CIS 9490 FINAL PROJECT
FALL 2019

Jin Young Chung
Madhuri Goenk
Allie Liu
Viola (Yijia) Cai
Jordon Yan

Current System (AS-IS):

I. Source documents, including screen dumps, from the website.

Please see attachment.

II. A brief description of functionality provided by the website.

a. Announcement

Announcement is one way instructors can communicate time sensitive information critical to the course with students, such as due dates of assignments and homework, exam schedule, correction and clarification of materials. When the instructors post an announcement, students will get an email notification about the announcement and can view the announcement even without log into the Blackboard system.

b. Course Content

Students can access various format of course content uploaded by instructors, such as PDF files, PowerPoint slides, text, images, multimedia, surveys and links via Blackboard. The files can be grouped in different folders chosen by instructor, such as by week, by projects. Instructors can also choose to upload the course content ahead of time and make the content available to students later.

c. Assignment

- Assignment dissemination

  Instructor can create an assignment in Blackboard and manage the grades and feedback for each student. The assignment function allows students to see the due date and any material related to the materials if the instructors choose to upload any.
- Assignment submission

  Students can upload files to web apps that run in the cloud such as OneDrive, Dropbox and Google drive. Files are stored on secure, online servers where they're protected from accidents and viruses, you can instantly connect to multiple web apps where you store files.

d. Grades

Students can view the grades of enrolled courses via Blackboard. Students will get a notification in the Blackboard system that a new grade has been posted and go to "My Grades" and view the grades. Under "My grade" page, students can sort the grade by recently added or by course. There's also a feedback function with grading, the instructor can leave a comment about the assignment submission in the grading page. Blackboard allows instructors to configure the way they want to structure the grade, the function of grade modifiers allows instructors to curve students grade in either directions.

e. Group
Instructors can create group sections on Blackboard where students can collaborate, share documents, discuss ideas and create group homepage. Once created, groups can be formed by manual enrollment, random enrollment and self-enrollment based on how instructors set up. A group assignment can be submitted through 'assignment.'

III. A problem statement (Systems Proposal)

a. Cluttered web design taxonomy
Taxonomy is the science of categorization, of things based on a predetermined system. In reference to Blackboard website, the taxonomy is the way it organizes its data into categories and subcategories. For example, there are separate folders for "Syllabus" and "Course Documents", which is unnecessary for user experience. Another example is students can only view a course grade under "My Grades", it would be easier if students can view the grades of one course under that course instead of clicking away to another page.

b. Too many functions enabled
There are dozens of functions enabled in Blackboard right now, but most users only use about 20% of them. The transaction process should both enable and constrain functions. It is Blackboard supplier's job to offer as many functions as possible, but Baruch doesn't have to enable all of them. With limited training resources offered to the faculty, enabling all functions in backboard will confuse users and push them to stay away.

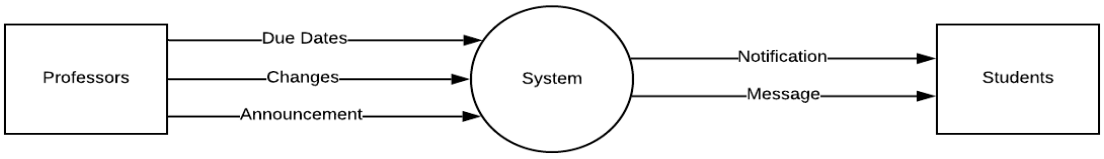
c. Integration from CUNYFirst to Blackboard is not Real Time or Near Real Time
While some of the functions can be posted to CUNYfirst to Blackboard instantaneously such as Grades, not all functions are. For course enrollment, it could take 24 - 48 hours to be transmitted from CUNYfirst to Blackboard, which can potentially be a big problem for students in the early stage of the semester.

d. Inconsistent fonts and colors
The current Blackboard system content is very inconsistent in terms of text font and text color and has random bold text all over the website which makes it harder for user to focus and navigate through the system and contribute to poor user experience.

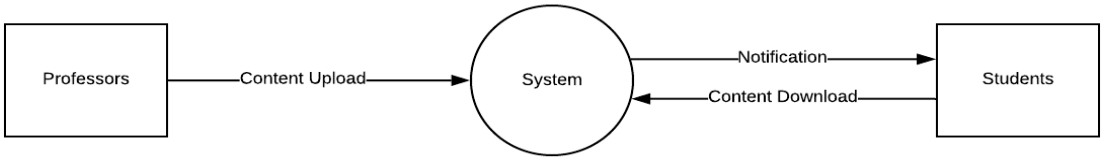
e. Group: self-enroll function
If a student enrolls in a multiple group by mistake, he/she cannot get out of any of the groups. He/she has to let the instructor know about situation and ask the instructor to kick him/her out from the groups which he/she does not mean to get in.

IV. **Context Diagrams**

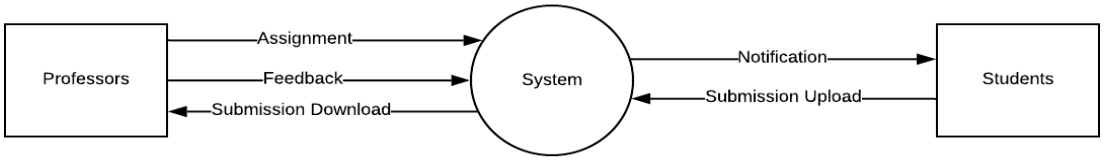
Announcement function



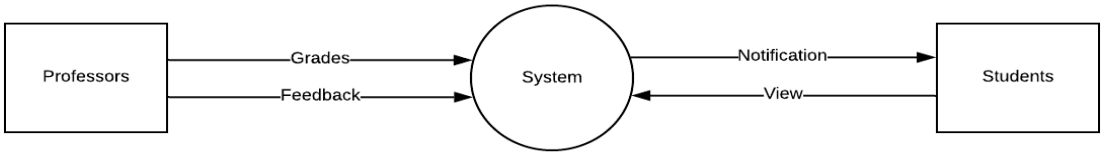
Course Content function



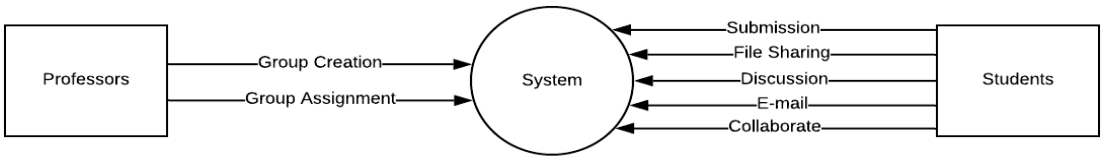
Assignment function



Grades function

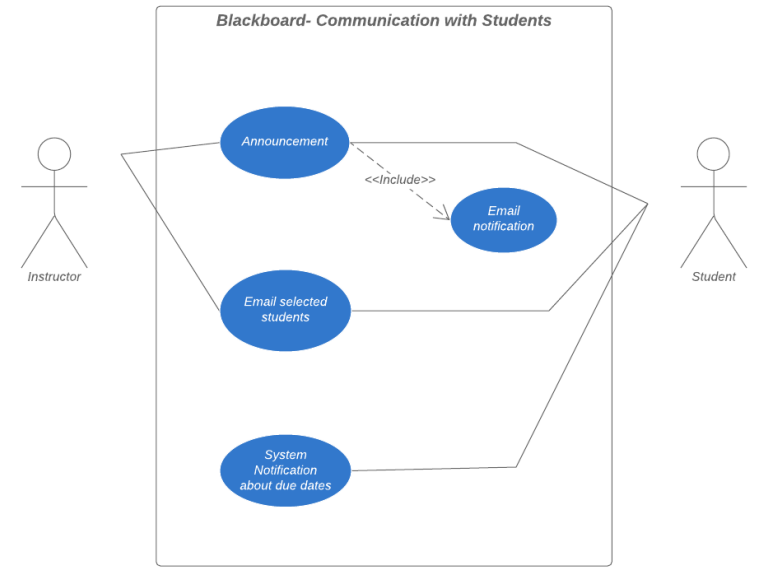


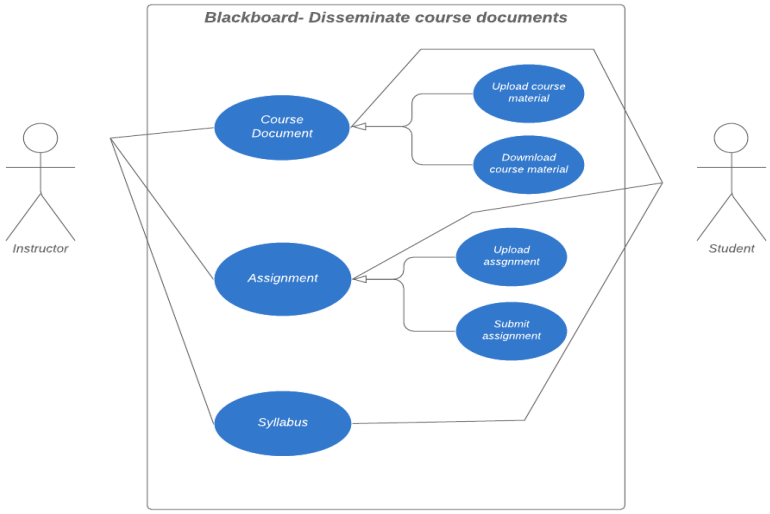
Group function

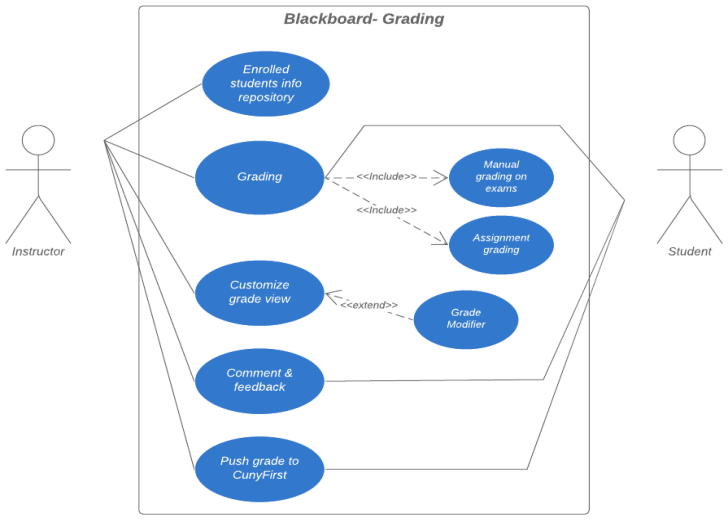


V. Process Model: Use-Case Diagrams for critical business processes

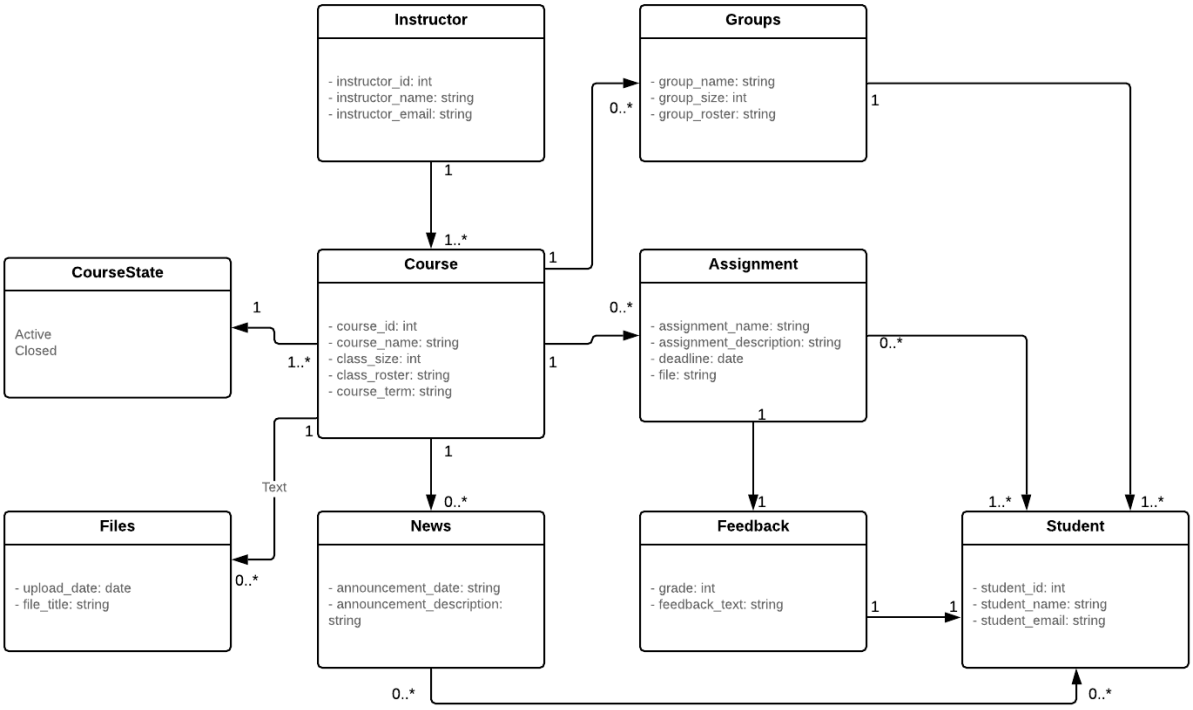
We have identified the critical business process of BlackBoard system to be communication with students, disseminate course documents and grading. Below are the Use-Case Diagrams for these three critical business processes.





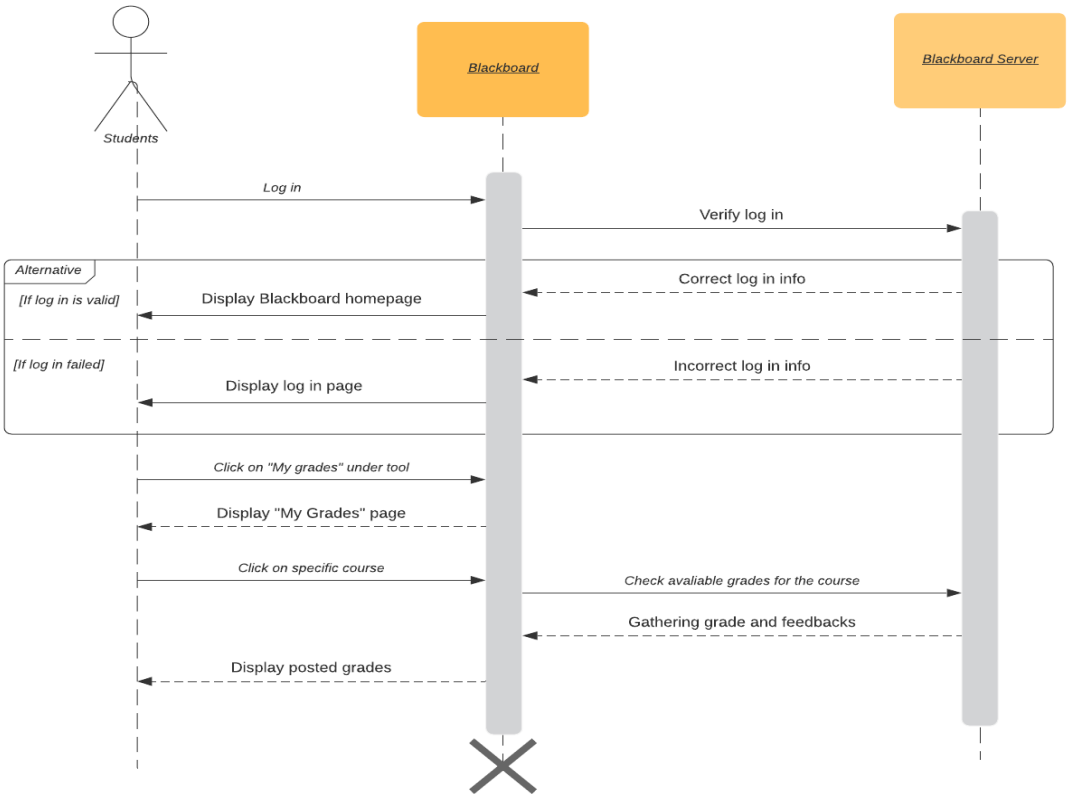


VI. Data Model: A class diagram



VII. Object Behavior Model: A Sequence Diagram for the major Use Case.

As we have discussed there are three critical business processes, communication; course material dissemination and grading. Below is a sequence diagram for the grading use case that shows the sequence of how a student can access their grade information via Blackboard.



VIII. Documentation

Please see attached.

Proposed System (TO-BE):

I. Proposal:

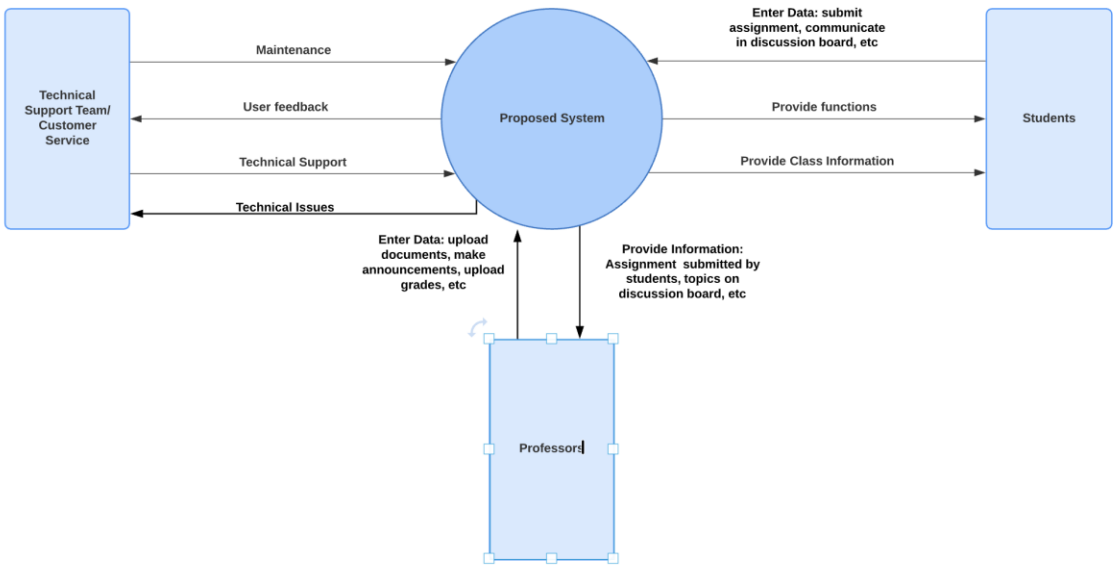
a. Differences between the current system and proposed system:

- Blackboard offers a lot of functions on the main page (the page that students see once they log in) but students usually use only a few of them. The proposed system will have limited functions listed on the main page. It makes the main page look more accessible and reduces the clutter. Students mainly use the system to get access to their class and related information like study materials, announcements and grades.

b. Problems we eliminate in the new system:
    - Eliminate the duplicated announcements - Often when using Blackboard, it is seen that announcement notifications get duplicated. The duplicate notifications are not soothing to the eye and are often distracting.
    - Consistent fonts and colors in the proposed system - The Home page of Blackboard is not consistent with its color scheme and sizing.

c. Improvements we add into the proposed system:
    - The new system allows students to use google doc and google slides link for assignment submission and presentation (With the old system only Calendar can be integrated with Google Calendar). It reduces duplication of effort by integrating systems.
    - The new system will offer students' academic calendars that show the deadlines for projects, tests, assignments and finals. Once professor posts a deadline, it will automatically get updated on the calendar.
    - Better group system - Groups under Blackboard can be beneficial for the students as it allows better integration and helps with communication between teams. Some students may have classes with a different group in each class. The group function can be used to differentiate each group and set group targets/goals with respect to each group.

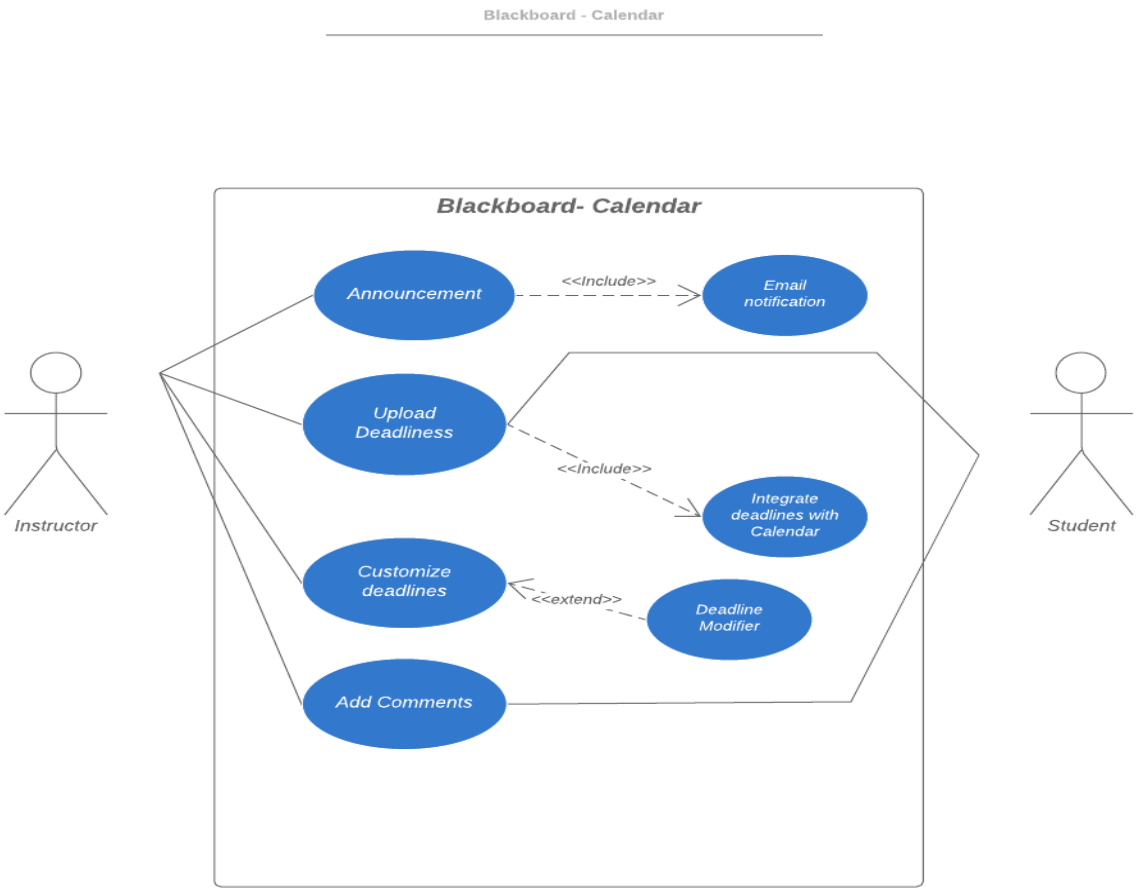
II. A Context Diagram



Context diagram is a high-level map of a system and its surrounding environment which doesn't show internal processes. In the context diagram, the proposed system is in the context bubble, and the external entities include students, professors, and technical support team. For the technical support team's data flows, the technical support team is in charge of daily maintenance of the system and provides technical support to the users when there are any technical issues. The proposed system provides user feedback to the technical support team as well. For students' data

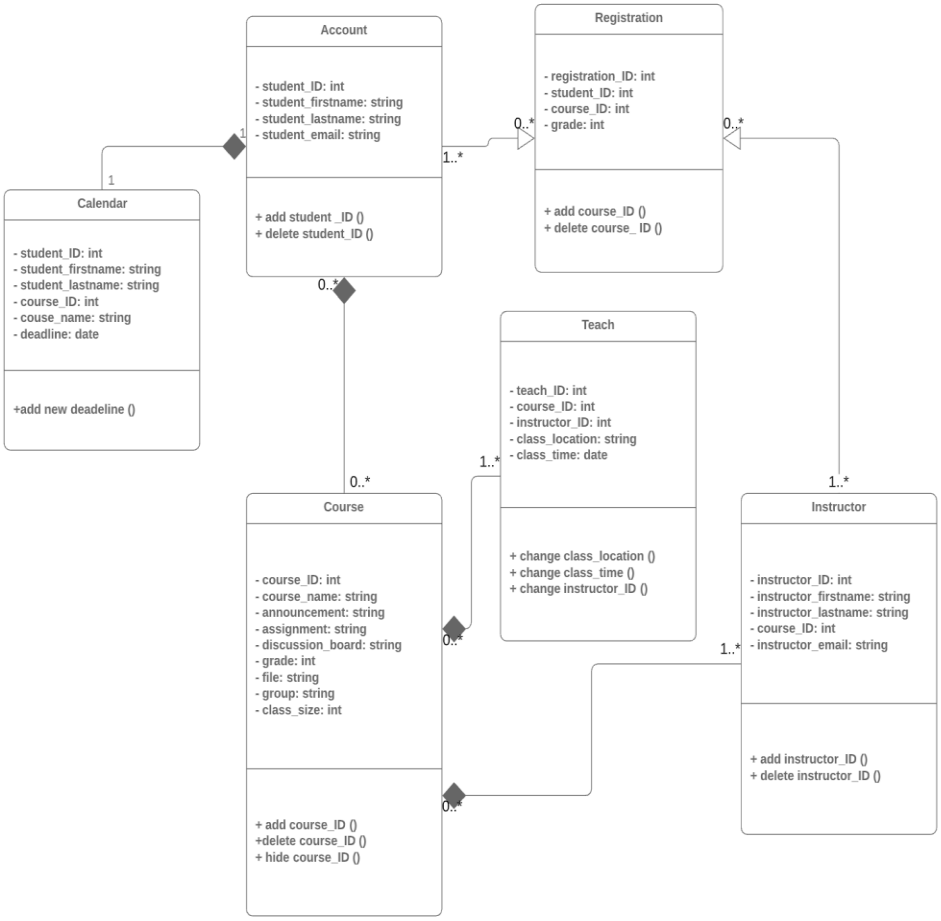
flows, students enter data to the proposed system by submitting assignments and communicating with other students in the discussion board, and the system will provide functions such as group and email to students and provide class information such as class documents and announcements to students. For professor's data flows, professors enter data to the proposed system. For example, professors can upload class materials and update grades to the system. Meanwhile, the proposed system provides professors necessary information such as students' assignments.

III. Process Model: Use-Case Diagram for critical business processes



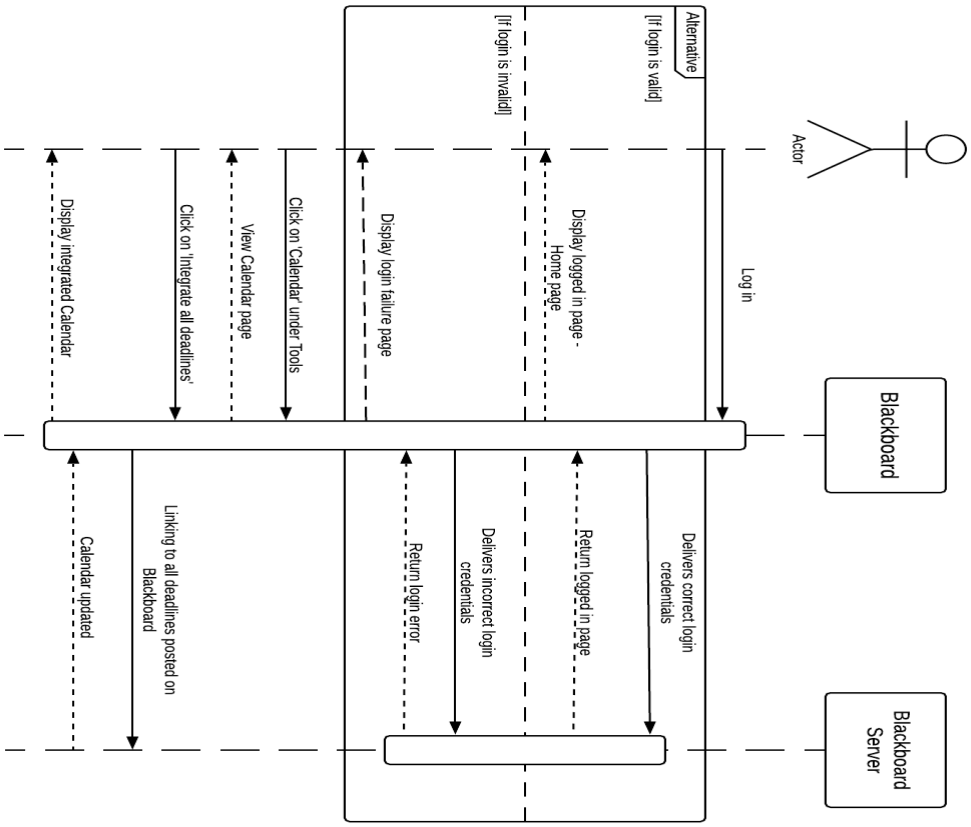
Use case diagram for the proposed system includes integration of deadlines with student calendar once the instructor/professor posts new deadlines.

IV. Data Model: A class diagram



In the proposed diagram, there are six classes: account, calendar, registration, course, teach and instructor. The relationships between these classes and the multiplicities are shown in the class diagram. Each class has its attributes and operations. For example, the attributes of account include student_ID, student_firstname, student_lastname, sutdent_email and its operations include adding student_ID and deleting student_ID.

V. Object Behavior Model: A Sequence Diagrams for the major Use Case.



The sequence diagram depicts interaction between user and objects. In this case it can be noted that Blackboard now allows the option to integrate deadlines for assignments, projects, tests and homework. The diagram above depicts the sequence of actions to be taken when integrating the Calendar with the deadlines.

VI. Documentation of all data used in the above models (a repository [a file folder is acceptable])

Please see attached.

VII. database design

Tables: attributes

Account: Student_ID (pk), Student_Firstname, Student_Lastname, Student_Email
Course: Course_ID (pk), Course_Name, Announcement, Assignment, Discussion_Board, Grade, File, Group, Class_Size
Calendar: Student_ID, Student_Firstname, Student_Lastname, Course_ID, Course_Name, Deadline
Registration: Registration_ID (pk), Student_ID (fk), Course_ID, Grade
Instructor: Instructor_ID (pk), Instructor_Firstname, Instructor_Lastname, Course_ID (fk), Instructor_Email
Teach: Teach_ID, course ID (fk), Instructor_ID (fk), Class_Location, Class_Time

Note: Table headings are highlighted on the left and attributes are mentioned on the right.
pk - Primary Key, fk - Foreign Key

VIII. Object design:

➢ **Creating a method to prevent posting duplicate announcement on BlackBoard with pre-condition (contract). (Created "class" object and attributes)**

```
class Course: #creating a class object
    Course_ID = 0
    Course_Name = ""
    Announcement = ""
    Assignment = ""
    Discussion_Board = ""
    Grade = ""
    File = ""
    Group = ""
    Class_Size = 0
    #above are attributes with rules in class object

    def __init__(self, courseID, courseName, announcement, assignment, discussion, grade, file, group, size):
        #Creating a constructor
        self.Course_ID = courseID
        self.Course_Name = courseName
        self.Announcement = announcement
        self.Assignment = assignment
        self.Discussion_Board = discussion
        self.Grade = grade
        self.File = file
        self.Group = group
        self.Class_Size = size

announcements = []
for row in Course:
    b = Course(row[2])
    announcements.append(b) #create a list of announcements

#defining a function for the method
ann = input("Input announcement: ")
def DuplicateAnnouncement(self):
    for announcement in announcements:
        assert len(ann) !=0, "Please input your message."
        #Pre-condition: if no letter in new announcement input, then break (Contract)
        continue
    assert ann not in announcement, "Duplicate announcement"
    return announcements.append(ann)
    #if announcement already exist, break. If not, create new announcement
```

➢ **Creating a method to calculate the deadline from the newly created calendar function with pre-condition (contract). (Created “class” object and attributes.)**

```
1
2 class Calendar: #Creating a class object
3     Student_ID = 0
4     Student_Firstname = ""
5     Student_Lastname = ""
6     Course_ID = 0
7     Course_name = ""
8     Deadline = "%m/%d/%Y"  # mm/dd/yyy format
9     #above are attributes with rule in class object
10
11    def __init__(self, studentId, firstname, lastname, courseId, coursename, deadline):
12        #Creating a constrctor
13        self.Student_ID = studentId
14        self.Student_Firstname = firstname
15        self.Student_Lastname = lastname
16        self.Course_ID = courseId
17        self.Course_name = coursename
18        self.Deadline = deadline
19
20 from datetime import date
21 today = date.today()
22 d1 = today.strftime("%m/%d/%Y") # mm/dd/yyy format
23 print("d1 - ", d1)
24
25 #defining a function for the method
26 def Untildeadline(self):
27     assert d1 <= self.Deadline, "Your assignment is past due!"
28         continue
29     #Pre-condition: if deadline is passed, break
30
31     if d1 == self.Deadline:
32         print("Submission due today")
33     elif d1 < self.Deadline:
34         print("You have ", d1-self.Deadline, " day(s) to submit.")
```

➢ **Creating a method to prevent duplicate enrollment in group function with pre-condition (contract). (Created “class” object and attributes.)**

```
1 studentid = Account.Student_ID #from Account Table
2
3 class Course: #creating a class object
4     Course_ID = 0
5     Course_Name = ""
6     Announcement = ""
7     Assignment = ""
8     Discussion_Board = ""
9     Grade = ""
10    File = ""
11    Group = ""
12    Class_Size = 0
13    #above are attributes with rules in class object
14
15    def __init__(self, courseID, courseName, announcement, assignment, discussion, grade, file, group, size):
16        #Creating a constructor
17        self.Course_ID = courseID
18        self.Course_Name = courseName
19        self.Announcement = announcement
20        self.Assignment = assignment
21        self.Discussion_Board = discussion
22        self.Grade = grade
23        self.File = file
24        self.Group = group
25        self.Class_Size = size
26
27 groups = []
28 for row in Course:
29     b = Course(row[7])
30     groups.append(b) #creating a list of groups
31
32 students = []
33 students.append(studentid) #creating a list of students
34
35 #Creating a function for the method
36 def DuplicateGroupEnroll(self):
37     for student in students:
38         assert student.Group != "", "No Group Name"
39         continue
40         #Contract: if no group name, then break
41
42         if student.Group in groups:
43             print("Can't enroll in more than one group.")
```

IX. Controls

a. Data Type Control with constraints

We assign data types to each attributes as one of the controls to control input data on each table. This will promote the data quality and ensure visibility of data.

Account	
Student_ID	int() NOT NULL
Student_Firstname	varchar(35)
Student_Lastname	varchar(55)
Student_Email	varchar(200)
Instructor	
Instructor_ID	int() NOT NULL
Instructor_Firstname	varchar(35)
Instructor_Lastname	varchar(55)
Course_ID	int()
Instructor_Email	varchar(200)

b. Fault Communication

We programmed the object design model to communicate with user when user input has invalid or fault value.

```
for announcement in announcements:
    assert len(ann) != 0, "Please input your message."

============================================================
grade = input("Grade: ")

def AvgGrade:
    assert grade >= 0, "Invalid input: Grade should be greater than 0"

============================================================
Class_Size = input("Size: ")

if Class_Size < 0:
    print("Class size must be greater than 0")
elif Class_Size > 300:
    print("Class size cannot be greater than 300")

============================================================
location = input("location: ")

building = [VC, LX]

if location[0:2] not in building:
    print("Please enter valid building name: ")
```

c. Unit Testing

Due to the limited time and scope of the project, we can only complete up to unit testing. Our unit testing is mainly conducted on the class level, which includes, but not limited to, the constructors

and deconstructs. Our testing also includes testing on flow control, code analysis, and peer code review. Although unit testing cannot guarantee for error-free on system-level, this testing will ensure that each block of code in the program will work independently from each other. Moreover, the testing will detect and prevent errors which will, in turn, reduce software development risks, time, and costs.