Final Year Project Management System

Software Requirements Specification

Version 2.5

28th April 2020

Behjet Ansari

Eraj Rizwan

Laiba Fatima Khan

Mehreen Shafiuddin

Shiza Khalidi

Software Engineers

Prepared for

CS 353—Software Engineering

Instructor: Umair Azfar Khan.

Spring 2020

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 28th March 2020 | Version 1.0 |  |  |
| 7th April 2020 | Version 1.1 |  |  |
| 9th April 2020 | Version 1.5 |  |  |
| 23rd April 2020 | Version 2.0 |  |  |
| 26th April 2020 | Version 2.5 |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
| (Approved via email) | Dr. Musabbir Majeed | Stakeholder | 13th April 2020 (version 1.0) |
|  | Dr. Musabbir Majeed | Stakeholder |  |
|  |  |  |  |

# Table of Contents

[**Revision History**](#_heading=h.xj0x0016uq7j) **2**

[**Document Approval**](#_heading=h.6v2soj7iib5j) **2**

[**Table of Contents**](#_heading=h.b5emd326oy9h) **2**

**1.** [**Introduction**](#_heading=h.aw8wfmunb1i6) **1**

[1.1 Purpose](#_heading=h.3znysh7) 1

[1.2 Scope](#_heading=h.2f2vawewv9eo) 1

[1.3 Definitions, Acronyms, and Abbreviations](#_heading=h.b3pmc1m666vy) 2

[1.4 References](#_heading=h.cx3qsuzhzpj1) 3

[1.5 Overview](#_heading=h.cam6zwzd9znz) 3

[**2. General Description**](#_heading=h.hb8fcxr9r4vr) **3**

[2.1 Product Perspective](#_heading=h.hm4iba8qxotd) 3

[2.2 Product Functions](#_heading=h.w13hwjtk7m10) 4

[2.3 User Characteristics](#_heading=h.hodkhgi2nio5) 6

[2.4 General Constraints](#_heading=h.dy8yt2lli0cg) 7

[2.5 Assumptions and Dependencies](#_heading=h.fb5xmfrjazsz) 7

[**3. Requirements**](#_heading=h.bd6qn491uy4s) **8**

[3.1 External Interface Requirements](#_heading=h.bqk4y03j13u3) 8

[3.1.1 User Interfaces](#_heading=h.44sinio) 8

[3.1.2 Hardware Interfaces](#_heading=h.m9vfixw5k69m) 10

[3.1.3 Software Interfaces](#_heading=h.2ezpf03c4eao) 10

[3.1.4 Communications Interfaces](#_heading=h.j9teiez9qijr) 10

[3.2 Functional Requirements](#_heading=h.5jvqjimw4tm7) 11

[3.2.1 Submitting Project Member Names](#_heading=h.6pevout3060k) 11

[3.2.2 Making Groups](#_heading=h.q1tf1klckq8) 11

[3.2.3 Submitting the Proposal](#_heading=h.dwq6lqla02qh) 12

[3.2.4 Send Proposals for Approval](#_heading=h.rw2amnfwr8ij) 12

[3.2.5 Assigning Supervisors](#_heading=h.f3roa4yykm5a) 13

[3.2.6 Accepting Request from KWG](#_heading=h.yedel7srap8c) 13

[3.2.7 Sending Reminders](#_heading=h.e4mu5iidqeon) 14

[3.2.8 Assigning a Grading Panel](#_heading=h.3ssr5t7tb90g) 14

[3.2.9 Sharing Custom Gradebook](#_heading=h.2swv9o46l2vf) 15

[3.2.10 First Evaluation](#_heading=h.756rramk8x9s) 15

[3.2.11 Peer Evaluations](#_heading=h.nuxq1wlrzukg) 16

[3.2.12 Sending Final Grades](#_heading=h.fshp31xwx0by) 16

[3.2.13 Compiling Gradebook](#_heading=h.oqtgekyy5dm4) 17

[3.2.14 Creating Calendar Event](#_heading=h.1l961p61aka1) 17

[3.2.15 Add, Edit, and Delete Group Members](#_heading=h.ypfnohna2oau) 18

[3.2.16 Dashboard Entries and Notes](#_heading=h.d9iu0ptpivha) 19

[3.2.17 Filter Search](#_heading=h.g6rtdyd285jk) 19

[3.2.18 Notes Features](#_heading=h.gkg2h33elp8b) 20

[3.2.19 Adding New Site Tool](#_heading=h.gzslukohrj53) 20

[3.3 Use-cases](#_heading=h.791gbjaruhnb) 22

[3.3.1 Use-case #1](#_heading=h.6l35iar1f3sh) 22

[3.3.2 Use-case #2](#_heading=h.snhmnobwfneu) 24

[3.3.3 Use-case #3](#_heading=h.sdgh0g6xu3tn) 26

[3.3.4 Use-case #4](#_heading=h.lu47u05b26zr) 29

[3.3.5 Use-case #5](#_heading=h.cmmfuchbbjcr) 31

[3.3.6 Use-case #6](#_heading=h.rmyjqxm040we) 34

[3.3.7 Use-case #7](#_heading=h.u34tc4u1wj9w) 38

[3.3.8 Use-case #8](#_heading=h.st11v87rkgr4) 41

[3.3.9 Use-case #9](#_heading=h.eeuo6c1gpcj9) 43

[3.3.10 Use-case #10](#_heading=h.vj4hxzg7l0ug) 46

[3.4 Classes / Objects](#_heading=h.gjdgxs) 52

[3.4.1 User](#_heading=h.z0z2i3sj8xq1) 56

[3.4.2 CS\_Students: User](#_heading=h.ota9dtbqpsu2) 57

[3.4.3 StudentGroup](#_heading=h.snvmmrgj8rgh) 57

[3.4.4 CS\_Faculty: User](#_heading=h.jy23ilhlblnw) 57

[3.4.5 KWG\_Member: CS\_Faculty](#_heading=h.hof5kls62xwn) 58

[3.4.6 Supervisor: CS\_Faculty](#_heading=h.h2olv850h4s9) 58

[3.4.7 GradingPanel](#_heading=h.fhsymj2lhzg6) 59

[3.4.8 DashboardMember](#_heading=h.nqencdtkgeir) 59

[3.4.9 LMS](#_heading=h.pkb87ymebo97) 59

[3.4.10 SiteTool](#_heading=h.bqro5db9xqv1) 60

[3.4.11 Site](#_heading=h.16wbxyotrwg8) 60

[3.4.12 OverviewTool : SiteTool](#_heading=h.pasdivyxg8wl) 60

[3.4.13 SiteInfoTool: SiteTool](#_heading=h.w7834ucan3j5) 61

[3.4.14 TasksTool: SiteTool](#_heading=h.j5h2ddud7n50) 61

[3.4.15 ResourcesTool: SiteTool](#_heading=h.rgieweo8xi6z) 61

[3.4.16 AnnouncementsTool: SiteTool](#_heading=h.8fjsbwcwcxt2) 62

[3.4.17 GradebookTool: SiteTool](#_heading=h.u6nrnm9qfvvu) 62

[3.4.18 RosterTool: SiteTool](#_heading=h.ze96ajbr0hl4) 62

[3.4.19 CalendarTool: SiteTool](#_heading=h.asjbtprbiibc) 63

[3.4.20 DashboardTool: SiteTool](#_heading=h.bkymjl7ffo7d) 63

[3.4.21 Task](#_heading=h.2wdf61f5plt3) 63

[3.4.22 Announcement](#_heading=h.6zyib4h5xy33) 64

[3.4.23 Resource](#_heading=h.ssnlrxvbxccd) 64

[3.4.24 Gradebook](#_heading=h.yrz0mitxragy) 64

[3.4.25 Note](#_heading=h.hazdbdzhsioc) 65

[3.4.26 Entry](#_heading=h.lpqj3i9skuby) 65

[3.4.27 Event](#_heading=h.ssqvswqyfkq2) 66

[3.5 Non-Functional Requirements](#_heading=h.4gqcwo5fmz9u) 66

[3.5.1 Performance](#_heading=h.2p2csry) 66

[3.5.2 Reliability](#_heading=h.far6uz7jkv9z) 66

[3.5.3 Availability](#_heading=h.ifhlky2pn23x) 66

[3.5.4 Security](#_heading=h.5hs8efoiul4v) 66

[3.5.5 Maintainability](#_heading=h.tdvf5fyffz3b) 67

[3.5.6 Portability](#_heading=h.hg9ly9mniuvv) 67

[3.6 Design Constraints](#_heading=h.x0u12unldib3) 67

[3.7 Logical Database Requirements](#_heading=h.1gi81vnnncn3) 67

[3.8 Other Requirements](#_heading=h.6zhmndlms1in) 67

[**4. Analysis Models**](#_heading=h.lxl6wit9dt08) **67**

[4.1 Sequence Diagrams](#_heading=h.2pvu8cf6m9qp) 67

[**Appendices**](#_heading=h.z109fn7hs9zd) **72**

[A.1 Appendix 1](#_heading=h.9ww9vrp3c2g6) 72

[A.2 Appendix 2](#_heading=h.jshvr9w15he8) 72

# Introduction

## 1.1 Purpose

The purpose of this document is to provide guidelines necessary to design and build a Final Year Project Management System for the Kaavish Committee at Habib University. The committee is responsible for guiding and evaluating senior CS undergraduate students on their final year projects.

The software will ease the flow of information between the students and the faculty. It will enable students to submit their work, view their progress and receive feedback from their supervisor and grading panel, and allow the committee to control the overall system with ease of communication with students and external faculty.

This document illustrates the complete purpose for the development of the system and it includes system constraints, requirements and interfaces intended to be proposed to the stakeholder, Dr. Musabbir Majeed, for its approval, and development of the software by our team.

## 1.2 Scope

This document covers the FYP Management System. It will be a web-based application that will be an extension of the existing Sakai Learning Management System (LMS) version 12.5. It is supposed to be a convenient and easy-to-use application for students, Kaavish Committee and other faculty involved in the process of managing the final year projects.

This software will provide ease of communication and flow of information to the Kaavish Committee, as well as a way for students to share and reflect on their progress. It will be a tool to help supervisors be aware of the development phases of the project and to provide relevant feedback and advice. It will also help the members of the grading panel in providing their feedback.

The application should be accessible via IDs provided by the institution. It will not be accessible by all students and faculty, only the ones who are directly involved with the FYP management and control.

This ultimate goal of this application is to integrate the components of different applications, such as LMS and Outlook, that are currently being used to manage the entire process. It will be useful for users and Kaavish Committee members, who will also have their own interfaces (Section 3.3.1). This will make it easy for our stakeholder to organize and oversee the system. It will provide a way for the students to collaborate with their group and update their progress from time to time. Records will be maintained using a database at the backend that can be viewed by the Kaavish Committee members.

Sakai LMS 12.5 supports the following functionalities that will be used in the FYP management system:

Logging in and out of account, Creating new sites, adding participants to site, changing participant roles, Creating groups of site participants, enabling group submissions.

The following site tools are supported by LMS 12.5:

Announcements, Gradebook, Resources, Calendar, Roster.

The following features will be added:

**Tasks** site tool, to enhance the submission process by introducing the option to add automated messages and to share the submitted task with other site participants.

**Dashboard** site tool for keeping track of FYP progress and enhancing communication between student group and supervisor.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| FYP | Final Year Project |
| SRS | Software Requirement Specification |
| LMS | Learning Management System |
| Users | Students or faculty members who will be using this software. |
| Student Group | A group of 3-4 students who are using this software for submitting their work and viewing progress and results. |
| Supervisor | Faculty member who is responsible for supervising a group for their FYP. |
| Grading Panel | A group of faculty members responsible for grading students and using the software for releasing their final feedback and grades. |
| Kaavish Working Group | A committee consisting of faculty members who use this software as administrators. They are responsible for controlling and managing the system. |
| KWG | Kaavish Working Group |
| CS | Computer Science |

## 1.4 References

|  |  |
| --- | --- |
| [1] | K. Weigers, 20 March 2006, "Process Impact,". [Online]. Available: www.processimpact.com. [Accessed 25 3 2020]. |
| [2] | D. P. Mirenda, 02 Oct 2006, "2Communicate SRS". [Online]. Available: https://lostclouds.com/2Communicate/project/SRS.pdf. [Accessed 25 03 2020]. |
| [3] | A. Athuraliya, 12 June 2019, “Sequence Diagram Tutorial: Complete Guide with Examples.” *Creately Blog*, creately.com/blog/diagrams/sequence-diagram-tutorial/. |
| [4] | *Sakai Community Documentation*, sakai.screenstepslive.com/s/sakai\_help. |

## 1.5 Overview

This SRS is divided into 5 components. Beginning with the introduction, it follows the description of the software and then moves on to specific requirements in the next section, followed by analysis models based on those requirements. The last section consists of Appendices.

This section provides the basic information related to the SRS document and the proposed software.

Section 2 i.e. General Descriptions explains the functionality of the software, types of users, user interactions and the assumptions involved in developing this software.

Section 3 i.e. Specific Requirements contains descriptions of all requirements- functional and non-functional, use-cases and system features.

Section 4 i.e. Analysis Models features models that describe the specification of certain requirements.

The last section is Appendices, that contains all additional information related to the SRS.

# 2. General Description

## 2.1 Product Perspective

The FYP management system that is being developed, is a replacement for the existing tools used for FYP management at Habib University. Currently, LMS is used among other tools for the coordination of different activities between Kaavish committee, supervisors and students. The new system will configure the existing system, as well as integrate more tools to build a system that will allow the Kaavish committee, project supervisors and project members to monitor, contribute and communicate effectively. This system focuses on the development of integrated online submission, that connects each group’s work with the respective supervisor, grading panel and the Kaavish committee. The system also keeps track of individual contribution and progress.

## 2.2 Product Functions

**From student’s perspective:**

Log in

Log out

Open announcements tab

Open Tasks tab

Open Gradebook tab

Open Calendar tab

View Calendar events

View Dashboard FYP progress

Edit Dashboard

Complete and submit tasks

View gradebook

**From supervisor’s perspective:**

Log in

Log out

Open Tasks tab

Complete and submit tasks

Open Dashboard tab

View Dashboard for FYP progress

Edit Dashboard

Write feedback and grades for gradebook as task

Open announcements tab

Write announcements

Open resources

View grading rubrics and other uploaded resources

Open Calendar tab

View Calendar events

**From Kaavish committee perspective:**

Log in

Log out

Create Kaavish site

Add participants to site

Change participants’ role

Add site tools

Open announcements tab

Create announcements

Open Tasks tab

Create and send tasks

View task submissions

Forward tasks to selected site participants

Create groups

Add site participants to groups

Open calendar tab

Create calendar events

Open Dashboard tab

View projects’ progress on Dashboard

Create custom gradebooks

Fill gradebooks with feedback and grades

Upload gradebooks

Open resources tab

Upload grading rubrics and other resources

**From grading panel’s perspective:**

Log in

Log out

Open announcements tab

View announcements

Write and submit feedback and grades for gradebook as task

Write announcements

Open resources

View grading rubrics and other uploaded resources

Open Calendar tab

View Calendar events

## 2.3 User Characteristics

**Kaavish committee:**

The Kaavish committee members have the greatest access and permissions.

Group option is used by the KWG member to group the supervisor with their respective student groups. This allows the supervisor to keep track of their students’ progress and make assessments. The KWG members can also monitor the work progress for all the students and supervisors in a particular group through the Dashboard tool. This allows them to send reminders personally to those who have not adhered to any deadlines.

In addition, KWG members can also group the faculty members in the grading panel and provide them with all of the resources and information that they need to assess the FYP group.

**Students:**

Students are responsible to extract and read all the information uploaded by the KWG to ensure that they remain updated with the latest information. They also need to make the required submissions as tasks, and update their supervisor with the project's progress through the Dashboard tool. They are expected to discuss with their supervisors and make the necessary amendments based on the feedback.

**Supervisor:**

The supervisor is responsible to view the weekly progress made by the student group and add their feedback to the entries in the dashboard. They are required to set weekly meetings with the group to guide them and provide them with necessary guidance.

Once appropriate progress has been made by the student group, they are supposed to provide their feedback and assessment grades based on the rubric prepared by the KWG.

**Grading Panel:**

The grading panel members will provide their feedback and assessment grades based on the rubric prepared by the KWG. The students will not be able to see which faculty member has given what feedback and grades.

## 2.4 General Constraints

The existing system on LMS needs to be configured. A new system should not be made from scratch, as it will be difficult to introduce a new system for faculty and students in their last year, and the KWG members will have to justify the costs for the new system.

Since the Learning Management System (SAKAI) 12.5 version is in place at the university, it should be used, no other LMS can be used.

## 2.5 Assumptions and Dependencies

The users should know English language as the user interface will be provided in English.

The campus should have internet facilities to access LMS.

LMS accounts should be already made for all users.

The system has access to the University’s student and faculty database.

Kaavish committee members, students and supervisors are all familiar with the working of LMS.

Each project will consist of 3-4 students and 1 supervisor.

Modifications can be made to the existing LMS.

# 3. Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The software is suitable for both web and mobile, although most of the interaction is done on the web version. The web interface is as follows:

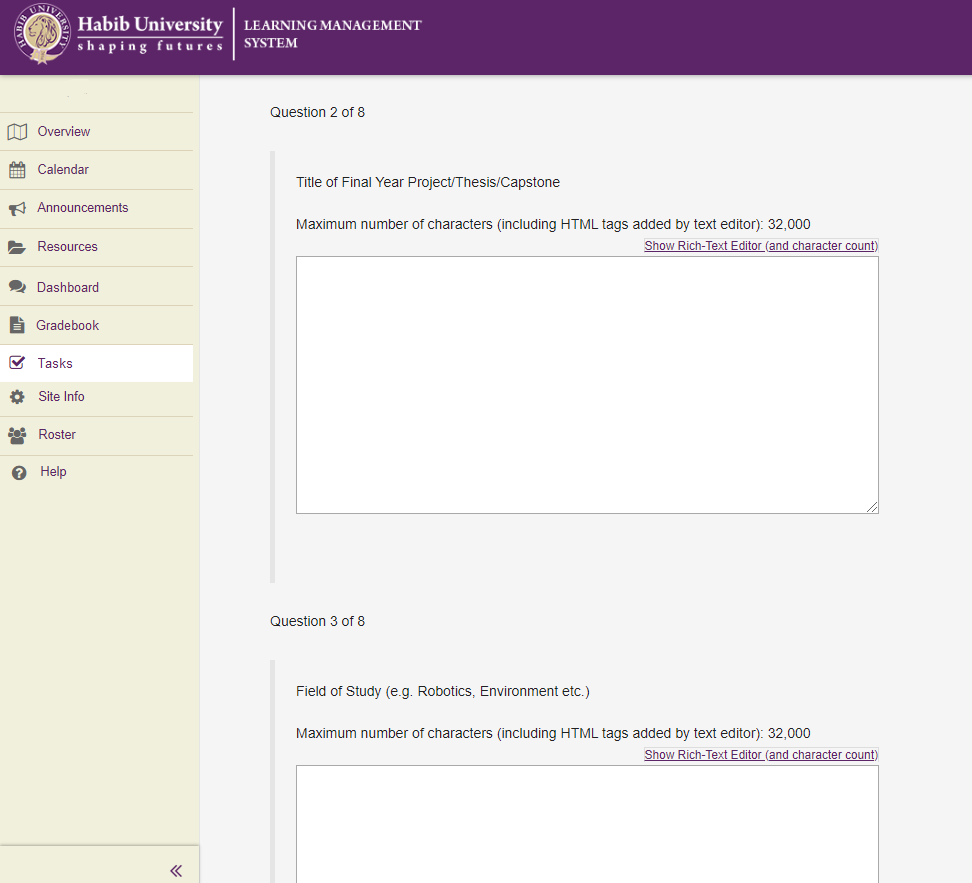


Figure 1. Proposals would be sent through the Tasks tab on LMS. Once all questions are answered, the user would click on “Submit”.

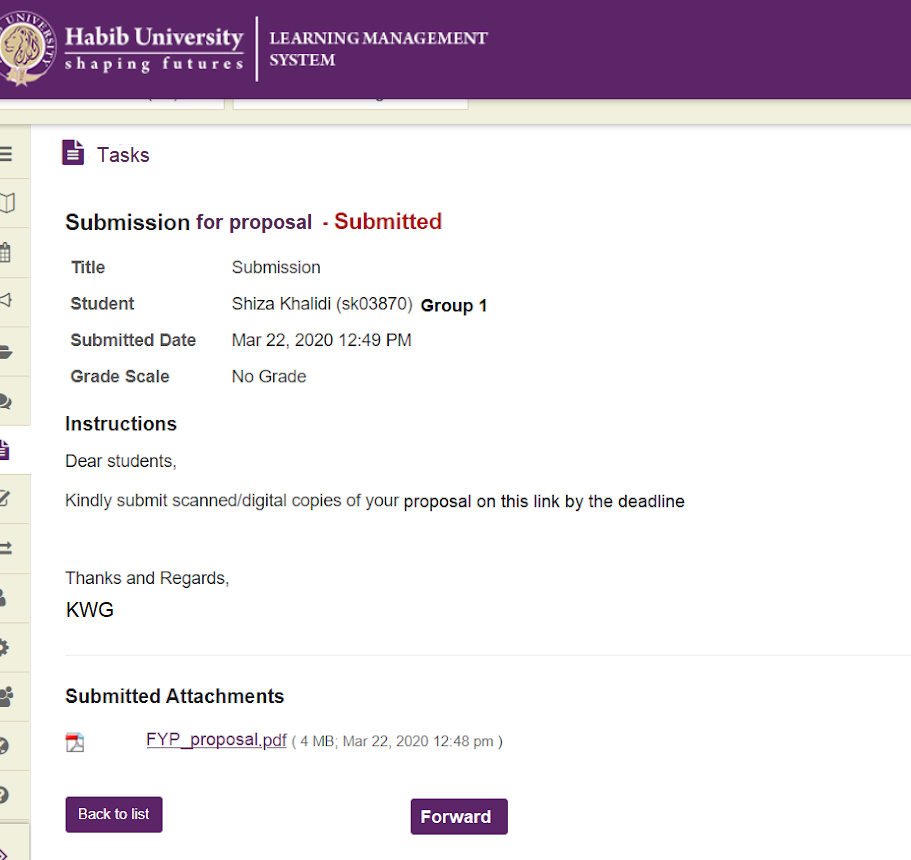


Figure 2. Once the KWG receives the project proposal, it can forward it to any faculty member by clicking ‘Forward’.

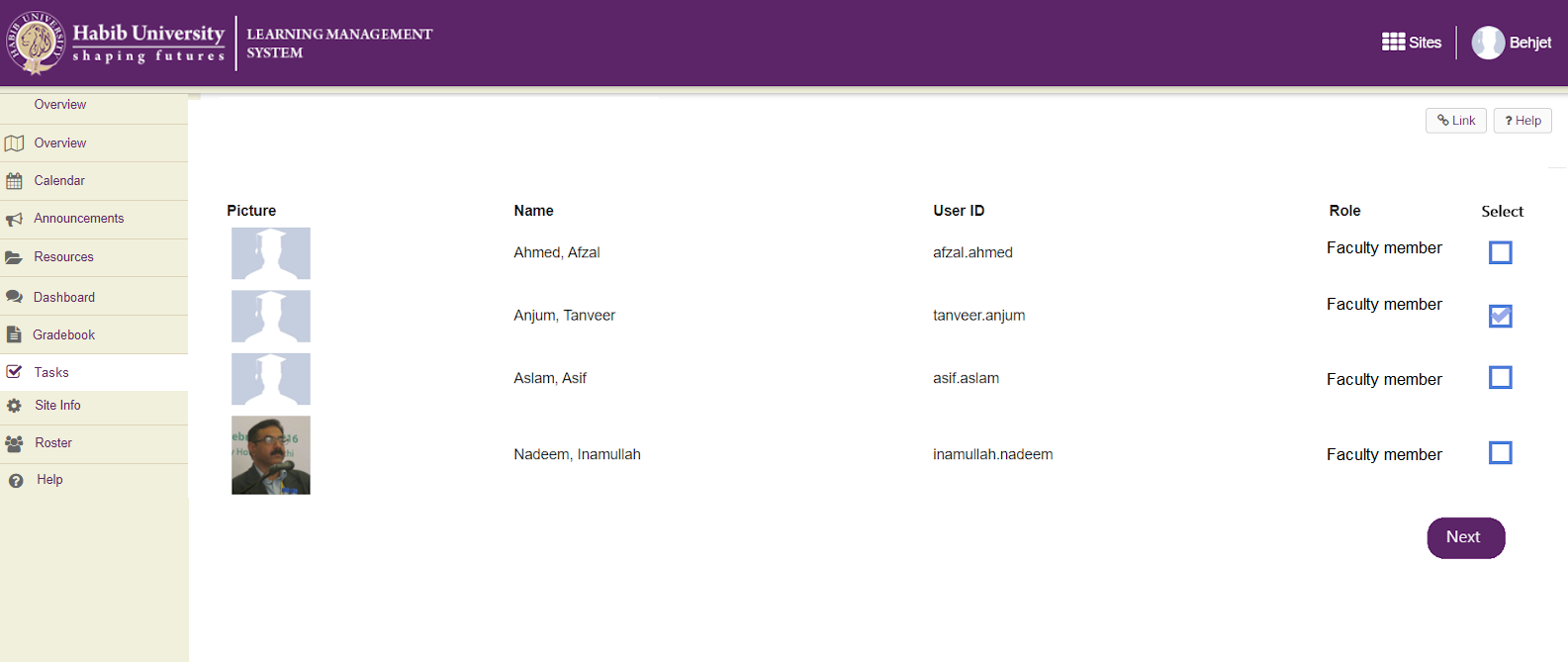


Figure 3. Supervisors can be selected from a pool of faculty by KWG. By clicking on the “Next” button, KWG can send the proposal along with an automated message requesting faculty member to accept or reject the proposal.

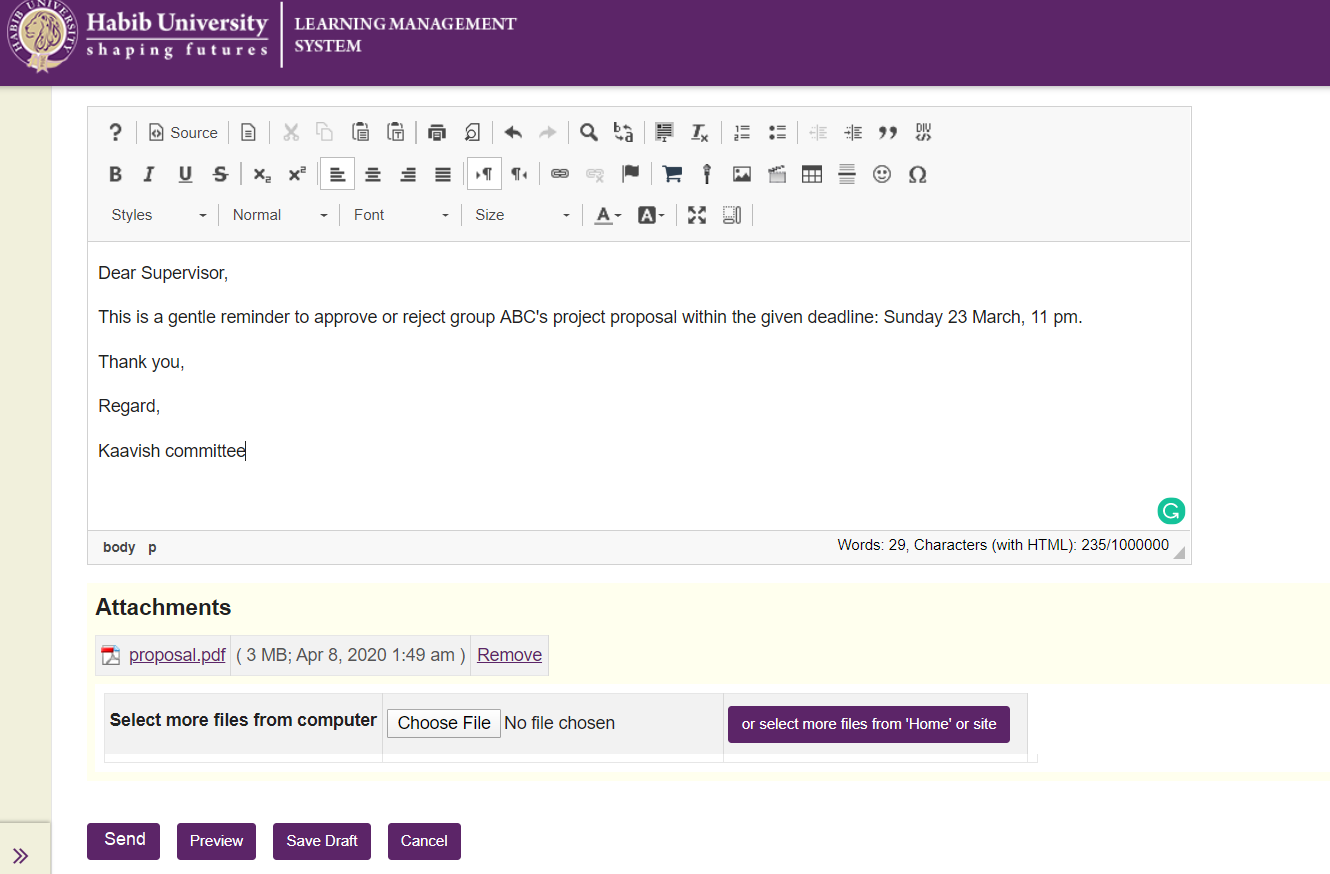


Figure 4. Upon clicking the “Send” button, the KWG can send the project proposal to the supervisor.

### 3.1.2 Hardware Interfaces

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

### 3.1.3 Software Interfaces

The only external software LMS would need to be integrated with is Outlook for sending emails directly from LMS and syncing Outlook calendar with LMS calendar events.

### 3.1.4 Communications Interfaces

There is a good amount of communication which takes place on LMS, but none of it requires a communication service. Everything is handled by the operating system of the hardware. It only requires an internet connection.

## 

## 3.2 Functional Requirements

### 3.2.1 Submitting Project Member Names

3.2.1.1 Introduction

Once a project group is decided, students would submit the group members’ names to the KWG on LMS.

3.2.1.2 Inputs

Task created by the KWG for all students to submit their group member names.

3.2.1.3 Processing

The students fill out and submit the task.

The submission is sent to KWG.

3.2.1.4 Outputs

KWG receives the submission and is notified.

3.2.1.5 Error Handling

Upon submitting more than 4 names, there would be an error message which would warn them to submit maximum 4 names.

### 3.2.2 Making Groups

3.2.2.1 Introduction

KWG should be able to select the members of each project group and put them into their respective groups on LMS, where only them, their supervisors, and the KWG would be able to communicate.

3.2.2.2 Inputs

Clicking the “create new group” button on site info manage groups.

3.2.2.3 Processing

Adding group title and description.

Adding the students to the group.

3.2.2.4 Outputs

The students are now members of the group on LMS.

Note: Once students are made members of a group, whenever the KWG creates a new entry on any site tool e.g. tasks, announcements, calendar events, gradebook, and selects a certain group as the recipients of that site tool entry, the new entry will be available on each of the group members’ LMS.

3.2.2.5 Error Handling

Upon adding more than 4 members into a group, there would be an error notification.

### 3.2.3 Submitting the Proposal

3.2.3.1 Introduction

Group should be able to submit their project proposal on LMS, to be received by the KWG.

3.2.3.2 Inputs

The submission would be similar to how quizzes are submitted on LMS, answering questions about their project idea.

3.2.3.3 Processing

The submission would be sent to the KWG to review.

3.2.3.4 Outputs

The KWG would receive the submission.

3.2.3.5 Error Handling

In case of any required fields being empty, submission would not be possible.

### 3.2.4 Send Proposals for Approval

3.2.4.1 Introduction

The KWG should be able to forward the proposal to at most 4 well-suited faculty members for approval on LMS by check-marking said faculty members.

3.2.4.2 Inputs

Selecting which faculty members to forward the proposal to.

3.2.4.3 Processing

Sending the proposal to the selected faculty.

3.2.4.4 Outputs

The faculty receives the proposal for their approval.

3.2.4.5 Error Handling

Making sure it goes to the right recipients.

### 3.2.5 Assigning Supervisors

3.2.5.1 Introduction

When the proposal is accepted by majority, one of the faculty members is requested to be the supervisor of that group.

3.2.5.2 Inputs

Selecting which faculty member to send the request to, along with an additional message.

3.2.5.3 Processing

Sending the request to the selected faculty.

3.2.5.4 Outputs

The faculty receives the request.

3.2.5.5 Error Handling

Making sure it goes to the right recipient.

### 3.2.6 Accepting Request from KWG

3.2.6.1 Introduction

When a request is received by the faculty member from KWG to supervise - they can choose to accept or reject it.

3.2.6.2 Inputs

Clicking on “Accept” or “Reject” to the request sent by KWG.

3.2.6.3 Processing

Sending the response to the KWG.

3.2.6.4 Outputs

The KWG receives the response.

3.2.6.5 Error Handling

Making sure it goes to the right recipient.

### 3.2.7 Sending Reminders

3.2.7.1 Introduction

KWG should be able to send reminders to the faculty and students to complete given tasks.

3.2.7.2 Inputs

Resending the task by adding a deadline and message after selecting faculty / student groups to send the reminder to.

3.2.7.3 Processing

The task is sent with the reminder notification to the respective faculty/ student group.

3.2.7.4 Outputs

The faculty/ student group receives the task.

3.2.7.5 Error Handling

Deadline to be set appropriately. Making sure it goes to the right recipients.

### 3.2.8 Assigning a Grading Panel

3.2.8.1 Introduction

Each group will need their own grading panel. This panel would be selected by KWG for a group on LMS the same way as supervisors are assigned.

3.2.8.2 Inputs

Choosing which 3-4 faculty members to assign to which project group.

3.2.8.3 Processing

Create a new group on LMS and add grading panel members in it.

3.2.8.4 Outputs

The faculty members are now members of the grading panel group on LMS.

The grading panel member receives a notification of the group chosen for them.

Note: Once faculty members have been made members of the grading panel group on LMS, whenever KWG adds any new announcement, task, calendar event, resource or gradebook entry and selects a certain grading panel group as recipient, the group members will receive a notification and a new entry will be created on the site tools of their LMS.

3.2.8.5 Error Handling

A faculty member must not be a panel member for grading more than 4 projects. The supervisor of a group must not be a panel member for its grading.

### 3.2.9 Sharing Custom Gradebook

3.2.9.1 Introduction

KWG will share a custom gradebook with the Grading Panel and the Supervisor for the first evaluations, where they will be required to enter their grades and detailed feedback.

3.2.9.2 Inputs

Creating a custom gradebook to be shared to the grading panel and supervisors.

3.2.9.3 Processing

Gradebook is sent to the grading panel and supervisor.

Feedback is written and submitted by the grading panel members and supervisor.

3.2.9.4 Outputs

Supervisor and grading panel receive the gradebook. The KWG receives the grades feedback on their LMS.

3.2.9.5 Error Handling

Making sure the gradebook goes to the right recipients.

### 3.2.10 First Evaluation

3.2.10.1 Introduction

Under the **Evaluation 1** category of the FYP group’s gradebook, all the grade and feedback categories will be entered by KWG. After editing the gradebook, KWG will release it to the respective FYP groups.

3.2.10.2 Inputs

KWG editing the gradebook and clicking “Upload” to upload it under **Evaluation 1.**

3.2.10.3 Processing

Gradebook being edited and uploaded on LMS.

3.2.10.4 Outputs

The students will receive a notification about the release of grades and feedback of **Evaluation 1** in their gradebook.

3.2.10.5 Error Handling

Error message will pop up if gradebook being uploaded is of the wrong format.

### 3.2.11 Peer Evaluations

3.2.11.1 Introduction

KWG would be able to create a task for peer evaluation and share it with all students (other than FYP groups).

3.2.11.2 Inputs

Clicking “Post” to launch the task for evaluation.

3.2.11.3 Processing

Peer evaluations task being uploaded and sent to students.

3.2.11.4 Outputs

Students receive notification for the task.

3.2.11.5 Error Handling

Making sure it doesn’t go to FYP group members.

### 3.2.12 Sending Final Grades

3.2.12.1 Introduction

The final grade is divided into different categories depending on the rubric decided by the KWG. The grading panel and supervisor are allotted some percentage of the grade. The KWG should be able to combine the grades from different components and release the final grade for the FYP to the respective FYP group.

3.2.12.2 Inputs

Supervisor’s and grading panels’ final grades.

3.2.12.3 Processing

The grades are combined to form a final grade that is sent to be displayed on the respective FYP group’s gradebook on LMS.

3.2.12.4 Outputs

KWG receives the final grades to add in the gradebook.

3.2.12.5 Error Handling

The grade components should be anonymous to the students but visible to KWG.

### 3.2.13 Compiling Gradebook

3.2.13.1 Introduction

After receiving the grades from the Grading Panel, Supervisor and peer evaluations, KWG will look through and compile such that the categories of the student gradebook are filled with the appropriate grades and remarks. They will then share the final gradebook.

3.2.13.2 Inputs

Adding all grades, peer evaluations and feedback in the appropriate sections of the gradebook, and then finally clicking “Upload” to share with students.

3.2.13.3 Processing

All editing being done in gradebook. Gradebook being uploaded to each groups’ LMS.

3.2.13.4 Outputs

Gradebook will be shared with all and students will be notified of it.

3.2.13.5 Error Handling

Maintaining the privacy of FYP groups by sharing gradebook with concerned groups only.

### 3.2.14 Creating Calendar Event

3.2.14.1 Introduction

The KWG will create a calendar event for the Student Group’s evaluations, which will happen twice a year for every Student Group. The time, location and description of the event will be added.

3.2.14.2 Inputs

Marking time and date for the event and sending invites to selected groups.

3.2.14.3 Processing

Invites being sent to recipients.

3.2.14.4 Outputs

Invited groups receiving the invite on Outlook email and LMS and the event is added to their calendars.

3.2.14.5 Error Handling

Making sure it goes to the right recipients.

### 3.2.15 Add, Edit, and Delete Group Members

3.2.15.1 Introduction

The KWG will be able to add new members, edit the existing members or delete members from the group as needed. They would go to “Manage Groups” on Site Info, choose the group and update group participants as needed.

3.2.15.2 Inputs

Selecting members and clicking the update button.

3.2.15.3 Processing

Member being removed/added.

3.2.15.4 Outputs

KWG or supervisor successfully added/removed a member.

3.2.15.5 Error Handling

Warning at adding more than 4 students in a group.

### 3.2.16 Dashboard Entries and Notes

3.2.16.1 Introduction

Students would be able to add entries in their dashboard. The entries represent the overall workflow while the notes in each entry will contain all information regarding a certain task. A group member can choose an option to **Create New Entry** which generates a new entry. in which they can **Add Note,** which generates a new note in the existing entry or **Edit Note** which allows one to edit any existing note of an entry. They can click on “Save” to save their work.

3.2.16.2 Inputs

Clicking any of the options mentioned above, writing entries, and then clicking “Save”.

3.2.16.3 Processing

Entries and notes being created, edited, added etc., and then being saved.

3.2.16.4 Outputs

Once an entry is created in the dashboard, the activity feed will be updated and a notification will be sent to KWG and the supervisor.

3.2.16.5 Error Handling

Error message at saving empty notes/entries.

### 3.2.17 Filter Search

3.2.17.1 Introduction

Students can search their notes by applying a filter. When Filter Search is performed, the search result will be returned which will be a list of filtered notes.

3.2.17.2 Inputs

Text entered in search box and/or other filters selected to search by (e.g. deadline, labels, assigned-to), then clicking “go” or pressing enter key.

3.2.17.3 Processing

Searching for notes that match the criteria selected.

3.2.17.4 Outputs

A list of filtered note(s) is displayed.

3.2.17.5 Error Handling

Filter with no matches will return nothing.

### 3.2.18 Notes Features

3.2.18.1 Introduction

A note can be edited, deleted, its title can be changed. Description, checklist, labels, assigned-to, and deadline can be added to it. Moreover, attachments can be added or removed and a new comment can be added. Assigned-to can be changed to assign the task to other members.

3.2.18.2 Inputs

Choosing to Edit, Delete. Adding description, checklist, labels, assigned-to, and deadline.

3.2.18.3 Processing

Note being edited, deleted, description, checklist, labels etc. being added.

3.2.18.4 Outputs

Note is updated according to inputs and is visible to all group members and KWG.

Group members i.e. supervisor and students and KWG receive notification of activity and Activity feed is updated.

3.2.18.5 Error Handling

Empty note cannot be saved.

### 3.2.19 Adding New Site Tool

3.2.19.1 Introduction

KWG would be able to create a new Site for FYP and add CS faculty, Kaavish enrolled students, and KWG members. The creator can add site tools e.g. Overview, Announcements, Calendar, Tasks, Gradebook, Resources, Dashboard, and Roster.

3.2.19.2 Inputs

Clicking Create New Site, selecting the desired tools and then adding site participants and their roles as student, faculty member or admin/ KWG member.

3.2.19.3 Processing

Site being published, tools and participants being updated.

3.2.19.4 Outputs

FYP Site created. Added participants are notified.

3.2.19.5 Error Handling

If a wrong participant is added to the site, they shall be removed by KWG.

## 

## 3.3 Use-cases

### 3.3.1 Use-case #1

#### 3.3.1.1 Brief Description

KWG member creates a site for Kaavish on LMS.

#### 3.3.1.2 Actors

**3.3.1.2.1 KWG**

**3.3.1.2.2 CS Students**

**3.3.1.2.3 Faculty**

**3.3.1.2.4 LMS Server**

#### 3.3.1.3 Preconditions

Kaavish site creator is logged into LMS.

Students are enrolled in Kaavish.

#### 3.3.1.4 Basic Flow of Events

1. The use-case begins when the KWG member clicks on Create New Site on LMS.
2. KWG member adds the academic term, site name and an optional description.
3. Upon clicking Continue, the KWG member will be allowed to select site tools.
4. KWG member will select the following tools: Overview, Announcements, Calendar, Tasks, Gradebook, Resources, Dashboard, Roster and Site Info.
5. KWG member will then add participants to the site i.e. senior CS students enrolled in Kaavish, KWG members and CS faculty.
6. KWG will click on Publish Site.
7. The added participants will receive a notification that they have been added to the site. It will appear as a tab on their LMS.
8. The use-case ends successfully.

#### 3.3.1.5 Alternative Flows

##### 3.3.1.5.1 Wrong participant added to Kaavish site.

In step 5, if a participant is incorrectly added to the Kaavish site,

1. KWG will select the participant and remove them from the Kaavish site.
2. Upon pressing Update Participants, the said participant will be removed.
3. The Kaavish site will no longer appear in the removed participant’s LMS.
4. The use-case ends.

#### 3.3.1.6 Key Scenarios

**3.3.1.6.1 Creating the site.**

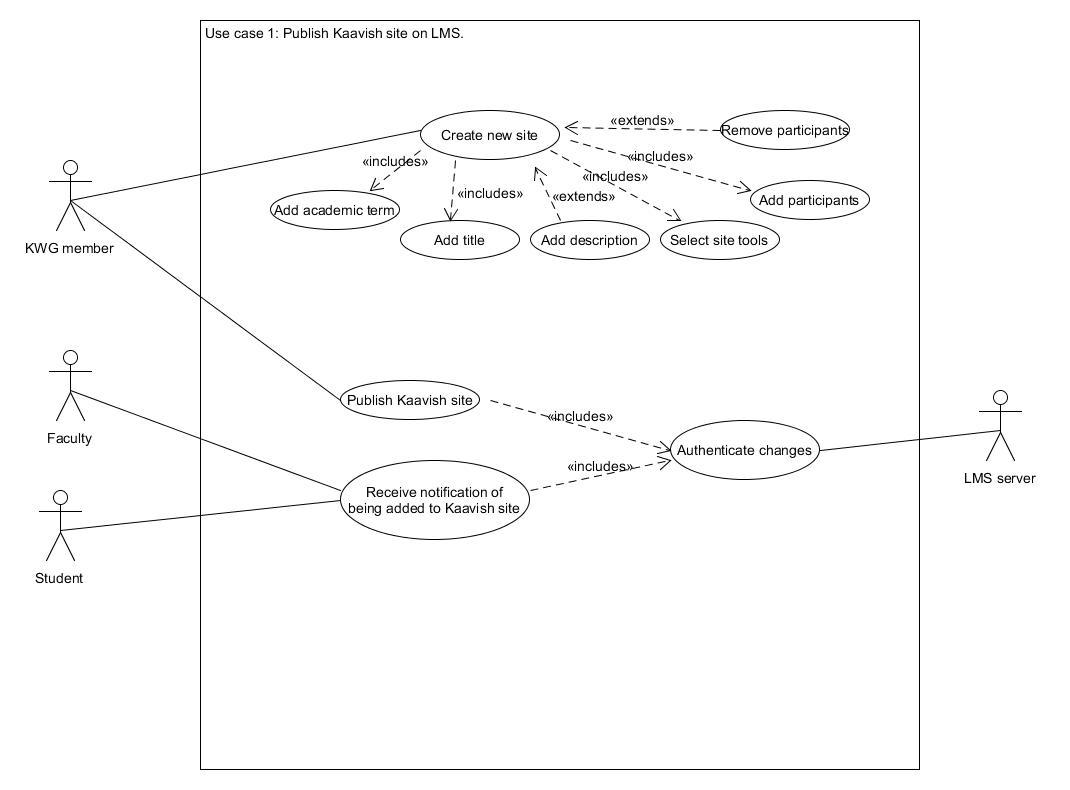
#### 3.3.1.7 Post-conditions

##### 3.3.1.7.1 Successful Completion

The project groups are finalized and all groups’ details are recorded in a file. KWG can successfully create groups.

##### 3.3.1.7.2 Failure Condition

KWG fails to make a site and has to start over.



### 3.3.2 Use-case #2

#### 3.3.2.1 Brief Description

Students submitting FYP group members’ names.

#### 3.3.2.2 Actors

**3.3.2.2.1 CS Student**

**3.3.2.2.2 KWG**

**3.3.2.2.3 LMS Server**

#### 3.3.2.3 Preconditions

Users have an active LMS account and are logged in.

Students are enrolled in Kaavish and are added to the FYP site on LMS.

Students have decided their group members.

#### 3.3.2.4 Basic Flow of Events

1. The use-case begins when the KWG creates a new task in the **Tasks** tab on LMS. The task asks for the team name, (three required and one optional) group members’ names and their IDs. The task has a submission deadline.
2. The task is saved and all students are notified.
3. One member from each group fills and submits the task with the name and ID of each group member.
4. KWG receives a notification for each submission.
5. KWG member verifies whether all enrolled students are part of some group.
6. The use-case ends successfully.

#### 3.3.2.5 Alternative Flows

##### 3.3.2.5.1 After the deadline has passed, a student enrolled in Kaavish is not part of any submitted group.

In step 5, if a student in class roster is not found in any of the submissions.

1. KWG sends an email to all such students.
2. Student group emails KWG and requests for an extension to submit the task, or withdraw from the course.
3. If the student informs about withdrawal, they are removed from the course site. If the student requests for extension, then KWG chooses to accept or reject.
4. If KWG extends the submission deadline, students can go back to Step 2 of basic flow. If rejected, the use-case ends with a failure.

##### 3.3.2.5.2 One student is part of two or more groups.

In step 5, if a student’s name and ID is found to be submitted by more than one group,

1. KWG sends an email to the concerned student to confirm which group they are actually part of.
2. Student chooses one group and emails KWG.
3. KWG emails members of other group(s) to resubmit the task by a new deadline.
4. KWG cancels the previous submission of other group(s).

#### 3.3.2.6 Key Scenarios

**3.3.2.6.1 Students completing the task within the deadline.**

#### 3.3.2.7 Post-conditions

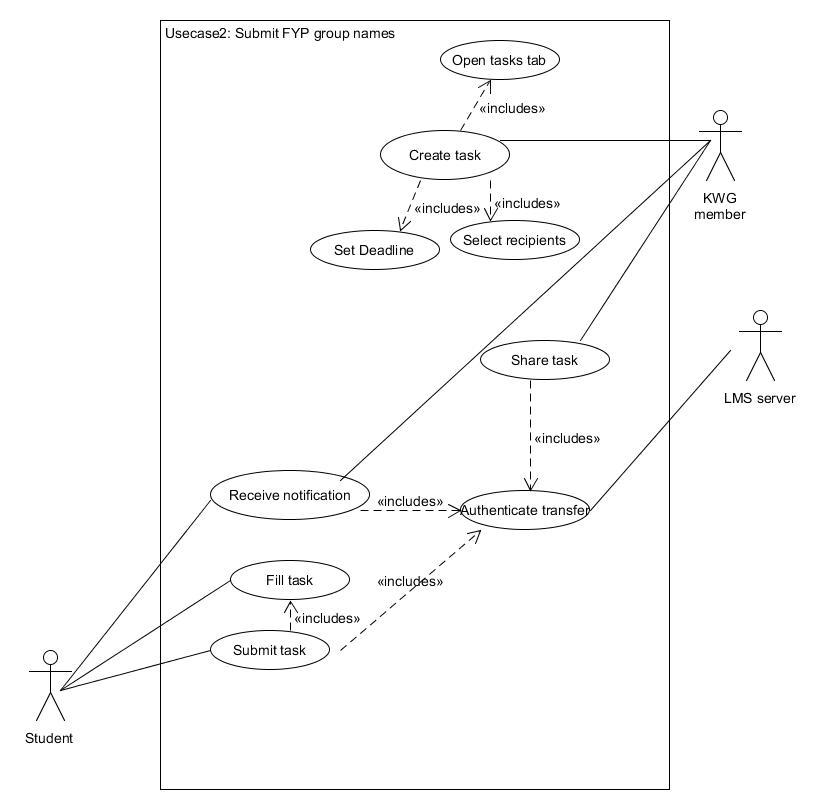
##### 3.3.2.7.1 Successful Completion

Kaavish site is successfully created and appears on added participants’ LMS.

##### 3.3.2.7.2 Failure Condition

Kaavish site is not updated and the KWG member/ site creator is notified.

### 



### 3.3.3 Use-case #3

#### 3.3.3.1 Brief Description

This use-case describes how KWG creates groups on LMS.

#### 3.3.3.2 Actors

**3.3.3.2.1 KWG**

**3.3.3.2.2 Student Group**

**3.3.3.2.3 LMS Server**

#### 3.3.3.3 Preconditions

Users have an active LMS account and are logged in.

Students are enrolled in Kaavish, all groups have submitted group details and KWG has verified these details.

#### 3.3.3.4 Basic Flow of Events

1. The use-case begins after KWG verifies submission of all student groups and information is recorded in file.
2. KWG creates groups on LMS by clicking “site info”, “manage groups” and “create new groups”. (Supported by existing LMS 12.5)
3. The description of the group is added by importing group information from the file.
4. The student group members are added to their respective groups.
5. Students are notified about group creation.
6. Use-case ends successfully.
7. Now, whenever the KWG creates a new entry on any site tool e.g. tasks, announcements, calendar events, gradebook, and selects a certain group as the recipients of that site tool entry, the new entry will be available on each of the group members’ LMS.

#### 3.3.3.5 Alternative Flows

##### 3.3.3.5.1 Wrong participant added to Kaavish site.

In step 4, if a student is incorrectly added to the FYP group,

1. KWG will select the student and remove them from the group.
2. Upon pressing Update Participants, the said participant will be removed.
3. The FYP group will no longer appear in the removed participant’s LMS.
4. The use-case ends.

#### 3.3.3.6 Key Scenarios

**3.3.3.6.1 Creating the groups.**

#### 3.3.3.7 Post-conditions

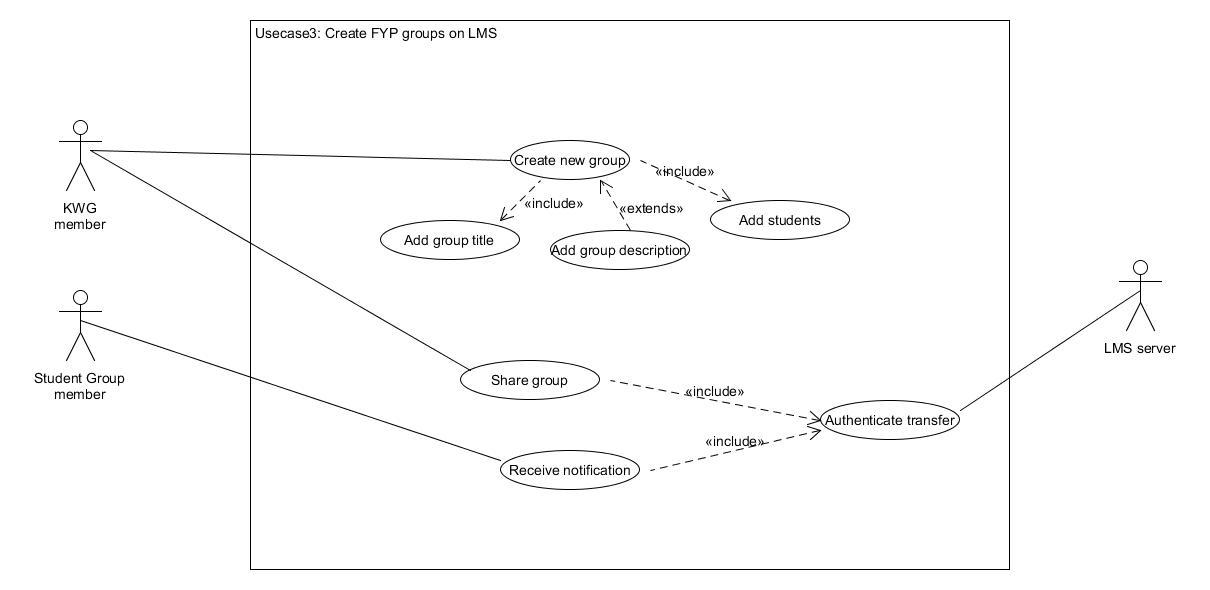
##### 3.3.3.7.1 Successful Completion

A students group is created on LMS.

##### 3.3.3.7.2 Failure Condition

KWG is notified of a system failure.

The logs have been updated accordingly.



### 

### 3.3.4 Use-case #4

#### 3.3.4.1 Brief Description

This use-case describes how the student group submits their project proposal.

#### 3.3.4.2 Actors

**3.3.4.2.1 KWG**

**3.3.4.2.2 Student Group**

**3.3.4.2.3 LMS Server**

#### 3.3.4.3 Preconditions

Users have an active LMS account and are logged in.

FYP student groups have been created on LMS.

#### 3.3.4.4 Basic Flow of Events

1. The use-case begins after the FYP groups are formed on LMS.
2. KWG creates a formatted task for proposal submission in **Tasks** tab. KWG enables group submission for the task and selects FYP groups as recipient of the Task.
3. Once the task is posted, all members of the FYP group will receive a notification and an entry of the task will be created on their LMS under Tasks tab.
4. Since group submissions are allowed on the task, any one of the group members can submit a fully filled proposal in the task entry.
5. They submit the proposal within the deadline by clicking on the “Submit” button.
6. All group members will receive a notification for task submission on their LMS.
7. The KWG will be notified that the group has submitted their proposal.
8. The use-case ends successfully.

#### 3.3.4.5 Alternative Flows

##### 3.3.4.5.1 Deadline passed

If the student group fails to submit their proposal within the deadline as in step 3, they won’t be allowed to submit until the deadline is extended for them by KWG.

1. The student group emails KWG for extension.
2. KWG rejects or accepts their request by email.
3. If accepted, the use-case resumes from step 2. If not, the use-case ends.

#### 3.3.4.6 Key Scenarios

**3.3.4.6.1 Students submitting the task.**

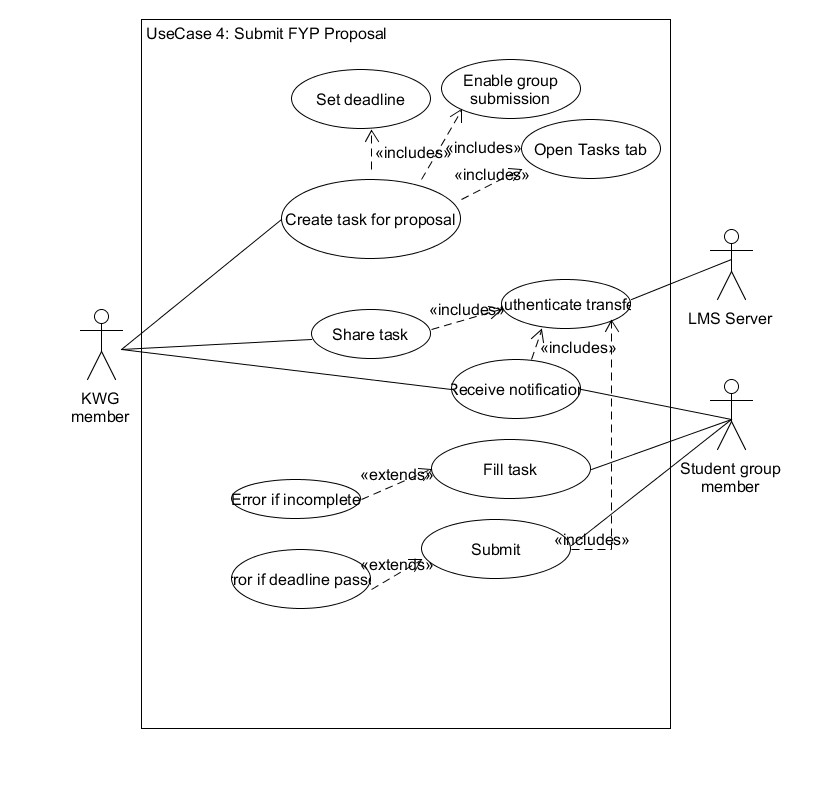
#### 3.3.4.7 Post-conditions

##### 3.3.4.7.1 Successful Completion

The group has successfully submitted their proposal, and KWG receives it.

##### 3.3.4.7.2 Failure Condition

The group fails to submit the proposal and has to submit again if granted a resubmission by KWG.



### 3.3.5 Use-case #5

#### 3.3.5.1 Brief Description

This use-case describes how the KWG accepts or rejects a project proposal based on feedback from faculty.

#### 3.3.5.2 Actors

**3.3.5.2.1 KWG**

**3.3.5.2.2 Faculty member**

**3.3.5.2.3 Student Group**

**3.3.5.2.4 LMS Server**

#### 3.3.5.3 Preconditions

Users have an active LMS account and are logged in.

Use-case 3 has already occurred i.e. proposal has been submitted by the student group to the KWG.

The KWG member has gone through the proposal and decided on 3-4 faculty members to send the proposal to, based on their expertise.

#### 3.3.5.4 Basic Flow of Events

1. The use-case begins after the KWG member decides on the faculty who will be best suited to assess whether a group’s proposal should be accepted or not.
2. KWG member clicks on the **Tasks tab** and opens the student group’s submitted project proposal task.
3. They press the **Forward** button and select from an available pool of faculty members to send the proposal to. Upon clicking **Next**, an <automated> message will appear on the textbox, requesting the supervisor to respond within a specified deadline.
4. Upon pressing **send**, a task is created under the Tasks tab of the said faculty member’s LMS and a deadline is set to complete the task within the due date.
5. Faculty member will open the task and fill in their feedback. They will either accept or reject the proposal and click **send**.
6. KWG will be notified of the submitted task.
7. Once all of the responses have been received by KWG, the KWG member will compile the feedback and post an announcement for the student group informing them if their proposal has been accepted or not.
8. The students are notified (via Announcement).
9. The use-case ends successfully.

#### 3.3.5.5 Alternative Flows

##### 3.3.5.5.1 No response from faculty member (deadline has passed)

If in step 5 of the basic flow, not all faculty members have completed the task for acceptance of the proposal,

1. The KWG member has to re-send the proposal to the said faculty.
2. The use-case resumes at step 2.

##### 3.3.5.5.2 Faculty member not available.

If in step 5 of the basic flow, the faculty member who needs to send their feedback for acceptance of the proposal to KWG, is not available due to some circumstance.

1. The KWG member re-evaluates the proposal and thinks of another faculty available in the pool that would suit the proposal.
2. The use-case resumes at step 2.

#### 3.3.5.6 Key Scenarios

**3.3.5.6.1 Creating task for Faculty to complete**

#### 3.3.5.7 Post-conditions

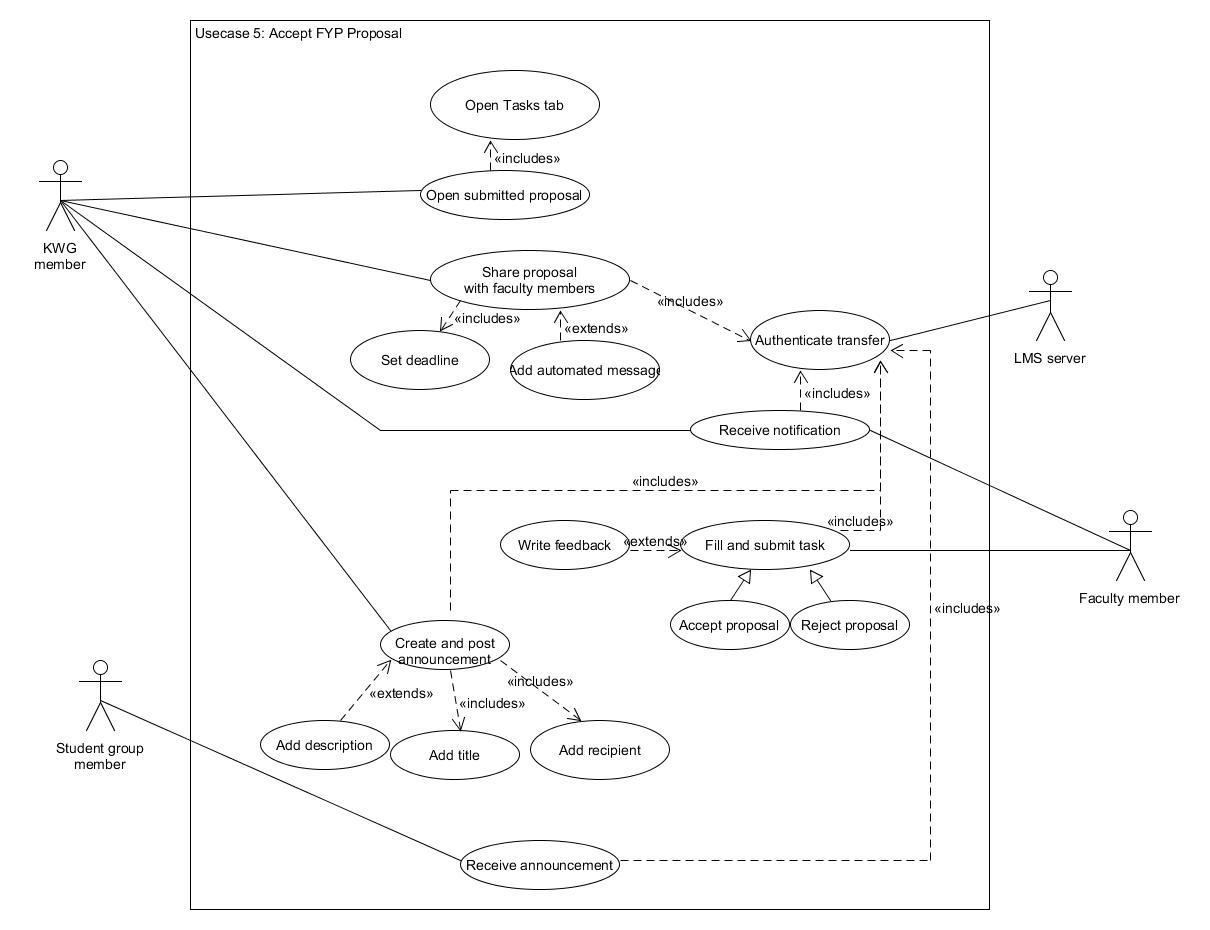
##### 3.3.5.7.1 Successful Completion

The project group has been notified (via announcements) if their project proposal has been accepted and the status of the group is updated on the system.

##### 3.3.5.7.2 Failure Condition

KWG is notified of a system failure.

The logs have been updated accordingly.



### 3.3.6 Use-case #6

#### 3.3.6.1 Brief Description

This use-case describes how the KWG assigns supervisors to each FYP group.

#### 3.3.6.2 Actors

**3.3.6.2.1 KWG**

**3.3.6.2.2 Faculty member**

**3.3.6.2.3 LMS Server**

#### 3.3.6.3 Preconditions

Users have an active LMS account and are logged in.

Use-case 3 and 4 have already occurred.

The KWG member has gone through the proposal and decided on 3-4 supervisors.

#### 3.3.6.4 Basic Flow of Events

* 1. The use-case begins after the KWG member decides on the faculty who will be best suited to supervise a group based on the group’s proposal, based on students’ preference for the supervisor mentioned in the proposal.
  2. KWG member clicks on the Tasks tab, opens student group’s proposal and presses the **Forward** button. KWG member selects the faculty member to supervise the group from an available pool and presses **Next**.
  3. KWG member sends an <automated> message with the attached proposal to the supervisor, requesting them to respond within a deadline.
  4. Upon pressing **send**, a task is created under the Tasks tab of the said faculty member’s LMS and a deadline is set to complete the task within the due date.
  5. Faculty member will open the task and either accept or decline the option to supervise and click **send**.
  6. KWG will be notified of the submitted task.
  7. If the faculty member accepts the supervision, then KWG member will add them to the FYP group on LMS with the role as their Supervisor.
  8. Said faculty member is automatically removed from the pool if they have been assigned as supervisor to more than 4 groups.
  9. The students are notified.
  10. The use-case ends successfully.

#### 3.3.6.5 Alternative Flows

##### 3.3.6.5.1 No one appropriate available in pool

If in step 2 of the basic flow KWG is unable to find a faculty member that would suit the proposal,

1. the KWG member re-evaluates the proposal and finds another faculty available in the pool that would suit the proposal.
2. the use-case resumes at step 2.

##### 3.3.6.5.2 Rejected request

If in step 4 of the basic flow, a chosen supervisor rejects the given proposal and doesn’t agree to be the supervisor for that FYP group for whatever reason, then

1. The KWG is notified of rejection.
2. KWG decides on a different faculty member available in the pool.
3. The use-case resumes at step 2.

#### 3.3.6.6 Key Scenarios

**3.3.6.6.1 Assigning supervisors according to student groups’ preference.**

#### 3.3.6.7 Post-conditions

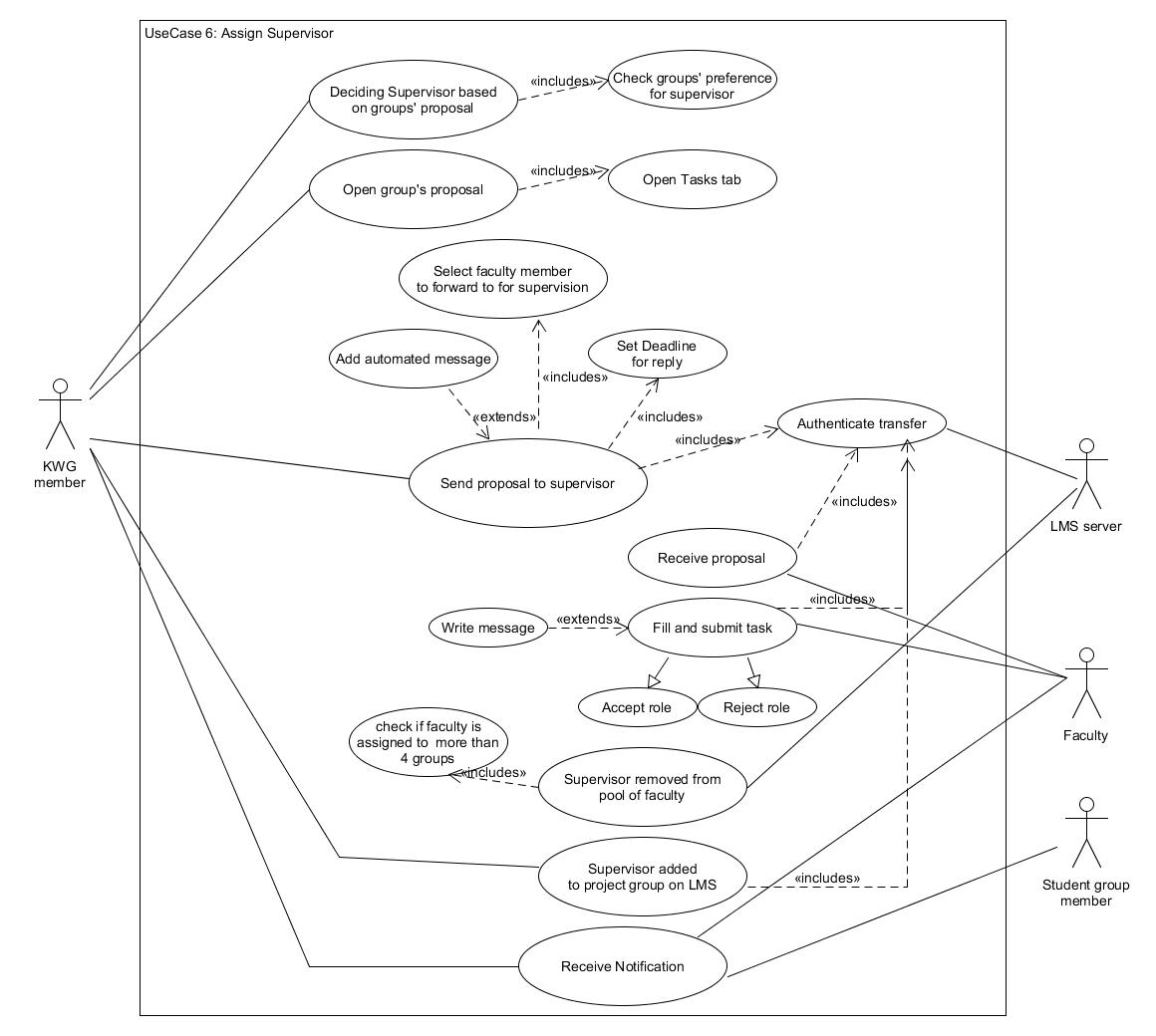
##### 3.3.6.7.1 Successful Completion

The project group has been assigned a supervisor successfully and the data has been updated in the system.

##### 3.3.6.7.2 Failure Condition

KWG is notified of a system failure.

The logs have been updated accordingly.



### 3.3.7 Use-case #7

#### 3.3.7.1 Brief Description

This use-case describes how the FYP group keeps track of their progress using the Dashboard feature.

#### 3.3.7.2 Actors

**3.3.7.2.1. Supervisor**

**3.3.7.2.2. Student Group**

**3.3.7.2.3. KWG**

**3.3.7.2.4. LMS Server**

#### 3.3.7.3 Preconditions

Users have an active LMS account and are logged in.

All students have been added to their respective groups i.e. all of the previous use-cases have occurred.

All FYP group members and supervisors have access to the Dashboard tab.

#### 3.3.7.4 Basic Flow of Events

1. The KWG member enables group options for the dashboard. This enables all FYP group members to access the same dashboard.
2. Dashboard will consist of three windows; the Activity feed, the workspace and list of members.
3. In the workspace, the entries will represent the overall workflow while the notes in each entry will contain all information regarding a certain task. Activity feed will show every update that happens on the Dashboard.
4. A member of the Dashboard chooses an option to **Create New Entry** which generates a new entry or **Add Note** which generates a new note in the existing entry or **Edit Note** which allows one to edit an existing note of an entry.
5. A member can **Search by Filter,** where a filter is selected and/or search by text is performed. The result of the search will be returned which will be a list of filtered notes.
6. The user can select and edit any note from the Search result.
7. When an entry is created, the creator will be asked to give it a title (e.g. To-Do or In-Progress etc.) and optional description. It will pop up on the main workspace once created.
8. All members will receive a notification when an entry is created. The modification to the Dashboard is shown as a new message in the Activity feed.
9. When a new note is created in an entry, it will have multiple attributes such as title, description, assigned-to, deadline, checklist(sub-tasks), attachments (docs) and label. This new note will be displayed in its respective entry..
10. When a member edits a note, they will be allowed to delete the note or change title, description, checklist, labels, assigned-to, deadline. Moreover, attachments can be added or removed and a new comment can be added. Assigned-to can be changed to assign the task to other members.
11. All members will receive a notification when a new note is added or edited. The modification to the Dashboard is shown as a new message in the Activity feed.
12. The use-case ends successfully.

#### 3.3.7.5 Alternative Flows

##### 3.3.7.5.1 Attachment not supported

If in step 2 of the basic flow, system rejects the file due to format or size,

1. Members are unable to share their progress.
2. The use-case ends with a failure condition.

#### 3.3.7.6 Key Scenarios

**3.3.7.6.1 Creating a timeline to represent student’s progress.**

**3.3.7.6.2 Providing feedback and comments on the progress.**

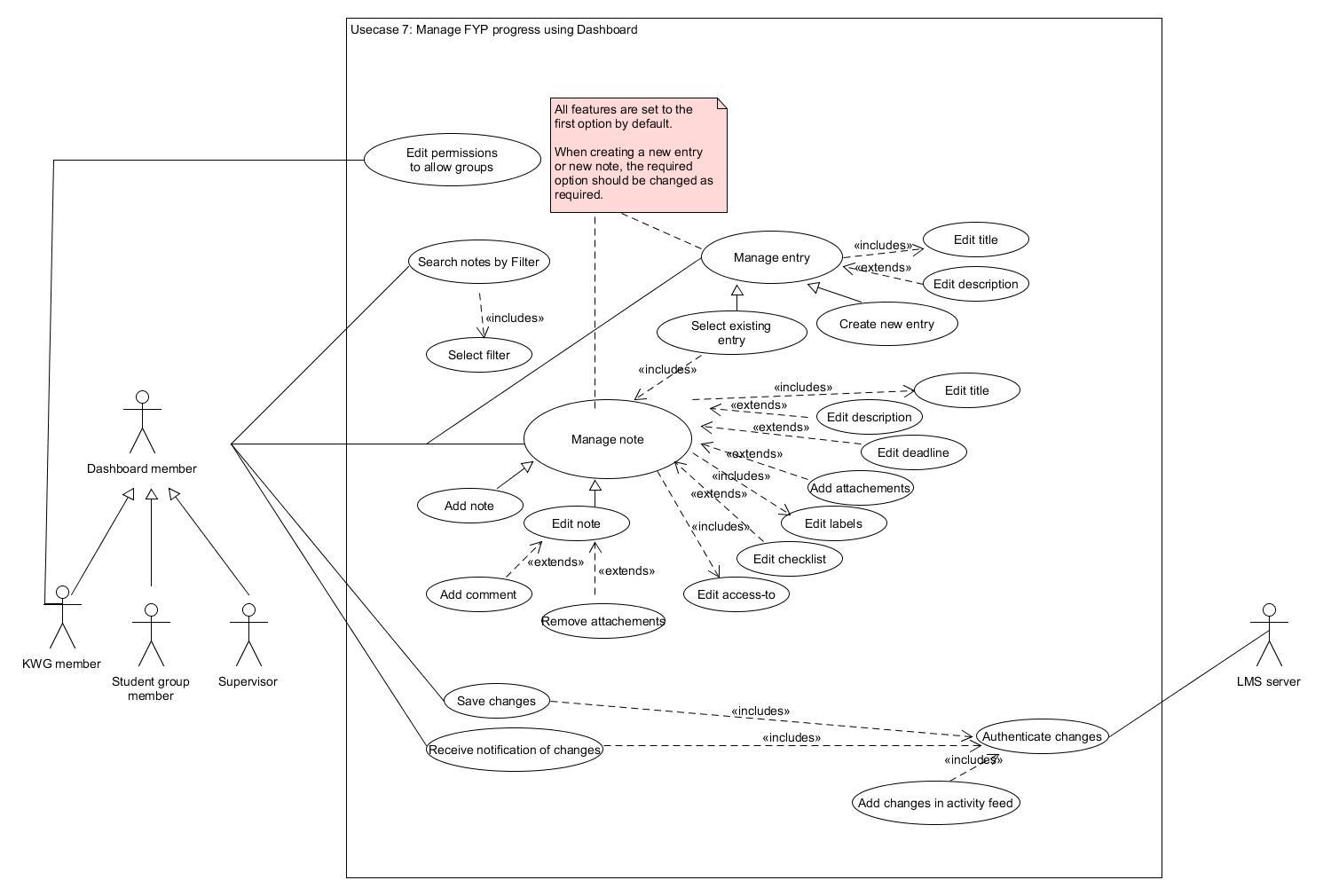
#### 3.3.7.7 Post-conditions

##### 3.3.7.7.1 Successful Completion

The project group and supervisor receives a notification of the modification in the Activity feed.

##### 3.3.7.7.2 Failure Condition

The modification is not recorded and has to be re-done.



### 3.3.8 Use-case #8

#### 3.3.8.1 Brief Description

The use-case describes how KWG creates the Grading Panel for each Student Group.

#### 3.3.8.2 Actors

**3.3.8.2.1 KWG**

**3.3.8.2.2 Student group**

**3.3.8.2.3 Grading panel members**

**3.3.8.2.4 FYP group supervisor**

**3.3.8.2.5 LMS server**

#### 3.3.8.3 Preconditions

Users have an active LMS account and are logged in.

All students have been added to their respective FYP groups.

All the FYP groups have a supervisor assigned and project timelines in the dashboard are being maintained i.e. all of the previous use-cases have worked successfully.

The Kaavish Committee has decided which faculty members will be part of the Student Group’s Grading Panel.

#### 3.3.8.4 Basic Flow of Events

1. The use-case begins when the KWG decides Grading Panel members for all Student Groups.
2. The KWG member will create a group for the FYP project’s Grading Panel on LMS, add the appropriate title and description. (see use-case 2 for more details)
3. Upon adding the Grading Panel members, an automatic notification will be sent to the added members.
4. KWG will also add the rubrics of assessment and the FYP group’s proposal in the Resources tab of the group.
5. The KWG member will click on the Calendar tab and create calendar events for the Student Group’s evaluations, which will happen twice a year for every Student Group. The time, location and description of the event will be added.
6. After being shared, the event will be added to the calendars of the FYP’s Grading Panel group members and the FYP Group members’ LMS.
7. Group members will receive a notification of the calendar events.
8. The use-case ends successfully.

#### 3.3.8.5 Alternative Flows

##### 3.3.8.5.1 Grading panel member is not available

Due to some circumstances, a faculty member is not able to continue being a part of the grading panel.

1. The KWG member re-evaluates the proposal and thinks of another faculty available in the pool to be added into the grading panel.
2. The KWG will remove the unavailable faculty member from the grading panel group.
3. Another faculty member will be added into the grading panel group. The said faculty member will be notified that they have been added into the group.
4. Use-case resumes at step 4.

#### 3.3.8.6 Key Scenarios

**3.3.8.6.1 Creating the Grading Panel.**

**3.3.8.6.2 Using the calendar tab to schedule events for the Grading panel and FYP group.**

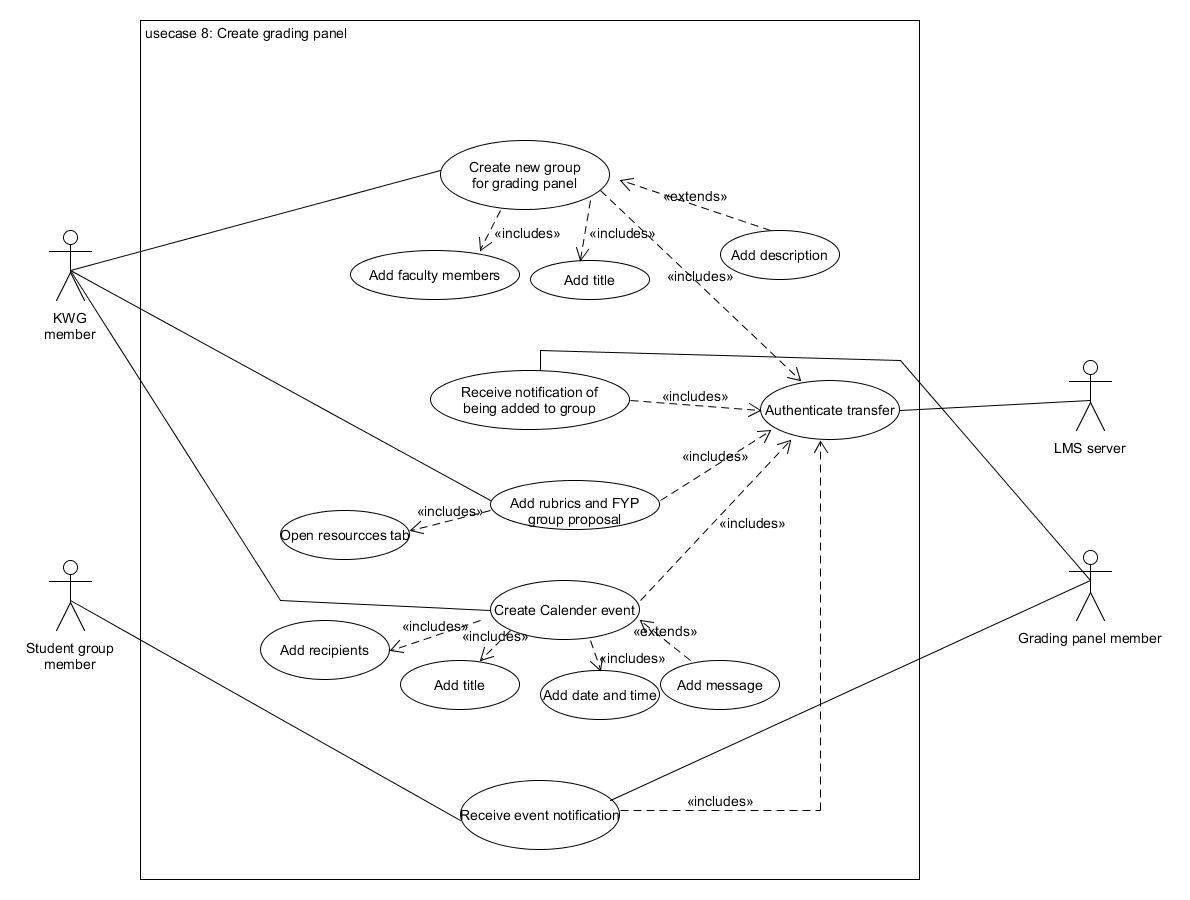
#### 3.3.8.7 Post-conditions

##### 3.3.8.7.1 Successful Completion

The Grading Panel for every Student Group is created. The schedule for evaluations, final submission and presentation is released.

##### 3.3.8.7.2. Failure Condition

KWG will need to re-create the Grading Panel and update the schedule.



### 3.3.9 Use-case #9

#### 3.3.9.1 Brief Description

The use-case describes how KWG creates the custom grade-book, sends it to the Grading Panel members and/or Supervisor and receives the marked grade-books when the said faculty sends it back.

#### 3.3.9.2 Actors

**3.3.9.2.1 KWG**

**3.3.9.2.2 Grading Panel**

**3.3.9.2.3 Supervisor**

**3.3.9.2.4 LMS server**

#### 3.3.9.3 Preconditions

Users have an active LMS account and are logged in.

KWG has decided the categories and weightages of those categories, according to the rubric for both the evaluations.

KWG has decided the supervisor and the members of the Grading Panel for each FYP Group.

#### 3.3.9.4 Basic Flow of Events

1. The use-case begins when KWG chooses the option of creating a new custom grade-book for a particular evaluation. Its title is the evaluation for which it is created.
2. KWG selects a particular FYP Group, and the grade-book is assigned to it. Now, the member of that group’s Grading Panel group and their Supervisor will be the only options for ‘recipients’ for that grade-book.
3. KWG then adds different grade categories that are part of the rubric. There is also a section to comment with each category for feedback.
4. KWG then assigns weightage to all the categories.
5. KWG then selects the option “send to:” and proceeds to the next page to select ‘recipients’, KWG will select the Grading Panel member and/or Supervisor accordingly. KWG also selects a deadline for this task before sending it.
6. KWG sends the customized grade-book to the respective recipients (Grading Panel members and/or Supervisor) who, on receiving it in their ‘Tasks’ tab on LMS with a deadline, will receive a notification.
7. The recipients open the grade-book shown in their ‘Tasks’. They fill-in the categories with their grades and click ‘submit’ to send the marked grade-books back to KWG.
8. KWG receives a notification on receiving the filled grade-books back.
9. As each grade-book is assigned to a particular FYP Group, all are shown as groups in a list. Each group consists of grade-books sent back by a particular FYP Group’s Grading Panel members and/or Supervisor.
10. The use-case ends successfully.

#### 3.3.9.5 Alternative Flows

##### 3.3.9.5.1 Grading panel members are not available.

Due to some circumstances, a faculty member is not able to continue being a part of the grading panel,

1. The KWG member re-evaluates the proposal and thinks of another faculty available in the pool to be added into the grading panel.
2. The KWG will remove the unavailable faculty member from the grading panel group.
3. Another faculty member will be added into the grading panel group. The said faculty member will be notified that they have been added into the group.
4. The KWG will forward the submitted files of the student group and the custom-gradebook to the new grading panel member.

##### 3.3.9.5.2 The recipients miss the deadline.

Due to some circumstances, a faculty member is not able to fill the custom grade-book on time and misses the deadline,

1. KWG will extend the deadline and re-open submission.
2. The recipient of the custom grade-book will have sufficient time to fill in the grade-book and submit with the new deadline.

#### 3.3.9.6 Key Scenarios

**3.3.9.6.1 Sharing a custom grade-book of each FYP group with their respective Grading Panel and Supervisor for grading and receiving it back.**

#### 3.3.9.7 Post-conditions

##### 3.3.9.7.1 Successful Completion

KWG successfully sends the customized grade-books to faculty and receives the filled grade-book from faculty.

##### 3.3.9.7.2 Failure Condition

KWG re-starts the process and creates a new grade-book of the FYP group.

### 

### 3.3.10 Use-case #10

#### 3.3.10.1 Brief Description

The use-case describes how KWG manages the grading process.

#### 3.3.10.2 Actors

**3.3.10.2.1 Supervisor**

**3.3.10.2.2 Student Group**

**3.3.10.2.3 Grading Panel**

**3.3.10.2.4 KWG**

**3.3.10.2.5 LMS Server**

#### 3.3.10.3 Preconditions

Users have an active LMS account and are logged in.

The Grading Panel of each FYP group is created and the grading rubric and schedule for evaluation have been shared with them i.e. all previous use-cases have completed successfully.

#### 3.3.10.4 Basic Flow of Events

1. The use-case begins when the first evaluation takes place according to the schedule shared with the Grading Panel and the Student Groups.
2. KWG will share a custom gradebook with the Grading Panel and the Supervisor for the first evaluations, who will receive it in their ‘Tasks’ tab.
3. The members of the Grading Panel the Supervisor will submit their marked gradebooks within the due-date and KWG will receive a notification about the submission.
4. KWG will be able to read the grades and feedback of the Grading Panel members and Supervisor for the first evaluation, and then will proceed to edit the gradebook of the FYP groups by compiling the grades and feedback from all custom gradebooks.
5. Under the *evaluation#1* category of the FYP group’s gradebook, all the grades and feedback will be entered. After editing the gradebook, KWG will release it to the respective FYP groups.
6. The students will receive a notification about the release of grades and feedback of *evaluation#1* in their gradebook.
7. KWG will create a final submission in ‘Tasks’ for all FYP Groups where they will be required to submit their final working project file’s github invite.
8. The FYP groups will share their github invite link. KWG will receive a notification on submission.
9. KWG is responsible for creating a custom gradebook for Grading Panel members and Supervisors for the final evaluations. KWG sends it to them with a deadline.
10. KWG creates a ‘Task’ for peer evaluation, with a deadline, and shares it with all students.
11. The Grading Panels and the Supervisor fills the gradebooks and submit it to the KWG, who receives a notification for it.
12. The students fill their peer evaluations for all other members of their group and submit, KWG will receive a notification when they receive their submissions.
13. After receiving the grades from the Grading Panel, Supervisor and peer evaluations, KWG will compile all grades, such that the categories of *evaluation#2* of the student gradebook are filled with the appropriate grades and remarks.
14. KWG will share the edited final FYP group’s gradebook.
15. The student group will be notified and will be able to view their grades.
16. The use-case ends successfully.

#### 3.3.10.5 Alternative Flows

##### 3.3.10.5.1 Gradebooks not submitted within the due-date.

If the Grading Panel member or the Supervisor forget to submit the filled gradebook on time, they will be reminded by the KWG and will be given a new extended deadline. This will not have any impact on the rest of the schedule.

1. KWG sends a reminder; the same way it was sent in use-case 2.
2. The Grading Panel members submit the grades within the new due-date.
3. The use-case continues successfully.

##### 3.3.10.5.2 Deadline passed for peer evaluations.

If a student fails to submit peer evaluations within the deadline as in step 12, they won’t be allowed to submit until the deadline is extended for them by KWG.

1. The student emails/messages KWG for extension.
2. KWG rejects or accepts their request by email/message.
3. If accepted, the use-case resumes from step 10. If not, the use-case ends.

##### 3.3.10.5.3 Deadline passed for project file submission.

If the student group fails to submit their project files within the deadline as in step 8, they won’t be allowed to submit until the deadline is extended for them by KWG.

1. The student group emails/messages KWG for extension.
2. KWG rejects or accepts their request by email/message.
3. If accepted, the use-case resumes from step 7. If not, the use-case ends.

#### 3.3.10.6 Key Scenarios

**3.3.10.6.1 Custom gradebooks shared with the Grading Panel and the Supervisor.**

**3.3.10.6.2 The students submit their final project files.**

**3.3.10.6.3 The students receive their final FYP grades.**

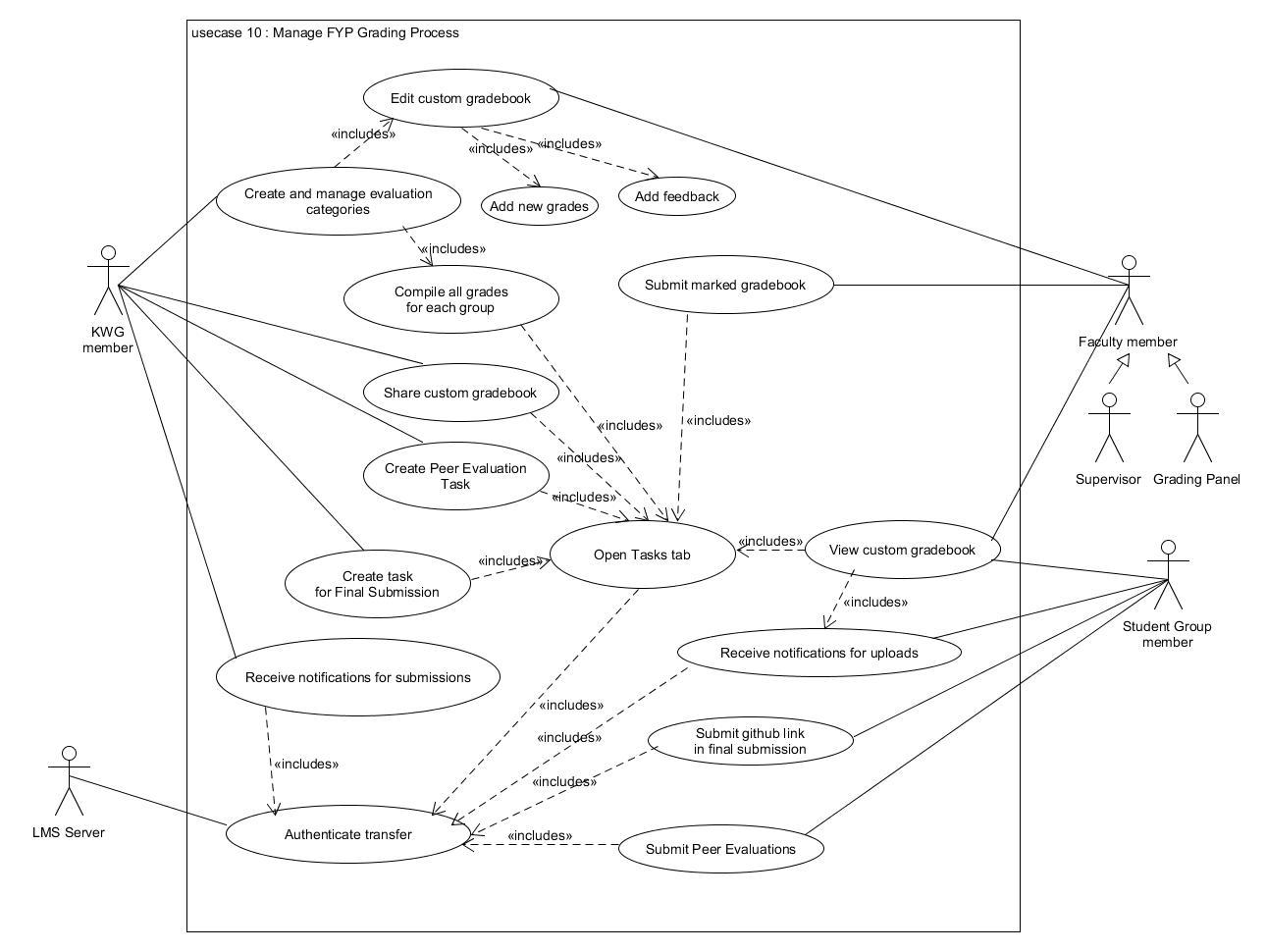
#### 3.3.10.7 Post-conditions

##### 3.3.10.7.1 Successful Completion

The grades are successfully compiled and shared with the FYP groups. The students receive their final FYP grades.

##### 3.3.10.7.2 Failure Condition

KWG will need to reschedule the evaluations and submissions.

****

****

## 

## 3.4 Classes / Objects

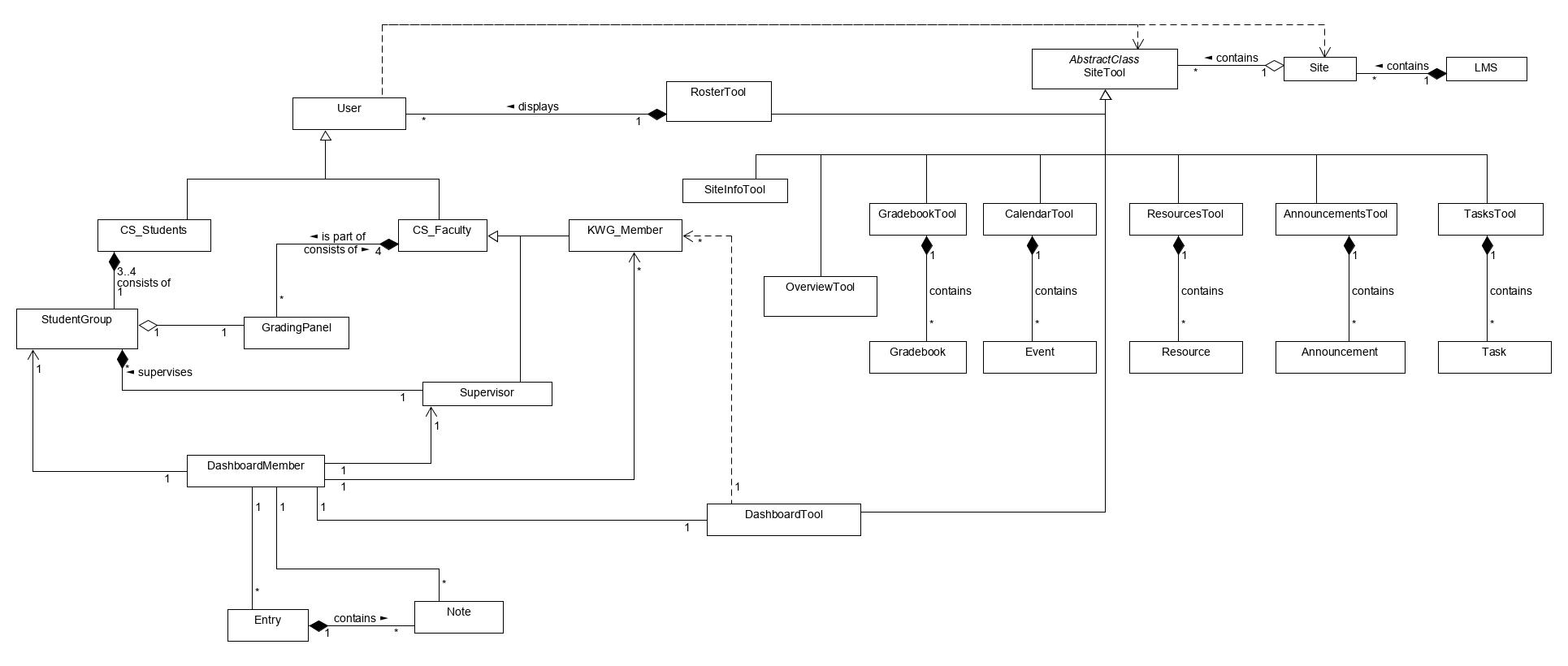


Figure 3.4.1: :Class diagram showing Class relations – UML

The above diagram showcases the relations of the classes. For a complete UML depicting the attributes and functions as well as the relations of the classes, please see the figure below (Figure 3.4.2). If it is unreadable due to quality issues, please refer to the attached jpg file (Class\_Diagram.jpg) or the snippets of the same figure (Figures 3.4.3 - 3.4.7).

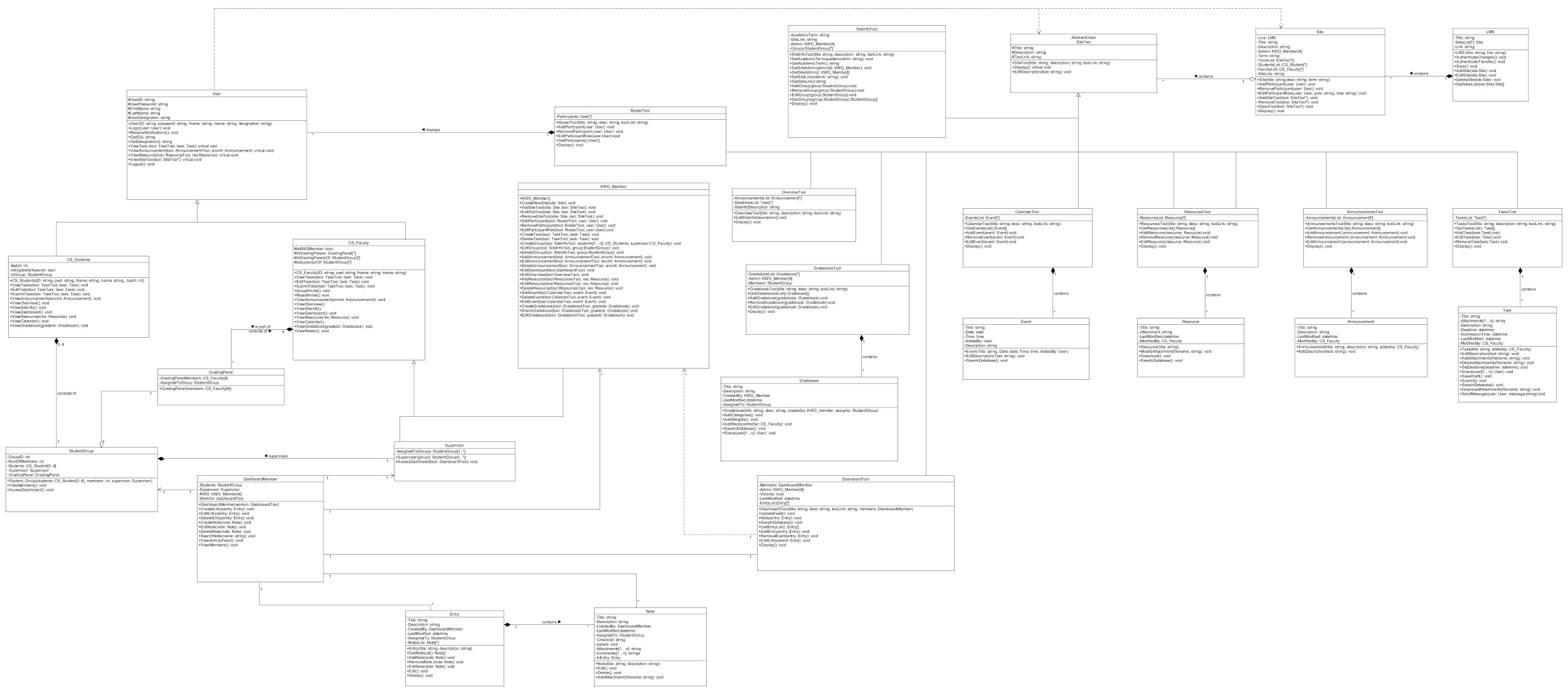
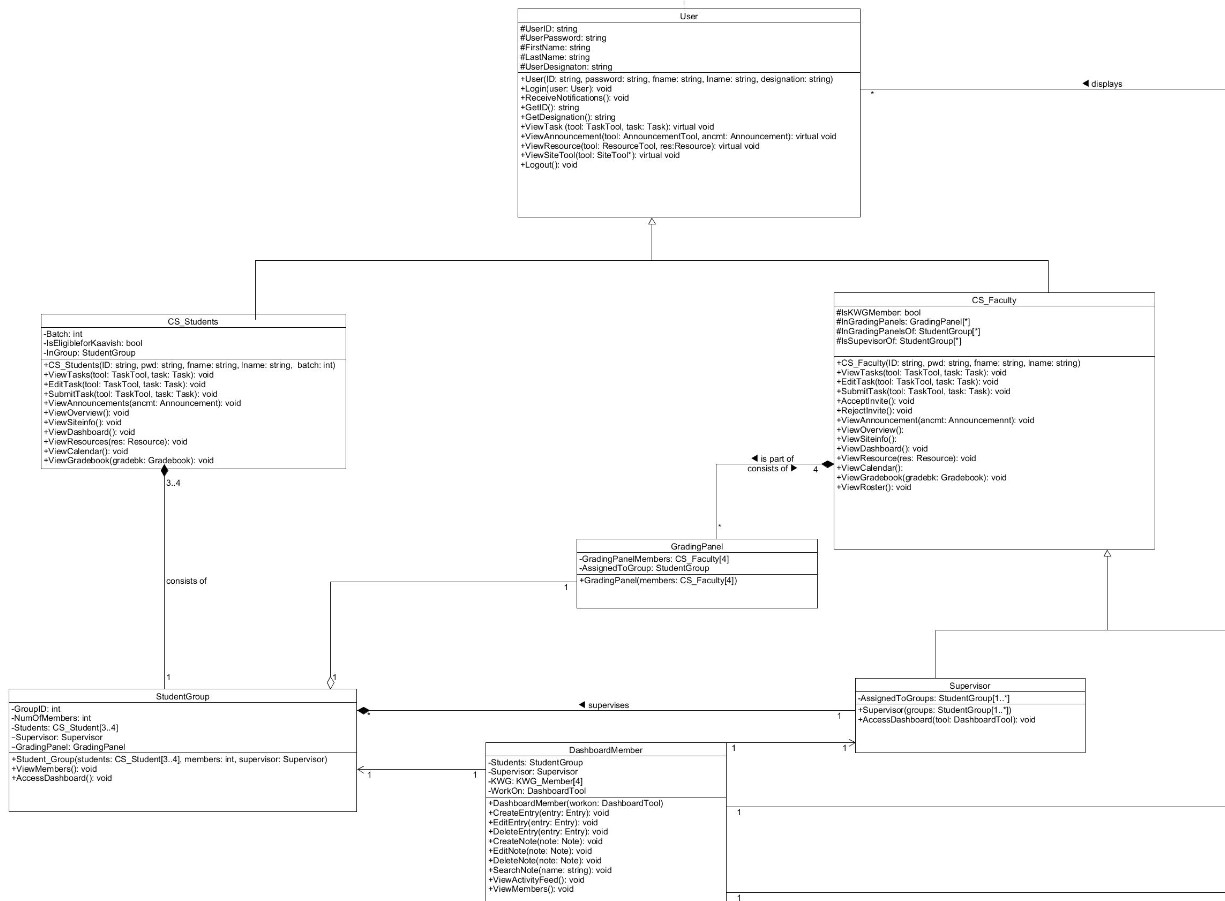


Figure 3.4.2: Class diagram – UML

The images below show close-ups of the complete UML of the Class diagram.

Figure 3.4.3: Class diagram – UML - Snippet 1

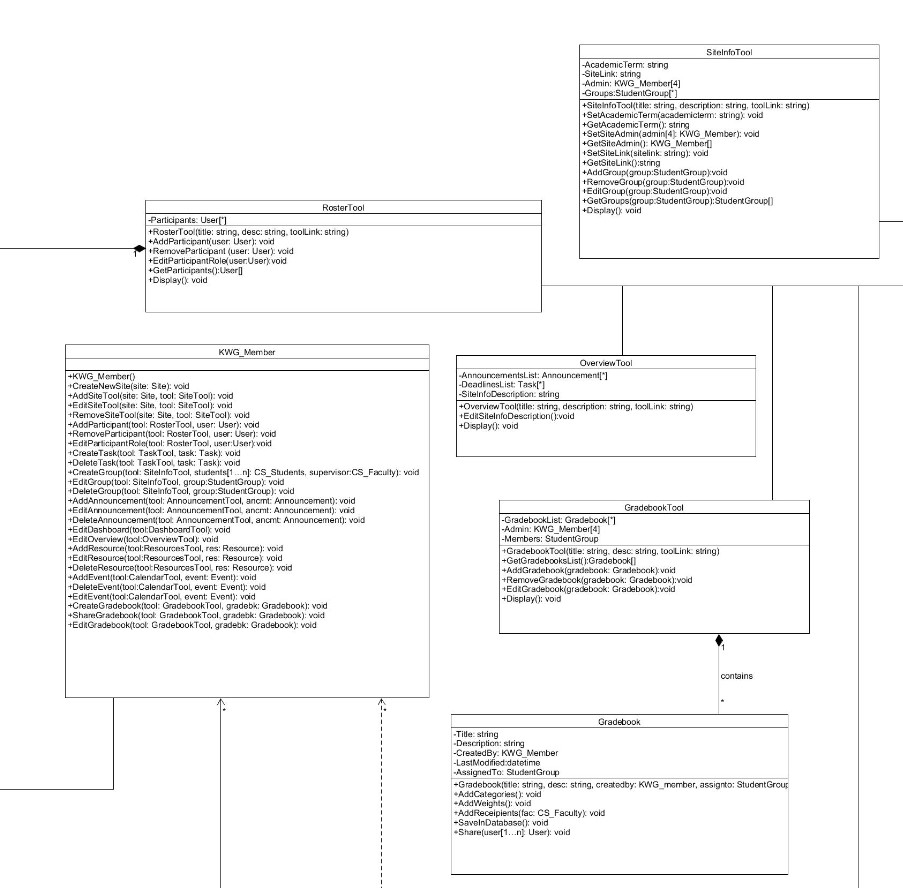


Figure 3.4.4: Class diagram – UML - Snippet 2

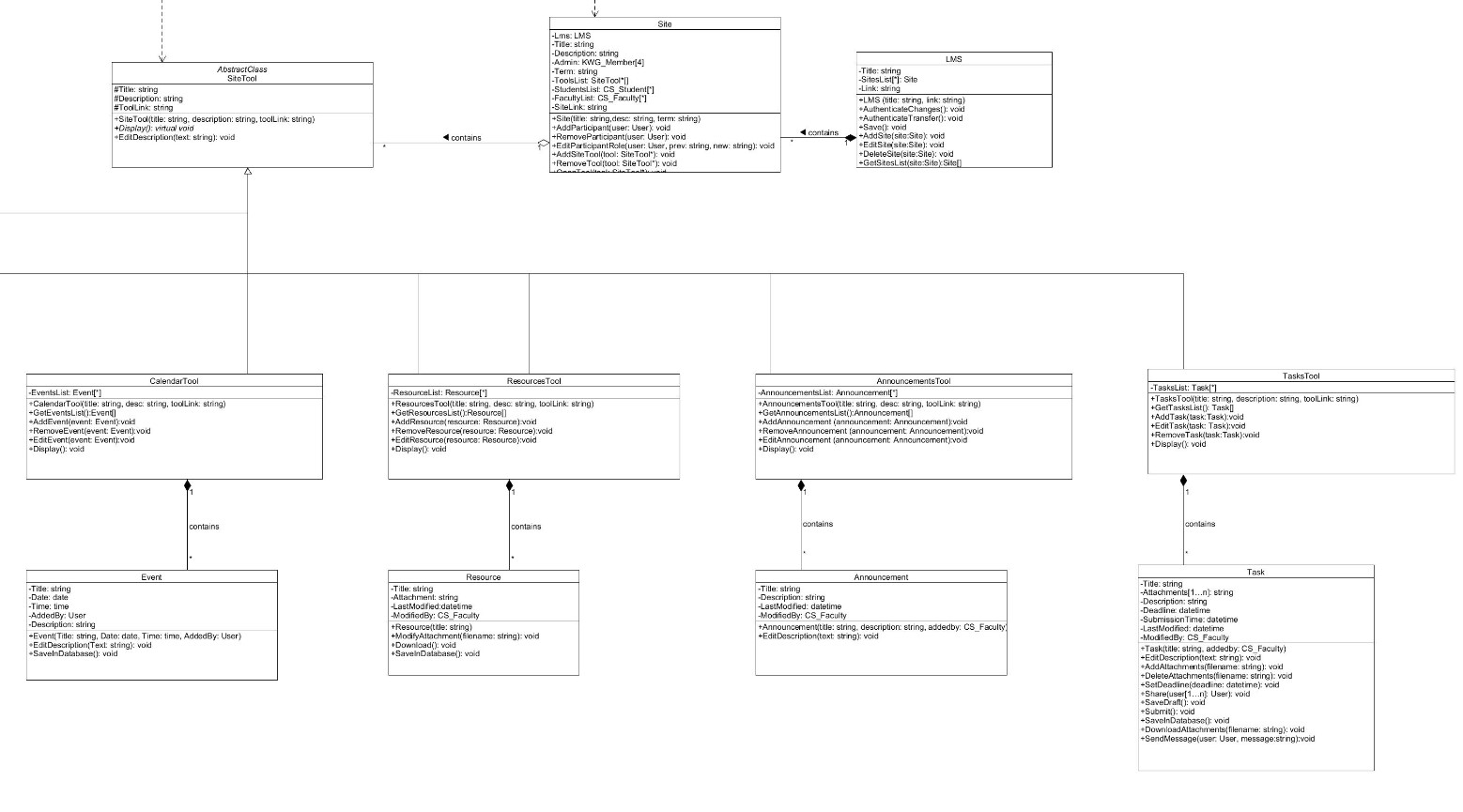


Figure 3.4.5: Class diagram – UML - Snippet 3

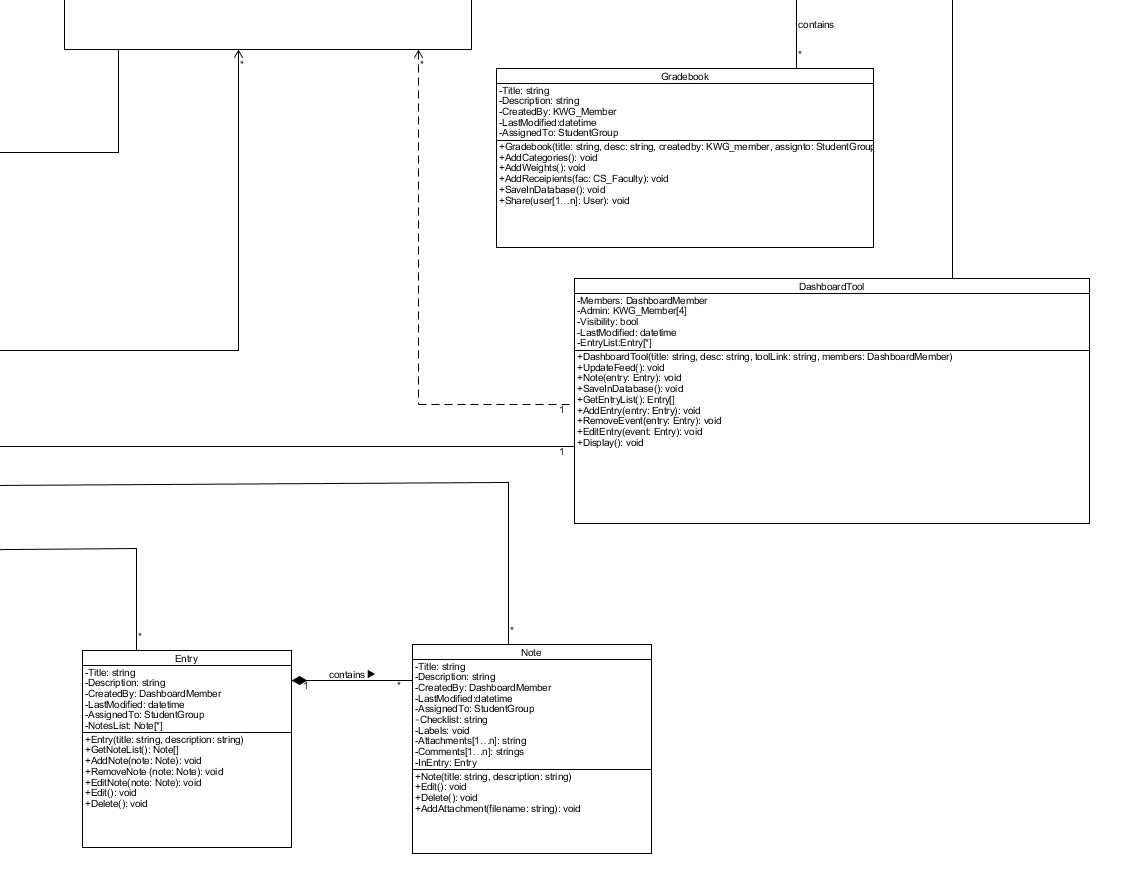


Figure 3.4.4: Class diagram – UML - Snippet 4

The classes along with their attributes and functions are also listed below:

### 3.4.1 User

3.4.1.1 Attributes

3.4.1.1.1 #UserID: string

3.4.1.1.2 #UserPassword: string

3.4.1.1.3 #FirstName: string

3.4.1.1.4 #LastName: string

3.4.1.1.5 #UserDesignaton: string

3.4.1.2 Functions

3.4.1.2.1 +User(ID: string, password: string, fname: string, lname: string, designation: string)

3.4.1.2.2 +Login(user: User): void

3.4.1.2.3 +ReceiveNotifications(): void

3.4.1.2.4 +GetID(): string

3.4.1.2.5 +GetDesignation(): string

3.4.2.2.2 +ViewTask (tool: TaskTool, task: Task): virtual void

3.4.2.2.5 +ViewAnnouncement(tool: AnnouncementTool, ancmt: Announcement): virtual void

3.4.2.2.5 +ViewResource(tool: ResourceTool, res:Resource): virtual void

3.4.2.2.6 +ViewSiteTool(tool: SiteTool\*): virtual void

3.4.1.2.6 +Logout(): void

Used in Section 3.3

### 3.4.2 CS\_Students: User

3.4.2.1 Attributes

3.4.2.1.1 #Batch: int

3.4.2.1.2 #IsEligibleforKaavish: bool

3.4.2.1.3 #InGroup: StudentGroup

3.4.2.2 Functions

3.4.2.2.1 +CS\_Students(ID: string, pwd: string, fname: string, lname: string, batch: int)

3.4.2.2.2 +EditTask(tool: TaskTool, task: Task): void

3.4.2.2.3 +SubmitTask(tool: TaskTool, task: Task):void

Used in Sections 3.3.1 and 3.3.2

### 3.4.3 StudentGroup

3.4.3.1 Attributes

3.4.3.1.1 -Students: CS\_Student[3..4]

3.4.3.1.2 -GroupID: int

3.4.3.1.3 -NumOfMembers: int

3.4.3.1.4 -Supervisor: Supervisor

3.4.3.1.5 -GradingPanel: GradingPanel

3.4.3.2 Functions

3.4.3.2.1 +StudentGroup(students: CS\_Student[3..4], members: int, supervisor: Supervisor)

3.4.3.2.2 +ViewMembers(): void

3.4.3.2.3 +AccessDashboard(): void

Used in Section 3.3.3 - 3.3.8 and 3.3.10

### 3.4.4 CS\_Faculty: User

3.4.4.1 Attributes

3.4.4.1.1 #IsKWGMember: bool

3.4.4.1.2 #InGradingPanels: GradingPanel[\*]

3.4.4.1.3 #InGradingPanelsOf: StudentGroup[\*]

3.4.4.1.4 #IsSupevisorOf: StudentGroup[\*]

3.4.4.2 Functions

3.4.4.2.1 +CS\_Faculty(ID: string, pwd: string, fname: string, lname: string)

3.4.4.2.3 +EditTask(tool: TaskTool, task: Task): void

3.4.4.2.4 +SubmitTask(tool: TaskTool, task: Task): void

3.4.4.2.5 +AcceptInvite(): void

3.4.4.2.6 +RejectInvite(): void

Used in Section 3.3

### 3.4.5 KWG\_Member: CS\_Faculty

3.4.5.1 Attributes

3.4.5.2 Functions

3.4.5.2.1 +KWG\_Member()

3.4.5.2.2 +CreateNewSite(site: Site): void

3.4.5.2.3 +AddSiteTool(site: Site, tool: SiteTool): void

3.4.5.2.4 +EditSiteTool(site: Site, tool: SiteTool): void

3.4.5.2.5 +RemoveSiteTool(site: Site, tool: SiteTool): void

3.4.5.2.6 +AddParticipant(tool: RosterTool, user: User): void

3.4.5.2.7 +RemoveParticipant(tool: RosterTool, user: User): void

3.4.5.2.8 +EditParticipantRole(tool: RosterTool, user:User):void

3.4.5.2.9 +CreateTask(tool: TaskTool, task: Task): void

3.4.5.2.10 +DeleteTask(tool: TaskTool, task: Task): void

3.4.5.2.11 +CreateGroup(tool: SiteInfoTool,students:CS\_Students[3..4], supervisor: CS\_Faculty) : void

3.4.5.2.12 +EditGroup(tool: SiteInfoTool, group:StudentGroup): void

3.4.5.2.13 +DeleteGroup(tool: SiteInfoTool, group:StudentGroup): void

3.4.5.2.14 +AddAnnouncement(tool: AnnouncementTool, ancmt: Announcement): void

3.4.5.2.15 +EditAnnouncement(tool: AnnouncementTool, ancmt: Announcement): void

3.4.5.2.16 +DeleteAnnouncement(tool: AnnouncementTool, ancmt: Announcement): void

3.4.5.2.17 +EditDashboard(tool:DashboardTool): void

3.4.5.2.18 +EditOverview(tool:OverviewTool): void

3.4.5.2.19 +AddResource(tool:ResourcesTool, res: Resource): void

3.4.5.2.20 +EditResource(tool:ResourcesTool, res: Resource): void

3.4.5.2.21 +DeleteResource(tool:ResourcesTool, res: Resource): void

3.4.5.2.22 +AddEvent(tool:CalendarTool, event: Event): void

3.4.5.2.23 +DeleteEvent(tool:CalendarTool, event: Event): void

3.4.5.2.24 +EditEvent(tool:CalendarTool, event: Event): void

3.4.5.2.25 +CreateGradebook(tool: GradebookTool, gradebk: Gradebook): void

3.4.5.2.26 +ShareGradebook(tool: GradebookTool, gradebk: Gradebook): void

3.4.5.2.27 +EditGradebook(tool: GradebookTool, gradebk: Gradebook): void

Used in Section 3.3

### 3.4.6 Supervisor: CS\_Faculty

3.4.6.1 Attributes

3.4.6.1.1 -AssignedToGroups: StudentGroup[1..\*]

3.4.6.2 Functions

3.4.6.2.1 +Supervisor(groups: StudentGroup[1..\*])

3.4.6.2.2 +AccessDashboard(tool: DashboardTool): void

Used in Section 3.3.6, 3.3.7, 3.3.9 and 3.3.10

### 3.4.7 GradingPanel

3.4.7.1 Attributes

3.4.7.1.1 -GradingPanelMembers: CS\_Faculty[4]

3.4.7.1.2 -AssignedToGroup: StudentGroup

3.4.7.2 Functions

3.4.7.2.1 +GradingPanel(members: CS\_Faculty[4])

Used in Section 3.3.8, 3.3.9 and 3.3.10

### 3.4.8 DashboardMember

3.4.8.1 Attributes

3.4.8.1.1 -Students: StudentGroup

3.4.8.1.2 -Supervisor: Supervisor

3.4.8.1.3 -KWG: KWG\_Member[4]

3.4.8.1.4 -WorkOn: DashboardTool

3.4.8.2 Functions

3.4.8.2.1 +DashboardMember(workon: DashboardTool)

3.4.8.2.2 +CreateEntry(entry: Entry): void

3.4.8.2.3 +EditEntry(entry: Entry): void

3.4.8.2.4 +DeleteEntry(entry: Entry): void

3.4.8.2.5 +CreateNote(note: Note): void

3.4.8.2.6 +EditNote(note: Note): void

3.4.8.5.7 +DeleteNote(note: Note): void

3.4.8.5.8 +SearchNote(name: string): void

3.4.8.5.9 +ViewActivityFeed(): void

3.4.8.5.10 +ViewMembers(): void

Used in Section 3.3.7

### 3.4.9 LMS

3.4.9.1 Attributes

3.4.9.1.1 -Title: string

3.4.9.1.2 -SitesList: Site[\*]

3.4.9.1.3 -Link: string

3.4.8.2 Functions

3.4.9.2.1 +LMS (title: string, link: string)

3.4.9.2.2 +AuthenticateChanges(): void

3.4.9.2.3 +AuthenticateTransfer(): void

3.4.9.2.4 +Save(): void

3.4.9.2.5 +AddSite(site:Site): void

3.4.9.2.6 +EditSite(site:Site): void

3.4.9.2.7 +DeleteSite(site:Site): void

3.4.9.2.8 +GetSitesList(site:Site):Site[]

Used in Section 3.3

### 3.4.10 SiteTool

3.4.10.1 Attributes

3.4.10.1.1 #Title: string

3.4.10.1.2 #Description: string

3.4.10.1.3 #ToolLink: string

3.4.9.2 Functions

3.4.10.2.1 +SiteTool(title: string, description: string, toolLink: string)

3.4.10.2.2 *+Display()*: virtual void

3.4.10.2.3 +EditDescription(text: string): void

Used in Section 3.3

### 3.4.11 Site

3.4.11.1 Attributes

3.4.11.1.1 -Lms: LMS

3.4.11.1.2 -Title: string

3.4.11.1.3 -Description: string

3.4.11.1.4 -Admin: KWG\_Member[4]

3.4.11.1.5 -Term: string

3.4.11.1.6 -ToolsList: SiteTool\*[]

3.4.11.1.7 -StudentsList: CS\_Student[\*]

3.4.11.1.8 -FacultyList: CS\_Faculty[\*]

3.4.11.1.9 -SiteLink: string

3.4.11.2 Functions

3.4.11.2.1 +Site(title: string, desc: string, term: string)

3.4.11.2.2 +AddParticipant(user: User): void

3.4.11.2.3 +RemoveParticipant(user: User): void

3.4.11.2.3 +EditParticipantRole(user: User, prev: string, new: string): void

3.4.11.2.6 +AddSiteTool(tool: SiteTool\*): void

3.4.11.2.7 +RemoveTool(tool: SiteTool\*): void

3.4.11.2.8 +OpenTool(tool: SiteTool\*): void

3.4.11.2.9 +Display(): void

Used in Section 3.3

### 3.4.12 OverviewTool : SiteTool

3.4.12.1 Attributes

3.4.12.1.1 -AnnouncementsList: Announcement[\*]

3.4.12.1.2 -DeadlinesList: Task[\*]

3.4.12.1.3 -SiteInfoDescription: string

3.4.12.2 Functions

3.4.12.2.1 +OverviewTool(title: string, description: string, toolLink: string)

3.4.12.2.2 +EditSiteInfoDescription(): void

3.4.12.2.3 +Display(): void

Used in Section 3.3.1

### 3.4.13 SiteInfoTool: SiteTool

3.4.13.1 Attributes

3.4.13.1.1 -AcademicTerm: string

3.4.13.1.2 -SiteLink: string

3.4.13.1.3 -Admin: KWG\_Member[4]

3.4.13.1.4 -Groups:StudentGroup[\*]

3.4.13.2 Functions

3.4.13.2.1 +SiteInfoTool(title: string, description: string, toolLink: string)

3.4.13.2.2 +SetAcademicTerm(academicterm: string): void

3.4.13.2.3 +GetAcademicTerm(): string

3.4.13.2.4 +SetSiteAdmin(admin: KWG\_Member[4]): void

3.4.13.2.5 +GetSiteAdmin(): KWG\_Member[]

3.4.13.2.6 +SetSiteLink(sitelink: string): void

3.4.13.2.7 +GetSiteLink(): string

3.4.13.2.7 +AddGroup(group:StudentGroup): void

3.4.13.2.7 +RemoveGroup(group:StudentGroup): void

3.4.13.2.7 +EditGroup(group:StudentGroup): void

3.4.13.2.7 +GetGroups(group:StudentGroup): StudentGroup[]

3.4.13.2.8 +Display(): void

Used in Section 3.3.1

### 3.4.14 TasksTool: SiteTool

3.4.14.1 Attributes

3.4.14.1.1 -TasksList: Task[\*]

3.4.10.2 Functions

3.4.14.2.1 +TasksTool(title: string, description: string, toolLink: string)

3.4.14.2.2 +GetTasksList(): Task

3.4.14.2.3 +AddTask(task:Task): void

3.4.14.2.4 +EditTask(task: Task): void

3.4.14.2.5 +RemoveTask(task:Task): void

3.4.14.2.6 +Display(): void

Used in Sections 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.8, 3.3.9, 3.3.10

### 3.4.15 ResourcesTool: SiteTool

3.4.15.1 Attributes

3.4.15.1.1 -ResourceList: Resource[\*]

3.4.15.2 Functions

3.4.15.2.1 +ResourcesTool(title: string, desc: string, toolLink: string)

3.4.15.2.2 +GetResourcesList(): Resource[]

3.4.15.2.3 +AddResource(resource: Resource): void

3.4.15.2.4 +RemoveResource(resource: Resource): void

3.4.15.2.5 +EditResource(resource: Resource): void

3.4.15.2.6 +Display(): void

Used in Section 3.3.8

### 3.4.16 AnnouncementsTool: SiteTool

3.4.16.1 Attributes

3.4.16.1.1 -AnnouncementsList: Announcement[\*]

3.4.16.2 Functions

3.4.16.2.1 +AnnouncementsTool(title: string, desc: string, toolLink: string)

3.4.16.2.2 +GetAnnouncementsList():Announcement[]

3.4.16.2.3 +AddAnnouncement (announcement: Announcement): void

3.4.16.2.4 +RemoveAnnouncement (announcement: Announcement): void

3.4.16.2.5 +EditAnnouncement (announcement: Announcement): void

3.4.16.2.6 +Display(): void

Used in Section 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.8, 3.3.9, 3.3.10

### 3.4.17 GradebookTool: SiteTool

3.4.17.1 Attributes

3.4.17.1.1 -GradebookList: Gradebook[\*]

3.4.17.1.2 -Admin: KWG\_Member[4]

3.4.17.1.3 -Members: StudentGroup

3.4.17.2 Functions

3.4.17.2.1 +GradebookTool(title: string, desc: string, toollink: string)

3.4.17.2.2 +GetGradebooksList(): Gradebook[]

3.4.17.2.3 +AddGradebook(gradebook: Gradebook): void

3.4.17.2.4 +RemoveGradebook(gradebook: Gradebook): void

3.4.17.2.5 +EditGradebook(gradebook: Gradebook): void

3.4.17.2.6 +Display(): void

Used in Sections 3.3.1, 3.3.8, 3.3.9 and 3.3.10

### 3.4.18 RosterTool: SiteTool

3.4.18.1 Attributes

3.4.18.1.1 -Participants: User[\*]

3.4.18.2 Functions

3.4.18.2.1 +RosterTool(title: string, desc: string, toolLink: string)

3.4.18.2.2 +AddParticipant(user: User): void

3.4.18.2.3 +RemoveParticipant (user: User): void

3.4.18.2.1 +EditParticipantRole(user:User): void

3.4.18.2.1 +GetParticipants(): User[]

3.4.18.2.4 +Display(): void

Used in Section 3.3.1

### 3.4.19 CalendarTool: SiteTool

3.4.19.1 Attributes

3.4.19.1.1 -EventsList: Event[\*]

3.4.19.2 Functions

3.4.19.2.1 +CalendarTool(title: string, desc: string, toolLink: string)

3.4.19.2.2 +GetEventsList(): Event[]

3.4.19.2.3 +AddEvent(event: Event): void

3.4.19.2.4 +RemoveEvent(event: Event): void

3.4.19.2.5 +EditEvent(event: Event): void

3.4.19.2.2 +Display(): void

Used in Sections 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8, 3.3.10

### 3.4.20 DashboardTool: SiteTool

3.4.20.1 Attributes

3.4.20.1.1 -Members: DashboardMember

3.4.20.1.2 -Admin: KWG\_Member[4]

3.4.20.1.3 -Visibility: bool

3.4.20.1.4 -LastModified: datetime

3.4.20.1.6 -EntryList: Entry[\*]

3.4.20.2 Functions

3.4.20.2.1 +DashboardTool(title: string, desc: string, toolLink: string, members:DashboardMember)

3.4.20.2.2 +UpdateFeed(): void

3.4.20.2.3 +Note(entry: Entry): void

3.4.20.2.6 +SaveInDatabase(): void

3.4.20.2.7 +GetEntryList():Entry

3.4.20.2.8 +AddEntry(entry: Entry): void

3.4.20.2.9 +RemoveEvent(entry: Entry): void

3.4.20.2.11 +EditEntry(event: Entry): void

3.4.20.2.11 +Display(): void

Used in Section 3.3.1 and 3.3.7

### 3.4.21 Task

3.4.21.1 Attributes

3.4.21.1.1 -Title: string

3.4.21.1.2 -Attachments: string[\*]

3.4.21.1.3 -Description: string

3.4.21.1.4 -Deadline: datetime

3.4.21.1.5 -SubmissionTime: datetime

3.4.21.1.6 -LastModified: datetime

3.4.21.1.7 -ModifiedBy: CS\_Faculty

3.4.21.2 Functions

3.4.21.2.1 +Task(title: string, addedby: CS\_Faculty)

3.4.21.2.2 +EditDescription(text: string): void

3.4.21.2.3 +AddAttachments(filename: string): void

3.4.21.2.4 +DeleteAttachments(filename: string): void

3.4.21.2.5 +SetDeadline(deadline: datetime): void

3.4.21.2.6 +Share(user: User[1..\*]): void

3.4.21.2.7 +SaveDraft(): void

3.4.21.2.8 +Submit(): void

3.4.21.2.9 +SaveInDatabase(): void

3.4.21.2.10 +DownloadAttachments(filename: string): void

3.4.21.2.11 +SendMessage(user: User, message:string):void

Used in Sections 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.8, 3.3.9, 3.3.10

### 3.4.22 Announcement

3.4.22.1 Attributes

3.4.22.1.1 -Title: string

3.4.22.1.2 -Description: string

3.4.22.1.3 -LastModified: datetime

3.4.22.1.4 -ModifiedBy: CS\_Faculty

3.4.22.2 Functions

3.4.22.2.1 +Announcement(title: string, description: string, addedby: CS\_Faculty)

3.4.22.2.2 +EditDescription(text: string): void

Used in Section 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.8, 3.3.9, 3.3.10

### 3.4.23 Resource

3.4.23.1 Attributes

3.4.23.1.1 -Title: string

3.4.23.1.2 -Attachment: string

3.4.23.1.3 -LastModified:datetime

3.4.23.1.4 -ModifiedBy: CS\_Faculty

3.4.23.2 Functions

3.4.23.2.1 +Resource(title: string)

3.4.23.2.2 +ModifyAttachment(filename: string): void

3.4.23.2.3 +Download(): void

3.4.23.2.4 +SaveInDatabase(): void

Used in Section 3.3.8

### 3.4.24 Gradebook

3.4.24.1 Attributes

3.4.24.1.1 -Title: string

3.4.24.1.2 -Description: string

3.4.24.1.3 -CreatedBy: KWG\_Member

3.4.24.1.4 -LastModified:datetime

3.4.24.1.5 -AssignedTo: StudentGroup

3.4.24.2 Functions

3.4.24.2.1 +Gradebook(title: string, desc: string, createdby: KWG\_Member, assignto: StudentGroup)

3.4.24.2.1 +AddCategories(): void

3.4.24.2.2 +AddWeights(): void

3.4.24.2.3 +AddRecipients(faculty: CS\_Faculty): void

3.4.24.2.4 +SaveInDatabase(): void

3.4.24.2.5 + Share(user: User[1..\*]): void

Used in Sections 3.3.8, 3.3.9 and 3.3.10

### 3.4.25 Note

3.4.25.1 Attributes

3.4.25.1.1 -Title: string

3.4.25.1.2 -Description: string

3.4.25.1.3 -CreatedBy: DashboardMember

3.4.25.1.4 -LastModified: datetime

3.4.25.1.5 -AssignedTo: StudentGroup

3.4.25.1.6 -Checklist: string

3.4.25.1.7 -Labels: void

3.4.25.1.8 -Attachments: string[\*]

3.4.25.1.9 -Comments: strings[\*]

3.4.25.1.10 -InEntry: Entry

3.4.25.2 Functions

3.4.25.2.1 +Note(title: string, description: string)

3.4.25.2.2 +Edit(): void

3.4.25.2.3 +Delete(): void

3.4.25.2.4 +AddAttachment(filename: string): void

Used in Sections 3.3.7

### 3.4.26 Entry

3.4.26.1 Attributes

3.4.26.1.1 -Title: string

3.4.26.1.2 -Description: string

3.4.26.1.3 -CreatedBy: DashboardMember

3.4.26.1.4 -LastModified: datetime

3.4.26.1.5 -AssignedTo: StudentGroup

3.4.26.1.6 -NotesList: Note[\*]

3.4.24.2 Functions

3.4.26.2.1 +Entry(title: string, description: string)

3.4.26.2.2 +GetNoteList(): Note[]

3.4.26.2.3 +AddNote(note: Note): void

3.4.26.2.4 +RemoveNote (note: Note): void

3.4.26.2.5 +EditNote(note: Note): void

3.4.26.2.6 +Edit(): void

3.4.26.2.7 +Delete(): void

Used in Sections 3.3.7

### 3.4.27 Event

3.4.27.1 Attributes

3.4.27.1.1 -Title: string

3.4.27.1.2 -Date: date

3.4.27.1.3 -Time: time

3.4.27.1.4 -AddedBy: User

3.4.27.1.5 -Description: string

3.4.27.2 Functions

3.4.27.2.1 +Event(title: string, date: date, time: time, addedby: User)

3.4.27.2.2 +EditDescription(text: string): void

3.4.27.2.4 +SaveInDatabase(): void

Used in Sections 3.3.2, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8, 3.3.9, 3.3.10

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

All interactions on LMS, including uploading files, sending reminders, etc. would not take more than 3 seconds. In order to select a number of people to send a message to, the user can simply check-mark the checkbox in front of their name and/or ID. Sending a reminder, alert, message or submission would take only one click. If the Internet connection is disrupted, the user would be informed.

### 3.5.2 Reliability

Since we will be extending the already-present LMS, the system would be 100% reliable, and the search results all accurate.

### 3.5.3 Availability

Given that there is network availability at all times, LMS will be available 99% of the time. It can access its database only if there is an Internet connection.

### 3.5.4 Security

The feedback and submissions can only be viewed by the parties they are shared with. All communications are encrypted, and no one can access anyone else’s account due to accounts being password protected.

### 3.5.5 Maintainability

Since there is the existing code for LMS which we’ll extend, changes would be easy to incorporate as compared to making the software from scratch. Furthermore, any errors would be easier to fix on our part.

### 3.5.6 Portability

The system can be run from all platforms; Android, iOS and Windows.

## 3.6 Design Constraints

The design constraints would be the same as in the existing software. No additional changes would be made.

## 3.7 Logical Database Requirements

LMS currently uses the server-side database. All acceptable file formats and size are mentioned in the Assignments section. Data would be saved there and can be accessed any time until removed by faculty.

## 3.8 Other Requirements

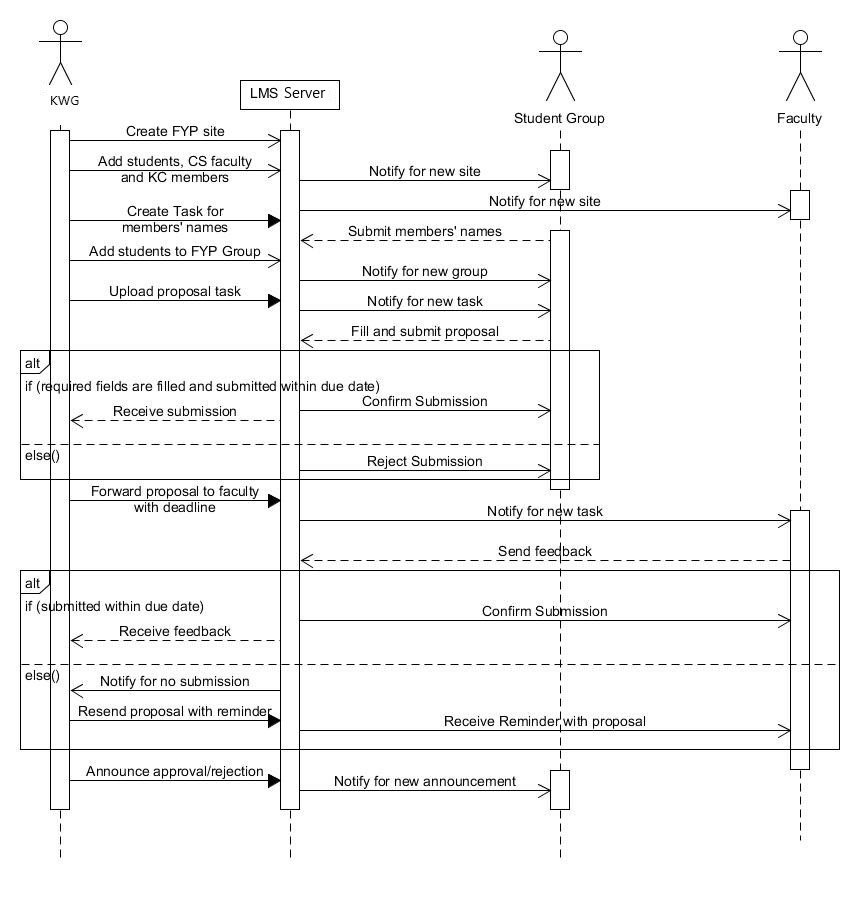
Other than the mentioned requirements in the previous sections, this software will also require active internet connection as it is an extension of LMS website.

# 4. Analysis Models

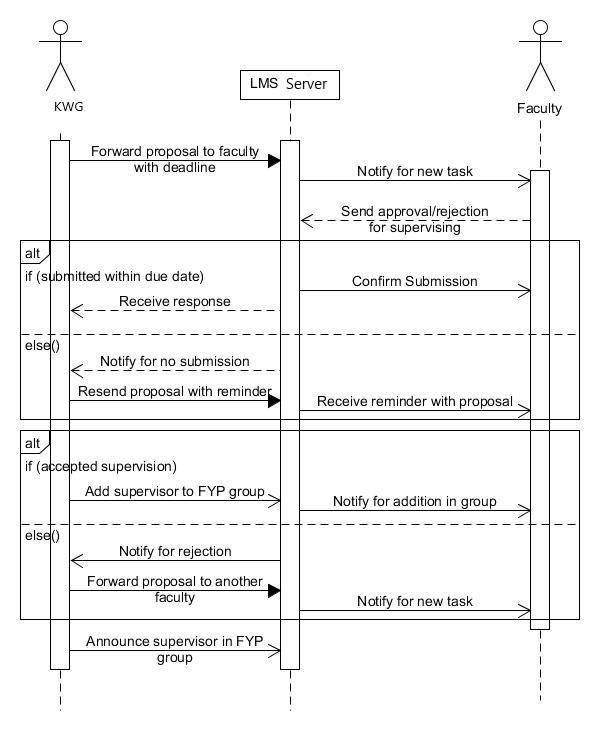
## 

## 4.1 Sequence Diagrams

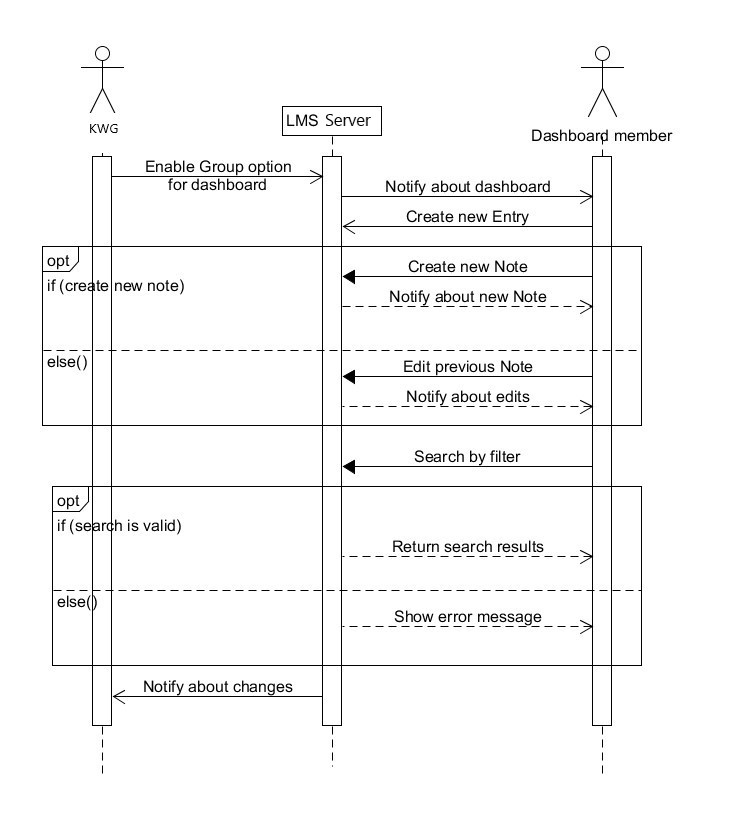
**4.1.1.** The first sequence to occur will start when KWG creates FYP site on LMS and adds CS faculty, Kaavish enrolled students and KWG members to it. KWG then uploads the proposal for the student group to submit. KWG decides to approve/reject the proposal based on faculty’s feedback, which they then announce to the student group on LMS.

****

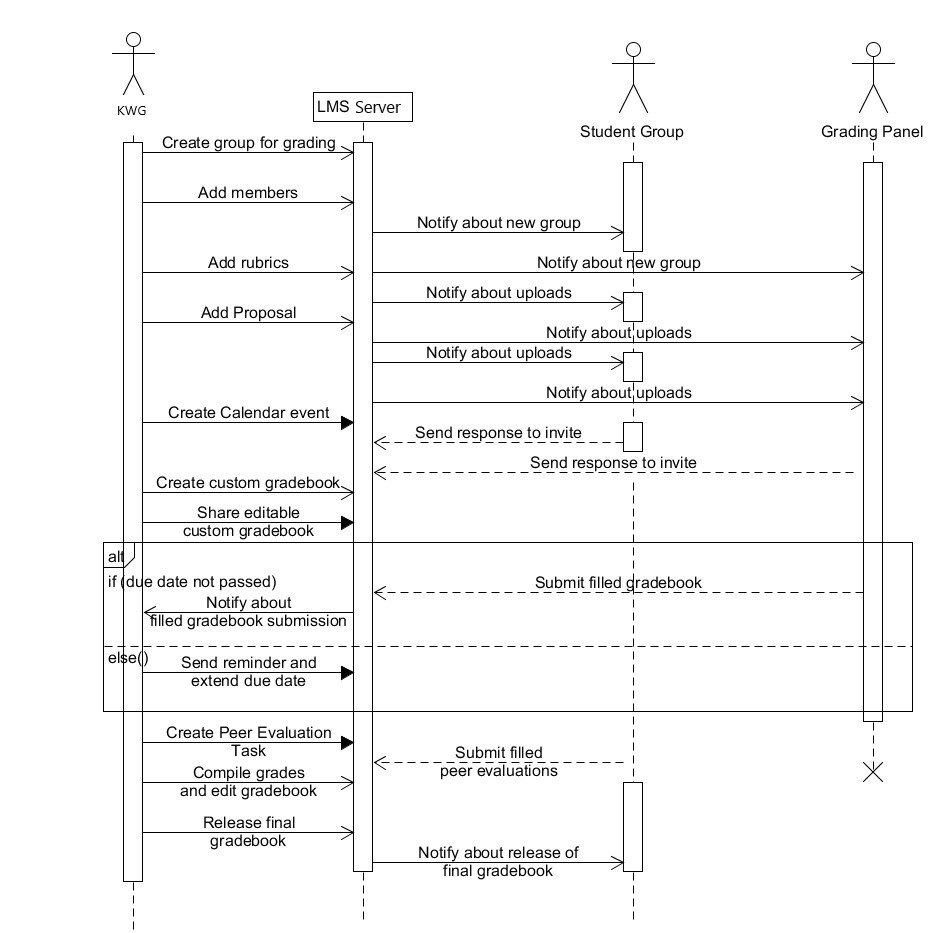
**4.1.2.** In the second sequence, KWG forwards the proposal to faculty suitable for supervising a student group. The faculty either accepts or rejects the role of supervisor. If accepted, KWG adds them to the FYP group on LMS.



**4.1.3.** In this sequence, KWG enables the dashboard for Student groups and Supervisor. Members of the dashboard could be students( from FYP groups) or faculty (KWG member or Supervisor). The members of the dashboard are then able to interact with the dashboard (make new entries, notes or edit previous ones or search through them).



**4.1.4.** In this sequence, KWG creates a group on lms for the grading process. KWG adds the concerned faculty and students. KWG forwards the unfilled gradebook to Grading panel members, and the peer evaluations to the students. Once the gradebook is filled by the Grading panel, KWG edits in the peer evaluations and compiles the gradebook and publishes it for the Student group to view.



# Appendices

## A.1 Appendix 1

As requested by the client, we are making use of already existing features of LMS. Some of the most basic amongst those features are:

* Creating groups on LMS and adding participants ([How do I create Groups?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728831-how-do-i-create-groups))
* Group submissions ([How do I enable group submissions for an assignment?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728544-how-do-i-enable-group-submissions-for-an-assignment))
* Sending an attachment with a message ([How do I send a message?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728684-how-do-i-send-a-message))
* Replying to a message ([How do I reply to a message?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728688-how-do-i-reply-to-a-message))
* Providing feedback on submissions and grading them ([How do I grade an assignment?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728550-how-do-i-grade-an-assignment))
* Adding deadlines and events on the calendar ([How do I add items to the Calendar?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728538-how-do-i-add-items-to-the-calendar))
* Adding a gradebook tab ([How do I set up my Gradebook?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728611-how-do-i-set-up-my-gradebook))
* Editing gradebook ([How do I enter and/or edit grades in the Gradebook?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728617-how-do-i-enter-and-or-edit-grades-in-gradebook))
* Adding an announcement ([How do I add an Announcement?](https://sakai.screenstepslive.com/s/sakai_help/m/68426/l/728514-how-do-i-add-an-announcement))

One major addition to LMS from our side would be creating a dashboard tab. The description of this tab is explained in detail in section 3.2.17.

## A.2 Appendix 2

In our initial meeting with the client, some issues regarding the current method of FYP management were raised by the client. Those issues were:

* Too many platforms for different jobs, namely Outlook, Trello, LMS. The client found it difficult to go back and forth between these platforms.
* Different Excel sheets for faculty to give feedback to multiple groups. It gets difficult for the faculty to keep all files consistent and up to date.
* No proper platform to keep track of students’ FYP progress.
* Grading done on Google sheets, which is not the best way to record data.

What the client wanted us to do was to integrate as much of these functions as we could into the latest version of LMS. He wanted a system like Trello to keep track of each groups’ weekly progress. He strongly suggested not to make the software from scratch, but only to build upon the existing version.

## 