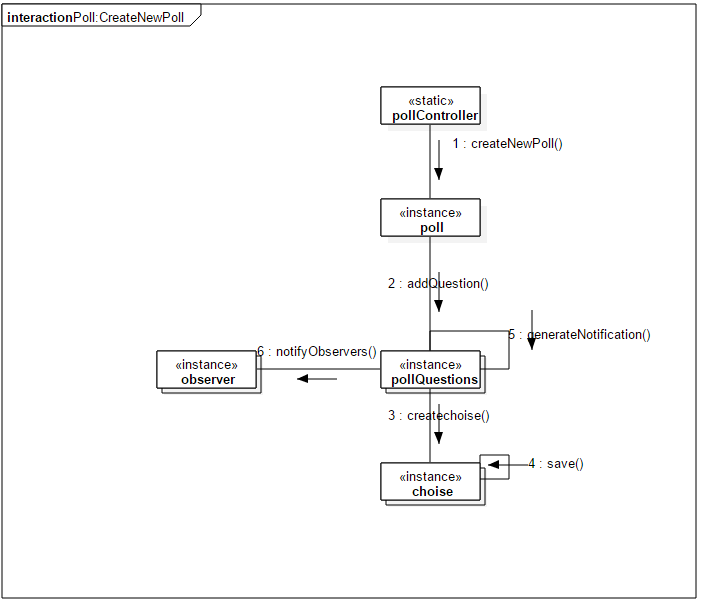
* + 1. collaboration enroll in courses



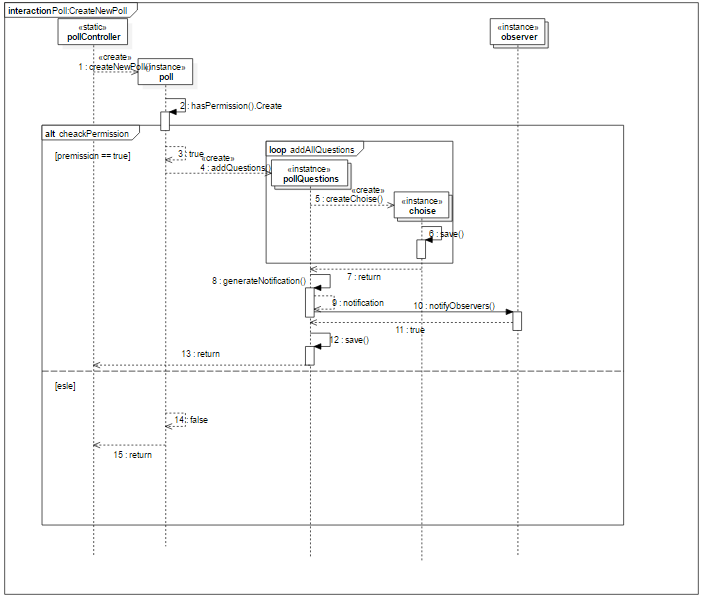
* 1. create poll
     1. activity create poll



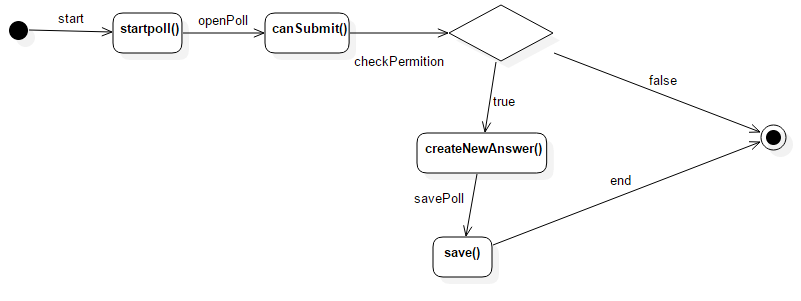
* + 1. collaboration create poll



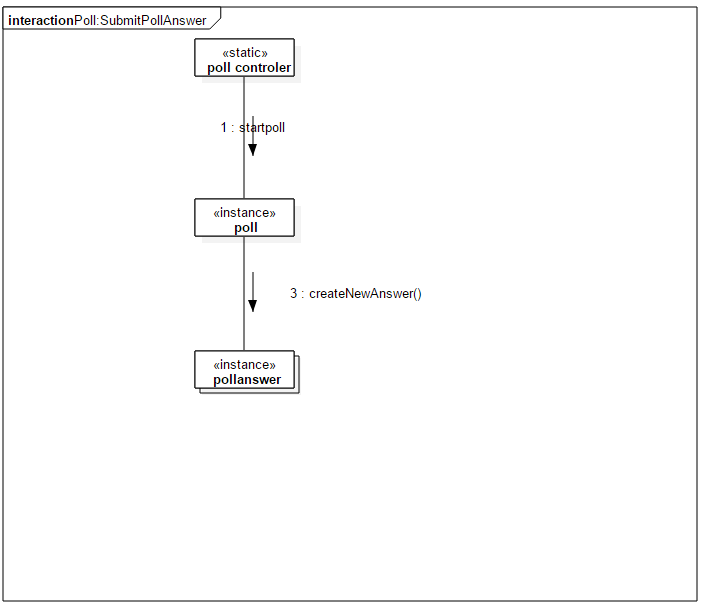
* + 1. sequence create poll



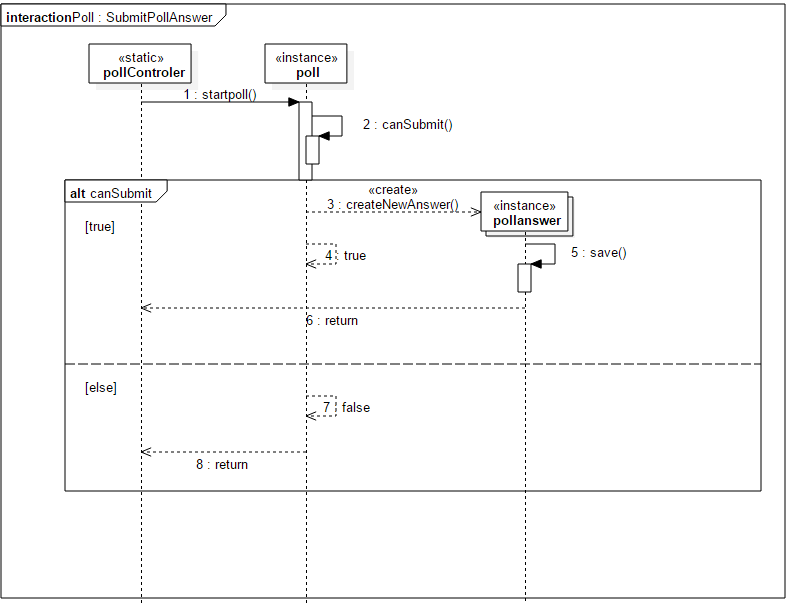
* 1. submit poll
     1. activity submit poll



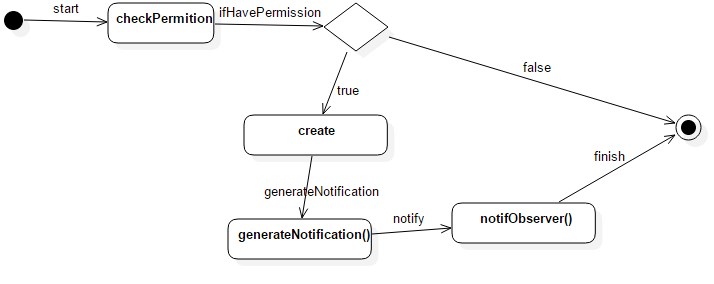
* + 1. collaboration submit poll



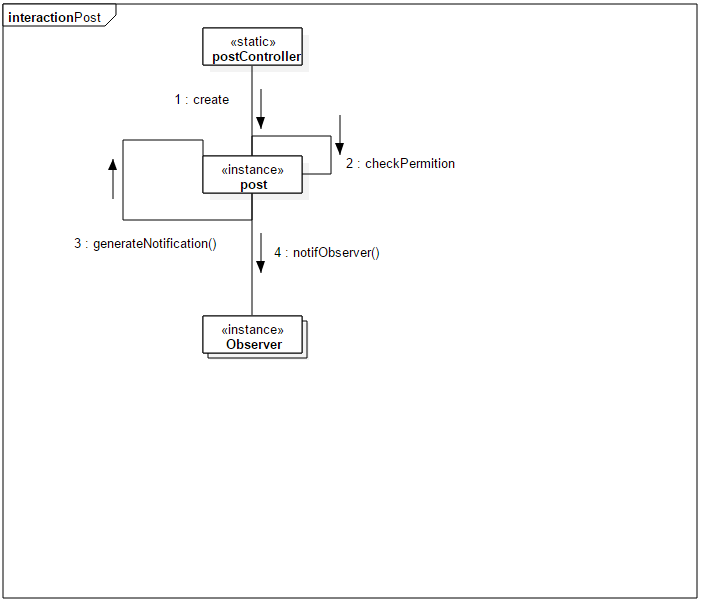
* + 1. sequence submit poll



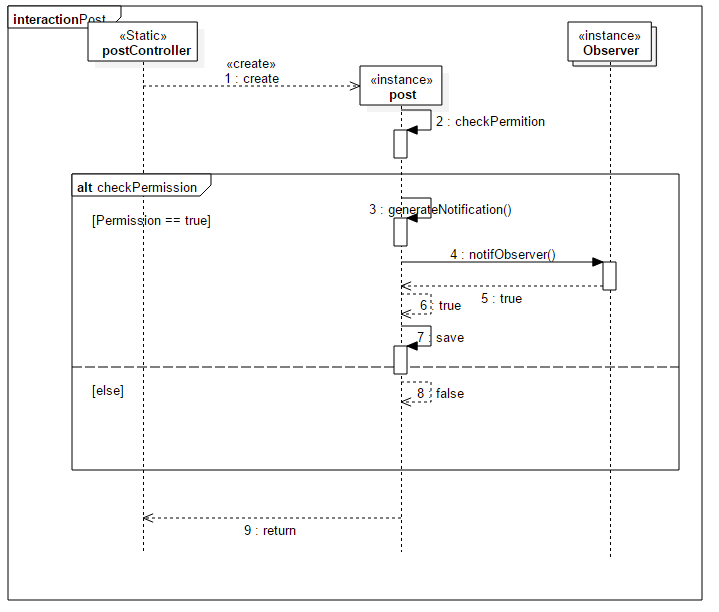
* 1. post
     1. activity post



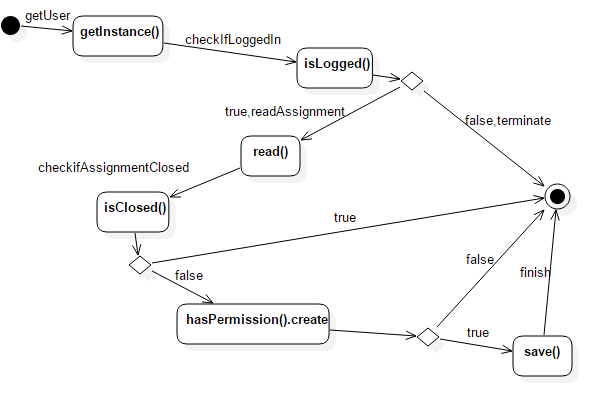
* + 1. collaboration post



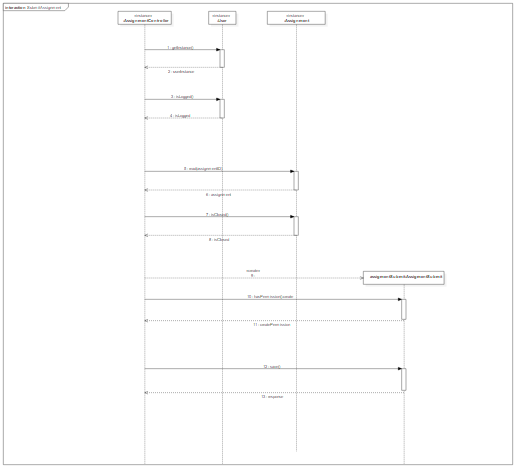
* + 1. sequence post



* 1. submit assignment
     1. activity submit assignment



* + 1. sequence submit assignment



1. **REFERENCES**
   1. **References**

Appleton, Brad . A Software Design Specification Template. N.d.

[<http://www.enteract.com/~bradapp/docs/sdd.html>.](http://www.enteract.com/%7Ebradapp/docs/sdd.html)

Booch, Grady, Ivar Jacobsen, and James Rumbaugh. The Unified Software Development Process (The Addison-Wesley Object Technology Series). 1st. Ed. New York: Addison Wesley, 1999..

GCC Home Page - GNU Project . Free Software Foundation. N.d. [<http://gcc.gnu.org/>.](http://gcc.gnu.org/) Sommerville, Ian. Software Engineering. 6th. Ed. New York: Addison Wesley, 2001.

|  |  |
| --- | --- |
| Names | contributors |
| Abdelrahman Mamdouh | First contributor SRS, Use Case Diagram and SDD |
| Mark Wafik William | First Contributor Class Diagram, SDD  second in activity and sequence diagram |
| Abdelrahman Mohamed Abdelhameed | first contributor in sequence diagram and SDD  Second Contributor in class, activity and collaboration diagram |
| Mohamed Magdy | First Contributor activity and collaboration and SDD  second in sequence diagram |
| Moatsem Salah | Second contributor in use case diagram, SRS, SDD |