INSY 5341 – 002 Analysis and Design

Spring 2018

Final Project Report on:

# Information System for a High School

**Team Members**

Oak, Aditya

Ramesh, Shiv Balaji

Yamsani, Santhosh

## 1. Introduction

In an Educational Institution where there are a growing number of students, teaching and non-teaching staffs, administration and management of the business processes becomes difficult. Simple day to day business routines take a lot of time and hence becomes cumbersome. More importantly there are overwhelming amounts of data which are not organized. All the data are not available to the entire organization all the time and hence data is underutilized as they are in silos. It becomes hard to record and maintain other data such as personal data, academic records etc. in paper.

Hence these Educational Institutions need a one stop solution for all their business problems. They need a web-based solution for managing their day to day routines. Our Information System provides the same thing. Our Information Systems has the following basic functionalities…

1. It will facilitate student enrollment and registration. Students will be able to enroll and register in courses.
2. It will also allow faculties to post the grades of homework and exams along with the feedback online.
3. This information systems will also include student feedback system where the students can rate and give feedback about their professors.

All the above functionalities will speed up the overall business processes. It will save a lot of time, money, effort spent in paperwork. It will also establish a health communication between the different stakeholders of the system.

## 2. System Request

- **Project Sponsor**: The Chairperson of the school.

- **Business need**:
  - Enrolling and registration for classes requires students to line-up in the office which causes space and time problems. This creates a lot of inconvenience to the students as they must allocate an entire day for this simple process.
  - Also having the grades of all the students for all the homework and exams on paper causes unwanted hassle. Students need to meet their professors for getting their grades and the feedback about their performance in the exam. A lot of time is wasted in this process.
  - At the end of the semester, feedback about the professors are usually obtained from students on paper. These feedbacks are usually not stored for a longer period. They are left unconnected with the past ratings and the papers are usually lost. This defeats the whole purpose of getting the feedback from the students as the feedbacks are neither reviewed nor decisions or actions are made.

  - This information systems solves all the above problems.

- **Business requirements**:
  - This information system is designed to facilitate easy enrollment and registration of students in the courses.
  - It also enables faculties to post the grades along with the feedback of homework and exams online.
  - It is also built to support student feedback system. That is, students can rate and give feedback about their professors at the end of the semester.

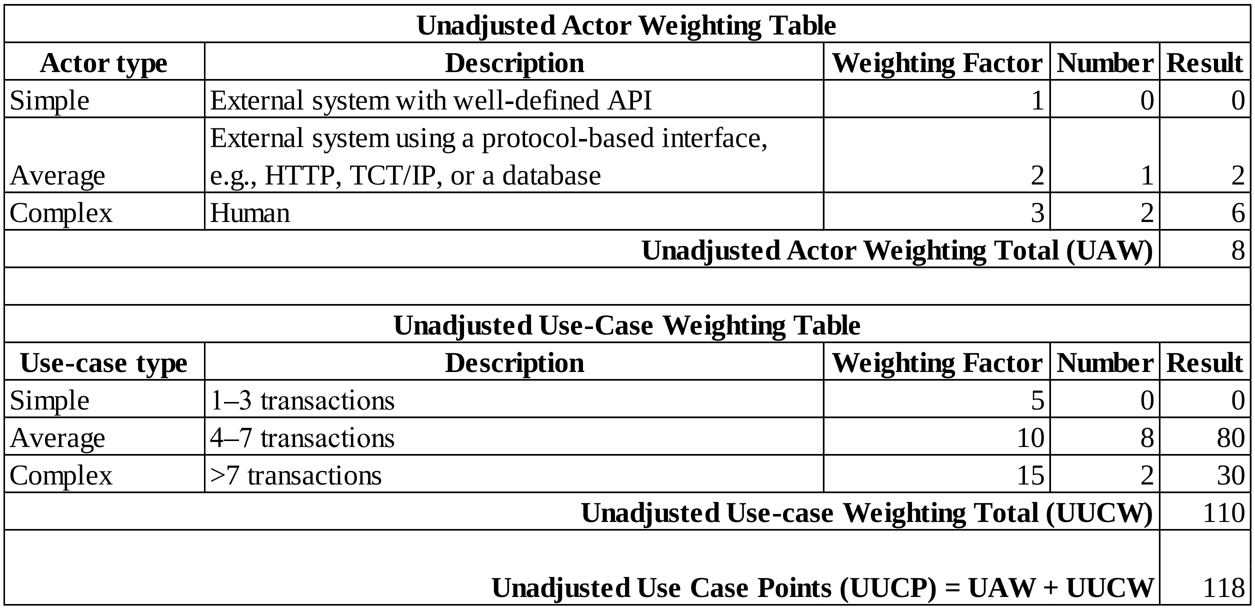
- **Business value**: This information system has the following business value
  - It establishes a healthy communication between all the stakeholders of the system. Thus, the different departments of the school are now more connected.

  - It will help reduce paper records as we go digital. This saves a lot of time, money and effort that is spent in maintaining the paperwork.
  - Having everything digital allows us to connect everything and perform a detailed analysis. This will in turn unlock new and interesting insights which will help in improving the business of the school.
  - Overall, many business processes are speeded up and the organization functions, efficiently as a whole instead of working and storing information in silos.

- **Special issues**: The Drain on the system - too many process running at the same time, though using the proper number of servers. The system may fail, because of too many students logging in and using the same process at the same time. This may slow down the connection or cause a failure to login to the system temporarily.

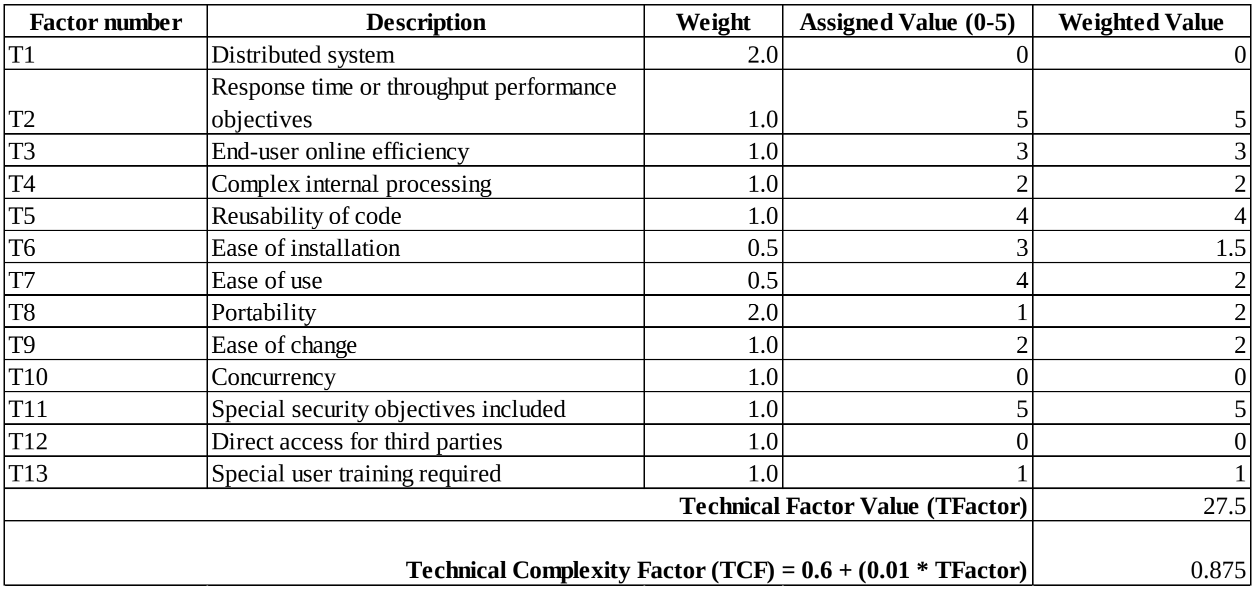
## 3. Effort Estimation

### Unadjusted Weighting Table:



| Unadjusted Actor Weighting Table | | | | |
|---|---|---|---|---|
| **Actor type** | **Description** | **Weighting Factor** | **Number** | **Result** |
| Simple | External system with well-defined API | 1 | 0 | 0 |
| Average | External system using a protocol-based interface, e.g., HTTP, TCT/IP, or a database | 2 | 1 | 2 |
| Complex | Human | 3 | 2 | 6 |
| | | | **Unadjusted Actor Weighting Total (UAW)** | 8 |
| | | | | |
| **Unadjusted Use-Case Weighting Table** | | | | |
| **Use-case type** | **Description** | **Weighting Factor** | **Number** | **Result** |
| Simple | 1–3 transactions | 5 | 0 | 0 |
| Average | 4–7 transactions | 10 | 8 | 80 |
| Complex | >7 transactions | 15 | 2 | 30 |
| | | | **Unadjusted Use-case Weighting Total (UUCW)** | 110 |
| | | | **Unadjusted Use Case Points (UUCP) = UAW + UUCW** | 118 |

### Technical Complexity Factor:



| Factor number | Description | Weight | Assigned Value (0-5) | Weighted Value |
|---|---|---|---|---|
| T1 | Distributed system | 2.0 | 0 | 0 |
| T2 | Response time or throughput performance objectives | 1.0 | 5 | 5 |
| T3 | End-user online efficiency | 1.0 | 3 | 3 |
| T4 | Complex internal processing | 1.0 | 2 | 2 |
| T5 | Reusability of code | 1.0 | 4 | 4 |
| T6 | Ease of installation | 0.5 | 3 | 1.5 |
| T7 | Ease of use | 0.5 | 4 | 2 |
| T8 | Portability | 2.0 | 1 | 2 |
| T9 | Ease of change | 1.0 | 2 | 2 |
| T10 | Concurrency | 1.0 | 0 | 0 |
| T11 | Special security objectives included | 1.0 | 5 | 5 |
| T12 | Direct access for third parties | 1.0 | 0 | 0 |
| T13 | Special user training required | 1.0 | 1 | 1 |
| | | | **Technical Factor Value (TFactor)** | 27.5 |
| | | | **Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor)** | 0.875 |

## Environmental Complexity Factor and Final Estimation of Effort:

| Factor number | Description | Weight | Assigned Value (0-5) | Weighted Value |
|---|---|---|---|---|
| E1 | Familiarity with system development process being used | 1.5 | 4 | 6 |
| E2 | Application experience | 0.5 | 3 | 1.5 |
| E3 | Object-oriented experience | 1.0 | 4 | 4 |
| E4 | Lead analyst capability | 0.5 | 5 | 2.5 |
| E5 | Motivation | 1.0 | 5 | 5 |
| E6 | Requirements stability | 2.0 | 4 | 8 |
| E7 | Part-time staff | -1.0 | 1 | -1 |
| E8 | Difficulty of programming language | -1.0 | 3 | -3 |
| **Environmental Factor Value (EFactor)** | | | | 23 |
| **Environmental Factor (EF) = 1.4 + (-0.03 * EFactor)** | | | | 0.71 |
| **Adjusted Use Case Points (UCP) = UUCP * TCF * ECF** | | | | 73.3075 |
| **Effort in person-hours = UCP * PHM** | | | | 1466.15 |

## 4. Feasibility Analysis

### Technical Feasibility:

→ **Familiarity with the information system**

- Since there are a few organization who are already in the same business doing things differently, the workforce involved in this project are familiar and have great knowledge about the business aspects of the system. The system analysts and the developers are very familiar with the functional area of the project.

→ **Technologies used**

- The system will be built in Android Studio. Android Studio and the IDE used are very sophisticated, hence helping in designing the system easily and on time. Coming to the database, ORACLE database is used for the project. The employees are aware and familiar with these technologies.

→ **Project size**

- Project size is relatively small, since the functionalities of the system are limited. In the future the present system can be enhanced by expanding the functionalities of the system.

→ **Compatibility**

- This system is perfectly compatible with the old system. The old system has personal data of the students and faculties. It also has the academic credentials of the students. This information need to be integrated into the new system. Hence the new system does not have any significant compatibility issues.

Hence from the above analysis, the technicality of the project doesn't pose any serious threats and the project is less risky.

### Economic Feasibility:

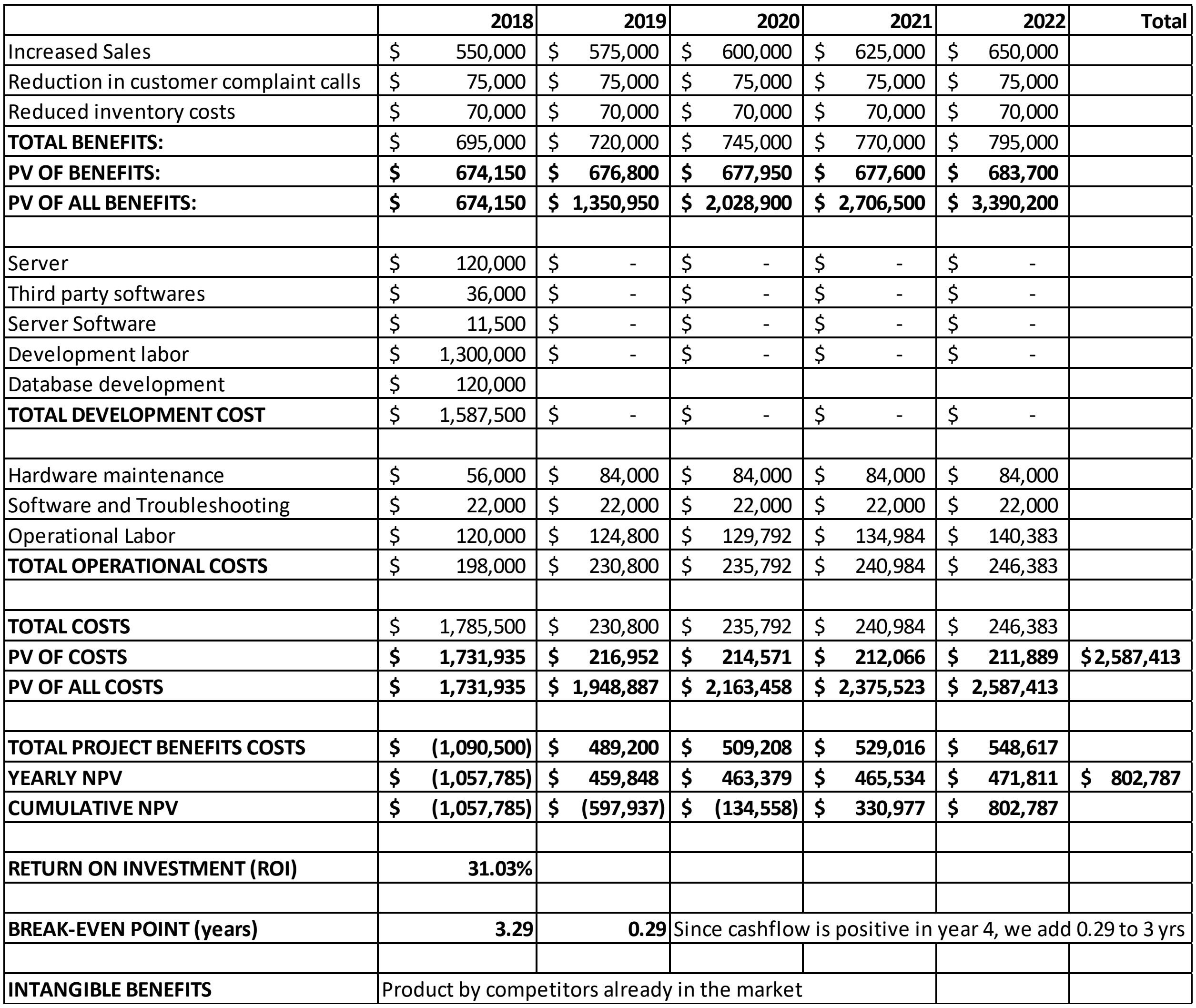
Costs and benefits have been identified along with their values. Cash Flow, Net Present Value, Return on Investment have been determined. Break-Even point has been determined and has been graphed.

From the cost benefit analysis given below for this information system we can say that, the financial risk associated with the project is very low. The benefits outweigh the initial costs in a short period of time.

### Intangible costs and Benefits

- Significant reduction in time, money and effort spent in paperwork.
- Better handling of the data.
- The communication between different departments is enhanced and hence the school is now more connected.
- Many business processes in the organization is speeded up.
- Having everything digital results in detailed analysis which in turn unlocks new and interesting business insights.

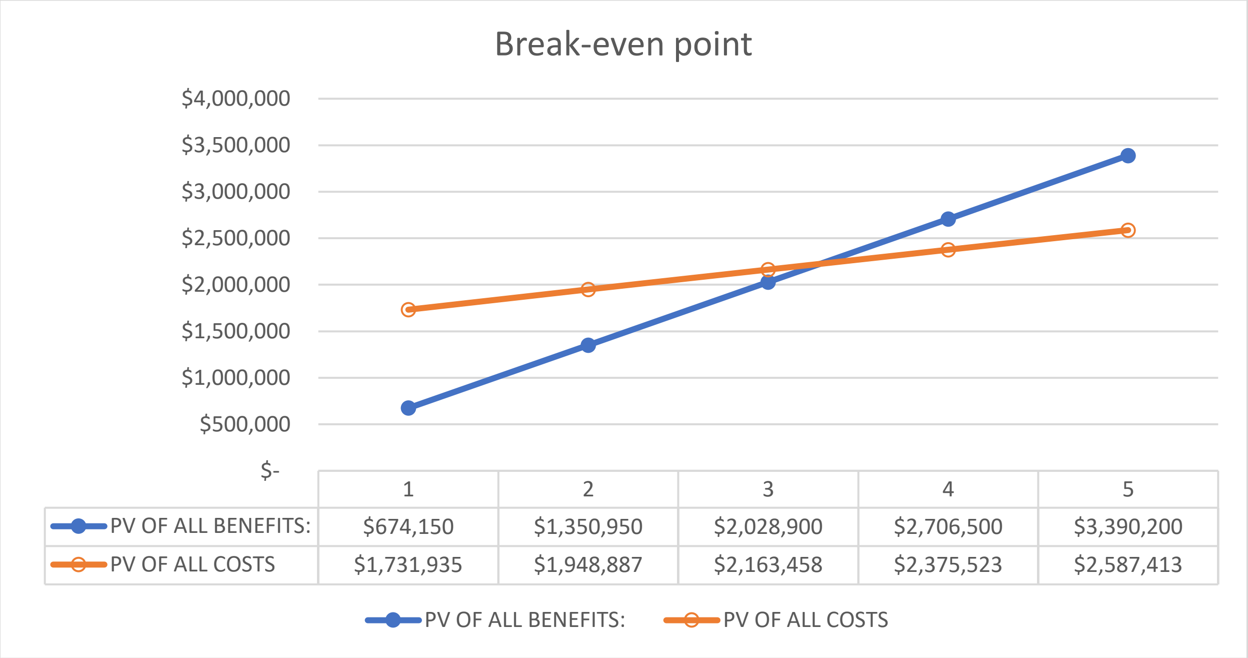
## Cost-Benefit Analysis:
## (modified since interim 1):



| | 2018 | 2019 | 2020 | 2021 | 2022 | Total |
|---|---|---|---|---|---|---|
| Increased Sales | $ 550,000 | $ 575,000 | $ 600,000 | $ 625,000 | $ 650,000 | |
| Reduction in customer complaint calls | $ 75,000 | $ 75,000 | $ 75,000 | $ 75,000 | $ 75,000 | |
| Reduced inventory costs | $ 70,000 | $ 70,000 | $ 70,000 | $ 70,000 | $ 70,000 | |
| **TOTAL BENEFITS:** | $ 695,000 | $ 720,000 | $ 745,000 | $ 770,000 | $ 795,000 | |
| **PV OF BENEFITS:** | **$ 674,150** | **$ 676,800** | **$ 677,950** | **$ 677,600** | **$ 683,700** | |
| **PV OF ALL BENEFITS:** | **$ 674,150** | **$ 1,350,950** | **$ 2,028,900** | **$ 2,706,500** | **$ 3,390,200** | |
| | | | | | | |
| Server | $ 120,000 | $ - | $ - | $ - | $ - | |
| Third party softwares | $ 36,000 | $ - | $ - | $ - | $ - | |
| Server Software | $ 11,500 | $ - | $ - | $ - | $ - | |
| Development labor | $ 1,300,000 | $ - | $ - | $ - | $ - | |
| Database development | $ 120,000 | | | | | |
| **TOTAL DEVELOPMENT COST** | $ 1,587,500 | $ - | $ - | $ - | $ - | |
| | | | | | | |
| Hardware maintenance | $ 56,000 | $ 84,000 | $ 84,000 | $ 84,000 | $ 84,000 | |
| Software and Troubleshooting | $ 22,000 | $ 22,000 | $ 22,000 | $ 22,000 | $ 22,000 | |
| Operational Labor | $ 120,000 | $ 124,800 | $ 129,792 | $ 134,984 | $ 140,383 | |
| **TOTAL OPERATIONAL COSTS** | $ 198,000 | $ 230,800 | $ 235,792 | $ 240,984 | $ 246,383 | |
| | | | | | | |
| **TOTAL COSTS** | $ 1,785,500 | $ 230,800 | $ 235,792 | $ 240,984 | $ 246,383 | |
| **PV OF COSTS** | **$ 1,731,935** | **$ 216,952** | **$ 214,571** | **$ 212,066** | **$ 211,889** | **$2,587,413** |
| **PV OF ALL COSTS** | **$ 1,731,935** | **$ 1,948,887** | **$ 2,163,458** | **$ 2,375,523** | **$ 2,587,413** | |
| | | | | | | |
| **TOTAL PROJECT BENEFITS COSTS** | **$ (1,090,500)** | **$ 489,200** | **$ 509,208** | **$ 529,016** | **$ 548,617** | |
| **YEARLY NPV** | **$ (1,057,785)** | **$ 459,848** | **$ 463,379** | **$ 465,534** | **$ 471,811** | **$ 802,787** |
| **CUMULATIVE NPV** | **$ (1,057,785)** | **$ (597,937)** | **$ (134,558)** | **$ 330,977** | **$ 802,787** | |
| | | | | | | |
| **RETURN ON INVESTMENT (ROI)** | **31.03%** | | | | | |
| | | | | | | |
| **BREAK-EVEN POINT (years)** | **3.29** | **0.29** | Since cashflow is positive in year 4, we add 0.29 to 3 yrs | | | |
| | | | | | | |
| **INTANGIBLE BENEFITS** | Product by competitors already in the market | | | | | |

**Break-Even Point:**

**(modified since interim 1):**



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| PV OF ALL BENEFITS: | $674,150 | $1,350,950 | $2,028,900 | $2,706,500 | $3,390,200 |
| PV OF ALL COSTS | $1,731,935 | $1,948,887 | $2,163,458 | $2,375,523 | $2,587,413 |

## Organizational Feasibility:

The main goal of the project is to reduce the unwanted time delay and establish a healthy communication between all the stakeholders of the system. It also helps in reducing the paper work as we go digital. Going digital open doors to detailed analysis which in turn leads to insights for improving the business of the organization. Since this simplifies many processes and makes life easier, the users will accept the system.

This also shows that there is strategic alignment. That is, the project goal is very well aligned with the business objectives of the organization. Hence from the organizational feasibility perspective, the project is less risky.

**Stakeholder Analysis:**

| **Stakeholders** | **Role** |
|---|---|
| Project Champion | The Chair of the School |
| Organizational Management | The Principal and other senior staff members of the school |
| System Users | Faculties and students of the school |

5. Requirements Definition:

## Functional Requirements:

1. **Manage Class Enrollment:**

   i. **Student enrolls in a class**
   Students can choose the course wanted from the list available and hit the enroll button under the enrollment tab.

   ii. **Student swaps a class**
   Students can swap between two courses using the swap option available under the enrollment tab.

   iii. **Student drops a class**
   Student can drop courses using the drop option available under the enrollment tab.

2. **Manage Grades:**

   i. **Professor enters grade**
   The faculty can enter grades manually under the grades tab.

   ii. **Professor changes grade**
   The faculty can change grades manually under the grades tab.

   iii. **Professor posts final grade at end of term**
   The professor enters the final grades manually under the grades tab.

3. **Student Feedback System**

   i. **Faculty provides the link for getting the feedback**
   The faculty enables the link in the feedback tab of the system so that it is available for all the students at the same time at the end of the semester.

   ii. **Students submit the feedback**
   Using the link, students answer various types of questions about the course and the faculty and hit the submit button.

## Non-Functional Requirements

1. **Operational Requirements**
    - The system should operate in all web browsers and mobile devices.
    - The system should log off automatically if the idle time is more than 15 minutes.

2. **Performance Requirements**
    - The system should autosave immediately, whenever data is entered to prevent data loss.
    - Faculty should be able to monitor student activities in the system.

3. **Security Requirements**
    - The system users should be protected with password access.
    - The data present in the system should be kept confidential.

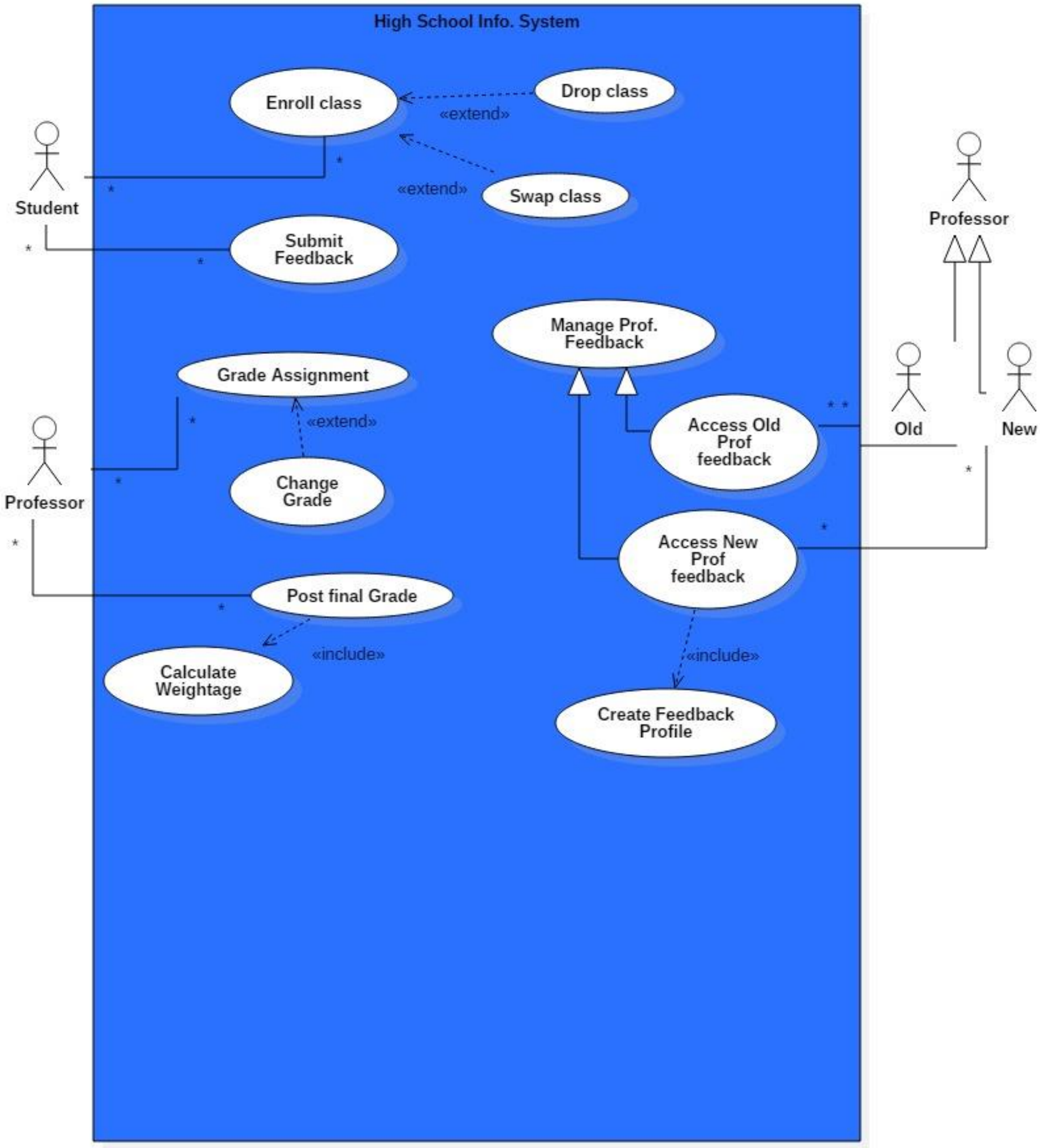
4. **Cultural and Political Requirements**

    No special cultural and political requirements are anticipated.

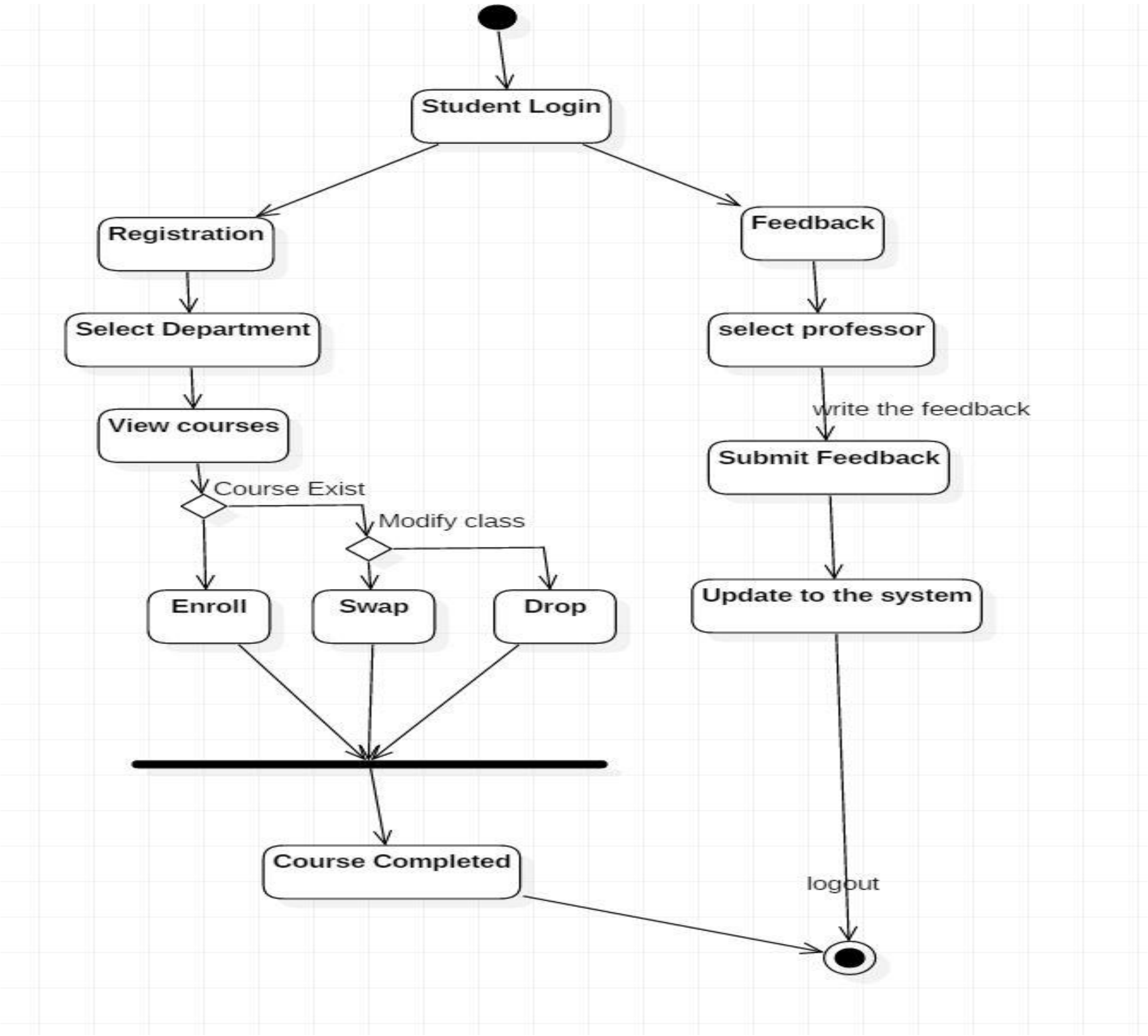
6. Functional Model:

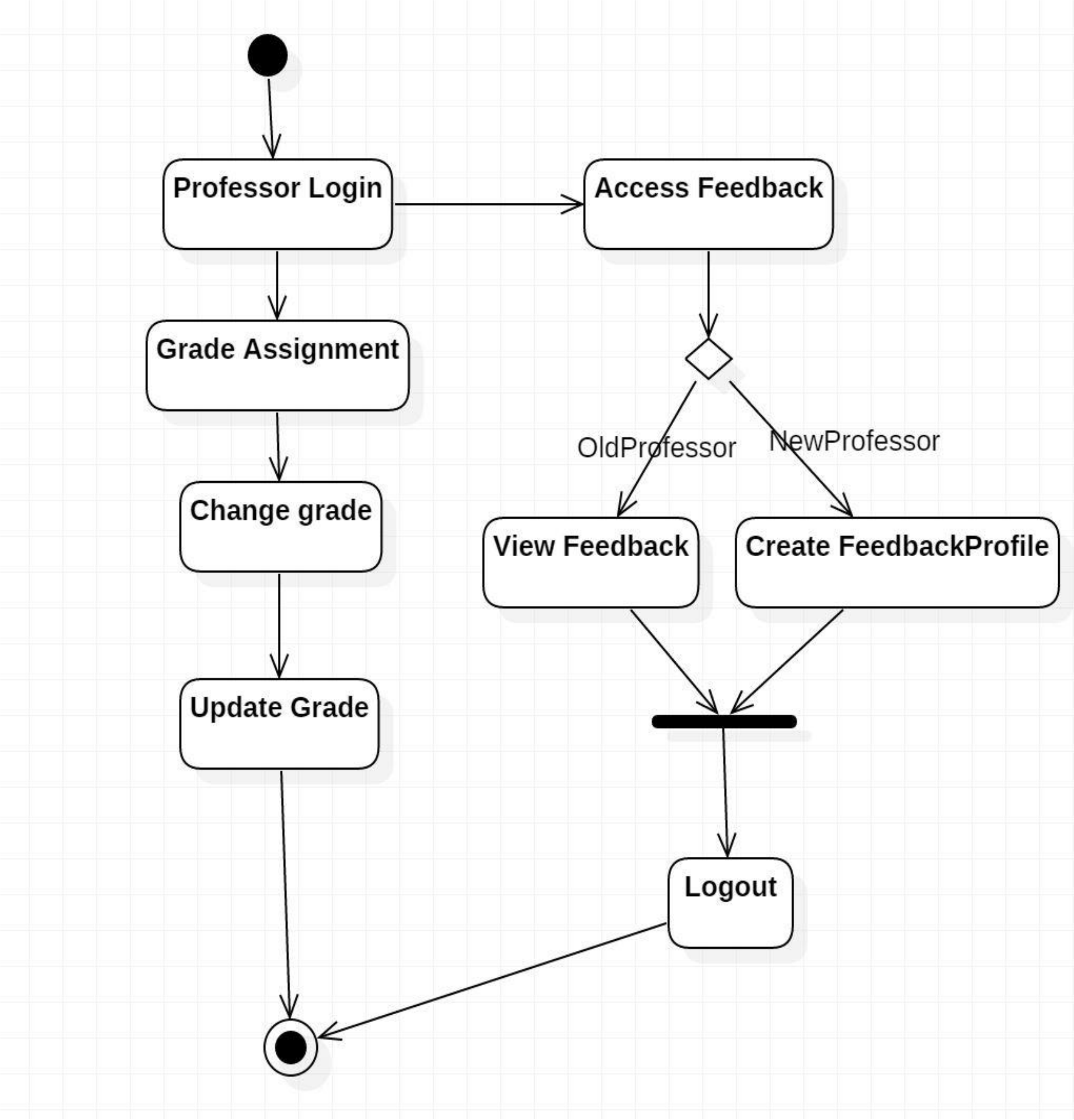
Use-Case diagram:

(modified based on feedback of interim 1):



## Activity Diagrams




Professor Login
Access Feedback
Grade Assignment
Change grade
Update Grade
OldProfessor
NewProfessor
View Feedback
Create FeedbackProfile
Logout

## Use-Case Descriptions

| | | |
|---|---|---|
| **Use Case Name:** Enroll Class | **ID:** 1 | **Importance Level:** High |
| **Primary Actor:** Student | **Use Case Type:** Overview, Essential | |
| **Stakeholders and interests:**<br>Student – wants to enroll, swap or drop a class | | |
| **Brief Description:**<br>This use case describes how a student can enroll in a class, swap with another class and drop a currently enrolled class. | | |
| **Trigger:**<br>Student clicks on enroll button after selecting desired course<br><br>**Type:** External | | |
| **Relationships:**<br>**Association:** Student<br>**Include:**<br>**Extend:** Swap class, Drop class<br>**Generalization:** | | |
| **Normal Flow of Events:**<br><br>1. The student visits the school's website<br>2. The system prompts to enter credentials<br>3. The student provides his Student ID and credentials<br>4. The system opens portal<br>5. The student clicks on enroll tab<br>6. The system displays list of classes with class ID and section numbers<br>7. The student selects the course he wants to enroll in.<br>8. The system prompts confirmation message<br>9. The student confirms enrollment<br>10. The system displays success/failure message<br>11. If the student wants to drop a class:<br>12. Drop class sub-flow is performed<br>13. If the student wants to Swap a class: | | |

14. Swap class sub-flow is performed
15. System displays details of class enrolled by the student. Class ID, Schedule

**Sub Flows:**

S-1: Drop Class

1. The student visits the school's website
2. The system prompts to enter credentials
3. The student provides his Student ID and credentials
4. The system opens portal
5. Student clicks on Drop class button
6. System asks class name and class ID
7. Student enters class ID and class name
8. System prompts confirmation message to the student
9. Students confirms drop
10. The system drops the student from that class and show success/ failure

S-2: Swap Class:

1. The student visits the school's website
2. The system prompts to enter credentials
3. The student provides his Student ID and credentials
4. The system opens portal
5. Student clicks on Swap class
6. System prompts to enter class ID of currently enrolled class
7. Student enters class ID of enrolled class
8. System prompts to enter class ID of the class they want to swap with
9. Student enters class ID 2
10. System prompts confirmation message to the student
11. Student consents

| |
|---|
| 12. System performs Drop Class sub-flow on class ID 1<br>13. System performs Enroll Class on class ID 2 |
| **Alternate/Exceptional Flows:** None |

| | | |
|---|---|---|
| **Use Case Name:** Submit feedback | **ID**: 2 | **Importance Level:** High |
| **Primary Actor:** Student | **Use Case Type:** Overview, Essential | |
| **Stakeholders and interests:**<br>Student – wants to submit feedback<br>Professor – wants to see performance | | |
| **Brief Description:**<br>This use case describes how a student can submit feedback | | |
| **Trigger:**<br>Student clicks on submit feedback<br><br>**Type:** External | | |
| **Relationships:**<br>**Association:** Student<br>**Include:**<br>**Extend:** None<br>**Generalization:** | | |
| **Normal Flow of Events:**<br><br>1. The student visits the school's website<br>2. The system prompts to enter credentials<br>3. The student provides his Student ID and credentials<br>4. The system opens portal<br>5. The student clicks on feedback tab<br>6. The system displays list of courses with class ID and section numbers<br>7. The student selects the course he wants to submit feedback for<br>8. The system displays feedback questions<br>9. The student types feedback and hits submit | | |

| |
|---|
| 10. The system displays success/failure message |
| **Sub Flows:** None |
| **Alternate/Exceptional Flows:** None |

| | | |
|---|---|---|
| **Use Case Name:** Grade assignment | **ID:** 3 | **Importance Level:** High |
| **Primary Actor:** Professor | **Use Case Type:** Overview, Essential | |
| **Stakeholders and interests:**<br>Student – wants to track progress<br>Professor – wants to help students learn from mistakes | | |
| **Brief Description:**<br>This use case describes how a professor can grade assignments online | | |
| **Trigger:**<br>Professor clicks on grade assignment<br><br>**Type:** External | | |
| **Relationships:**<br>**Association:** Professor<br>**Include:**<br>**Extend:** Change grade<br>**Generalization:** | | |

**Normal Flow of Events:**

1. The Professor visits the school's website
2. The system prompts to enter credentials
3. The Professor provides his ID and credentials
4. The system opens portal
5. The professor clicks on courses tab
6. The system displays list of courses with class ID and section numbers
7. The professor selects the course he wants to grade assignment for
8. The system displays list of students
9. Professor clicks on student he wants to grade for
10. The system displays submitted assignments
11. The professor clicks on assignment for viewing
12. The system displays assignment
13. The professor types grade for assignment and clicks submit
14. The system displays confirmation message
15. Professor consents
16. The system displays success/failure message

**Sub Flows:**

S-1: Change grade:

1. The Professor visits the school's website
2. The system prompts to enter credentials
3. The Professor provides his ID and credentials
4. The system opens portal
5. The professor clicks on courses tab
6. The system displays list of courses with class ID and section numbers
7. The professor selects the course he wants to change assignment grade for
8. The system displays list of students
9. Professor clicks on student
10. The system displays graded assignments
11. The professor clicks on assignment
12. The system displays assignment grade

| |
|---|
| 13. The professor types new grade for assignment and clicks submit<br>14. The system displays confirmation message<br>15. Professor consents<br>16. The system displays success/failure message |
| **Alternate/Exceptional Flows:** None |

| | | |
|---|---|---|
| **Use Case Name:** Post final grade | **ID:** 4 | **Importance Level:** High |
| **Primary Actor:** Professor | **Use Case Type:** Overview, Essential | |
| **Stakeholders and interests:**<br>Student – wants to see performance<br>Professor – wants to complete duty of posting final scores | | |
| **Brief Description:**<br>This use case describes how a professor can post final grade | | |
| **Trigger:**<br>Professor clicks on post final grade<br><br>**Type:** External | | |
| **Relationships:**<br>**Association:** Professor<br>**Include:** Calculate weightage<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Events:**<br><br>1. The Professor visits the school's website<br>2. The system prompts to enter credentials<br>3. The Professor provides his ID and credentials<br>4. The system opens portal<br>5. The professor clicks on courses tab | | |

6. The system displays list of courses with class ID and section numbers
7. The professor selects the course he wants to post final grade for
8. System displays list of students
9. Professor clicks on student
10. The system calculates grade and displays confirmation
11. The professor consents to the grade
12. The system displays confirmation message

**Sub Flows:**

**S-1:** Calculate weightage:

1. The Professor visits the school’s website
2. The system prompts to enter credentials
3. The Professor provides his ID and credentials
4. The system opens portal
5. The professor clicks on courses tab
6. The system displays list of courses with class ID and section numbers
7. The professor selects the course he wants to post final grade for
8. System displays list of students
9. Professor clicks on student
10. The system prompts to enter weightage of assignments and exams
11. The professor enters weightage
12. The system calculates grade and displays confirmation
13. The professor consents to the grade
14. The system displays confirmation message

**Alternate/Exceptional Flows:** None

|  |  |  |
|---|---|---|
| **Use Case Name:** Access Old Prof. Feedback | **ID:** 6 | **Importance Level:** High |
| **Primary Actor:** Professor | **Use Case Type:** Overview, Essential |  |
| **Stakeholders and interests:**<br>Professor – wants to review his performance |  |  |
| **Brief Description:**<br>This use case describes how an old professor can access feedback |  |  |
| **Trigger:**<br>Professor clicks on feedback tab<br><br>**Type:** External |  |  |
| **Relationships:**<br>**Association:** Professor<br>**Include:**<br>**Extend:**<br>**Generalization:** Manage Feedback |  |  |
| **Normal Flow of Events:**<br>1. The Professor visits the school's website<br>2. The system prompts to enter credentials<br>3. The Professor provides his ID and credentials<br>4. The system opens portal<br>5. The professor clicks on feedback tab<br>6. The system displays list of courses with class ID and section numbers<br>7. The professor selects the course he wants to view feedback for<br>8. System displays feedback analysis |  |  |
| **Sub Flows:** None |  |  |
| **Alternate/Exceptional Flows:** None |  |  |

| | | |
|---|---|---|
| **Use Case Name:** Access New Prof. Feedback | **ID**: 6 | **Importance Level:** High |
| **Primary Actor:** Professor | **Use Case Type:** Overview, Essential | |

**Stakeholders and interests:**

Professor – wants to review his performance

**Brief Description:**

This use case describes how a new professor can access feedback

**Trigger:**

Professor clicks on feedback tab

**Type:** External

**Relationships:**

**Association:** Professor

**Include:** Create Feedback Profile

**Extend:**

**Generalization:** Manage Feedback

**Normal Flow of Events:**

1. The Professor visits the school's website
2. The system prompts to enter credentials
3. The Professor provides his ID and credentials
4. The system opens portal
5. The professor clicks on feedback tab
6. The system displays list of courses with class ID and section numbers
7. The professor selects the course he wants to view feedback for
8. The system prompts to create a feedback profile
9. Professor creates feedback profile
10. System displays student feedback

**Sub Flows:** None

**Alternate/Exceptional Flows:** None

7. Structural Model

CRC Cards

Front

| Class Name: Student | Id: 1 | Type: Concrete, Domain |
|---|---|---|
| Description: An individual who registers for a class and submits feedbacks using the system. | | Associated Use Cases: 2 |
| Responsibilties | | Collaborators |
| Enroll class | | System |
| Drop class | | System |
| Swap class | | System |
| Submit feedback | | System |

Back

| Attributes |
|---|
| Student Name (Text) |
| Student Id (Integer) |
| |
| |
| Relationships |
| Generalization (a-kind-of): |
| Aggregation (has-parts): |
| Other Associations: Enroll Class, Submit feedback, System |

Front

| | | |
|---|---|---|
| Class Name: Login Details | Id: 2 | Type: Concrete, Domain |
| Description: A class which specifies the login details of the customer and the professor. | | Associated Use Cases: 0 |
| Responsibilties | | Collaborators |
| | | |
| | | |
| | | |
| | | |

Back

| |
|---|
| Attributes |
| Username (Alphanumeric) |
| Passwork (Alphanumeric) |
| |
| |
| Relationships |
| Generalization (a-kind-of): |
| Aggregation (has-parts): |

Front

| | | |
|---|---|---|
| Class Name: Enroll Class | Id: 3 | Type: Concrete, Domain |
| Description: A class which specifies details about the classes enrolled. | | Associated Use Cases: 2 |
| Responsibilties | | Collaborators |
| Enroll in class | | Student |
| | | |
| | | |
| | | |

| |
|---|
| Other Associations: Student, Professor |

Back

| Attributes |
|---|
| Class Name (Text) |

| |
|---|
| Class Code (Integer) |
| |
| |
| Relationships |
| Generalization (a-kind-of): |
| Aggregation (has-parts): |
| Other Associations: Student, Swap class, Drop class |

Front

| | | |
|---|---|---|
| **Class Name:** Swap Class | **Id:** 4 | **Type:** Concrete, Domain |
| **Description:** A class which specifies details about the classes swapped. | | **Associated Use Cases:** 1 |
| **Responsibilties** | | **Collaborators** |
| Swap a class | | Student |
| | | |
| | | |
| | | |

Back

| |
|---|
| **Attributes** |
| Class Name (Text) |
| Class Code (Integer) |
| |
| |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |

|  |
|---|
| **Other Associations:** Student, Enroll Class |

Front

|  |  |  |
|---|---|---|
| **Class Name:** Drop Class | **Id:** 5 | **Type:** Concrete, Domain |
| **Description:** A class which specifies details about the classes dropped. |  | **Associated Use Cases:** 1 |
| **Responsibilties** |  | **Collaborators** |
| Drop a class |  | Student |
|  |  |  |
|  |  |  |
|  |  |  |

Back

|  |
|---|
| **Attributes** |
| Class Name (Text) |
| Class Code (Integer) |
|  |
|  |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |

|  |
|---|
| **Other Associations:** Student, Enroll Class |

Front

|  |  |  |
|---|---|---|
| **Class Name:** Professor | **Id:** 6 | **Type:** Concrete, Domain |
| **Description:** An individual who posts grade for the assignments and exams. | | **Associated Use Cases:** 4 |
| **Responsibilties** | | **Collaborators** |
| Change Grade | | System |
| Enter Grade | | System |
|  | |  |
|  | |  |

Back

| **Attributes** |
|---|
| Professor Name (Text) |
| Professor Id (Integer) |

| |
|---|
| |
| |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |
| **Other Associations:** Change grade, Old professor, New Professor, Post final grade, System |

**Front**

| | | |
|---|---|---|
| **Class Name:** Old Professor | **Id:** 7 | **Type:** Concrete, Domain |
| **Description:** An individual who need not create feedback profile. | | **Associated Use Cases:** 1 |
| **Responsibilties** | | **Collaborators** |
| | | |
| | | |
| | | |
| | | |

**Back**

| |
|---|
| **Attributes** |
| Professor Name (Text) |
| Professor Id (Integer) |
| |
| |
| **Relationships** |
| **Generalization (a-kind-of):** Professor |
| **Aggregation (has-parts):** |
| **Other Associations:** New professor, System |

Front

| | | |
|---|---|---|
| **Class Name:** New Professor | **Id:** 8 | **Type:** Concrete, Domain |
| **Description:** An individual who needs to create a feedback profile. | | **Associated Use Cases:** 1 |
| **Responsibilties** | | **Collaborators** |
| | | |
| | | |
| | | |

| | |
|---|---|
| | |

Back

| |
|---|
| **Attributes** |
| Professor Name (Text) |
| Professor Id (Integer) |
| |
| |
| **Relationships** |
| **Generalization (a-kind-of):** Professor |
| **Aggregation (has-parts):** |
| **Other Associations:** Old professor**,** System |

Front

| | | |
|---|---|---|
| **Class Name:** System | **Id:** 9 | **Type:** Concrete, Domain |
| **Description:** A central application where class enrollment, posting grades and submitting feedback takes place. | | **Associated Use Cases:** 12 |
| **Responsibilties** | | **Collaborators** |
| Authenticate user | | Student, Professor |

| | |
|---|---|
| Grade management | Professor |
| Class enrollments | Student |
| Feedback Management | |

**Back**

| |
|---|
| **Attributes** |
| Username (Alphanumeric) |
| Password (Alphanumeric) |
| |
| |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |
| **Other Associations:** Student, Professor |

Front

| | | |
|---|---|---|
| **Class Name:** Feedback | **Id:** 10 | **Type:** Concrete, Domain |
| **Description:** A class which specifies details about the feedback submitted. | | **Associated Use Cases:** 2 |
| **Responsibilties** | | **Collaborators** |
| | | |
| | | |
| | | |
| | | |

Back

| |
|---|
| **Attributes** |
| Professor Id (Integer) |
| Professor Name (Text) |
| Class Name (Text) |
| Class Code (Integer) |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |
| **Other Associations:** Professor, System |

Front

| | | |
|---|---|---|
| **Class Name:** Assignment | **Id:** 11 | **Type:** Concrete, Domain |
| **Description:** A class which specifies details about the assignment submitted. | | **Associated Use Cases:** 4 |
| **Responsibilties** | | **Collaborators** |
| | | |
| | | |
| | | |
| | | |

Back

| **Attributes** |
|---|
| Student Id (Integer) |
| Student Name (Text) |
| Assignment Number (Integer) |

| |
|---|
| Assignment Grade (Integer) |
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |
| **Other Associations:** Professor, System |

Front

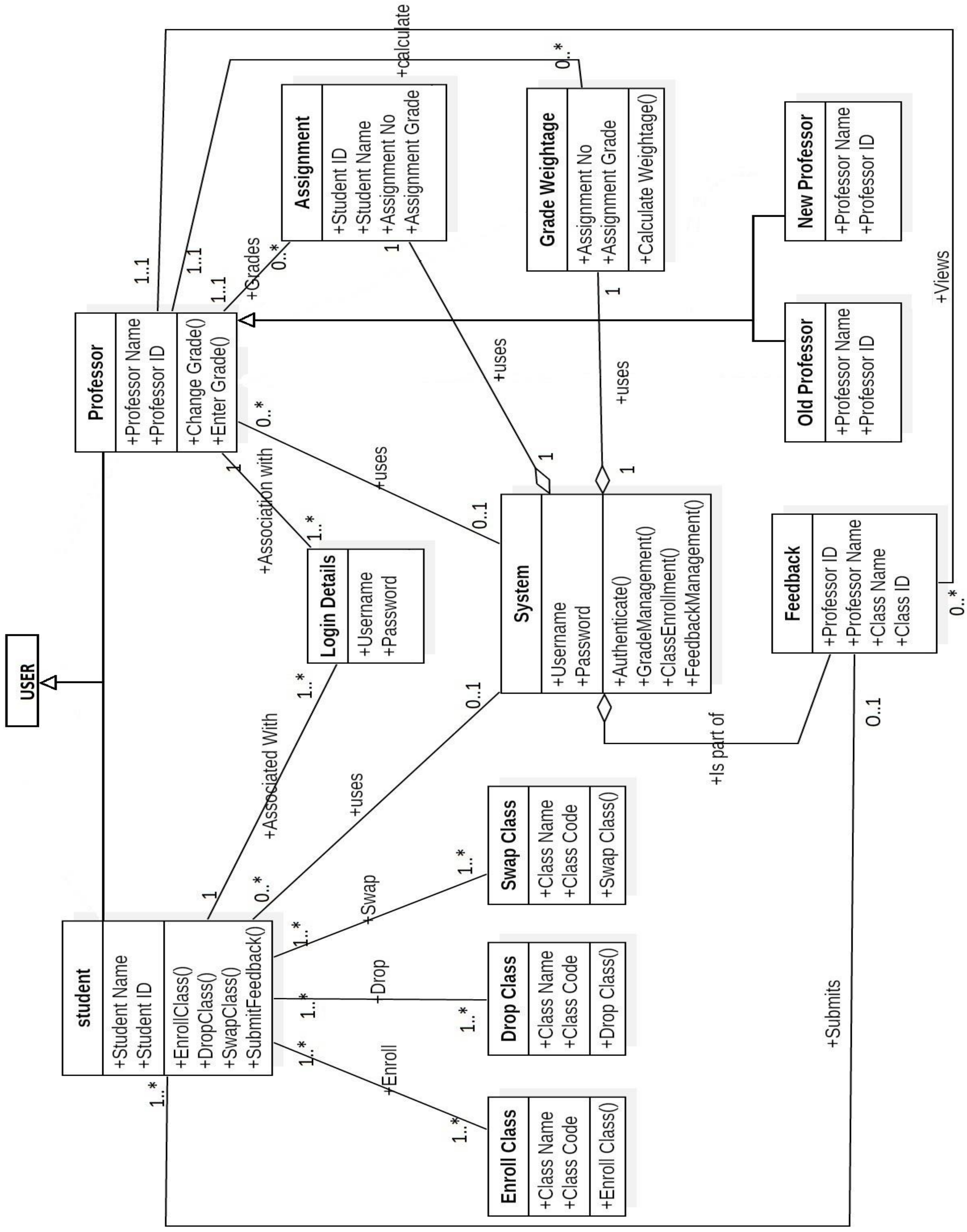
| | | |
|---|---|---|
| **Class Name:** Grade Weightage | **Id:** 12 | **Type:** Concrete, Domain |
| **Description:** A class which specifies details about the weightage distribution for each assignment. | | **Associated Use Cases:** 1 |
| **Responsibilties** | | **Collaborators** |
| Calculate weightage | | Professor, System |
| | | |
| | | |
| | | |

Back

| |
|---|
| **Attributes** |
| Assignment Number (Integer) |
| Assignment Grade (Integer) |
| |

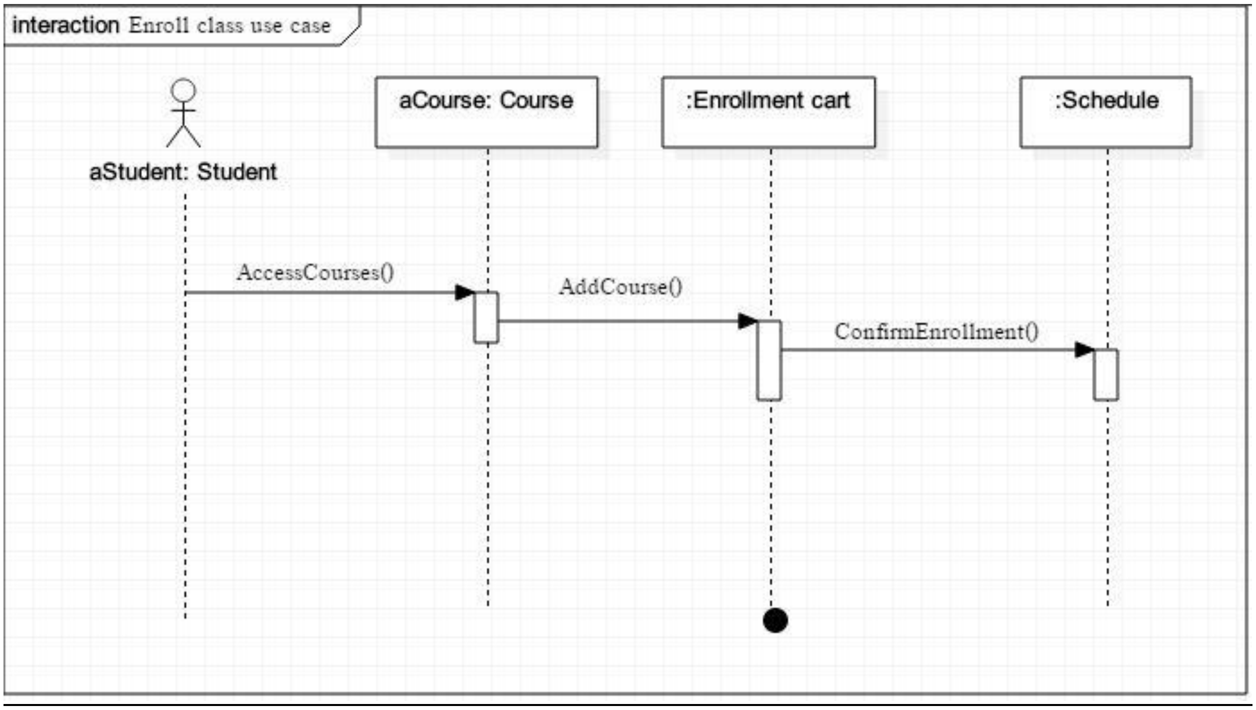
| |
|---|
| **Relationships** |
| **Generalization (a-kind-of):** |
| **Aggregation (has-parts):** |
| **Other Associations:** Professor, System |

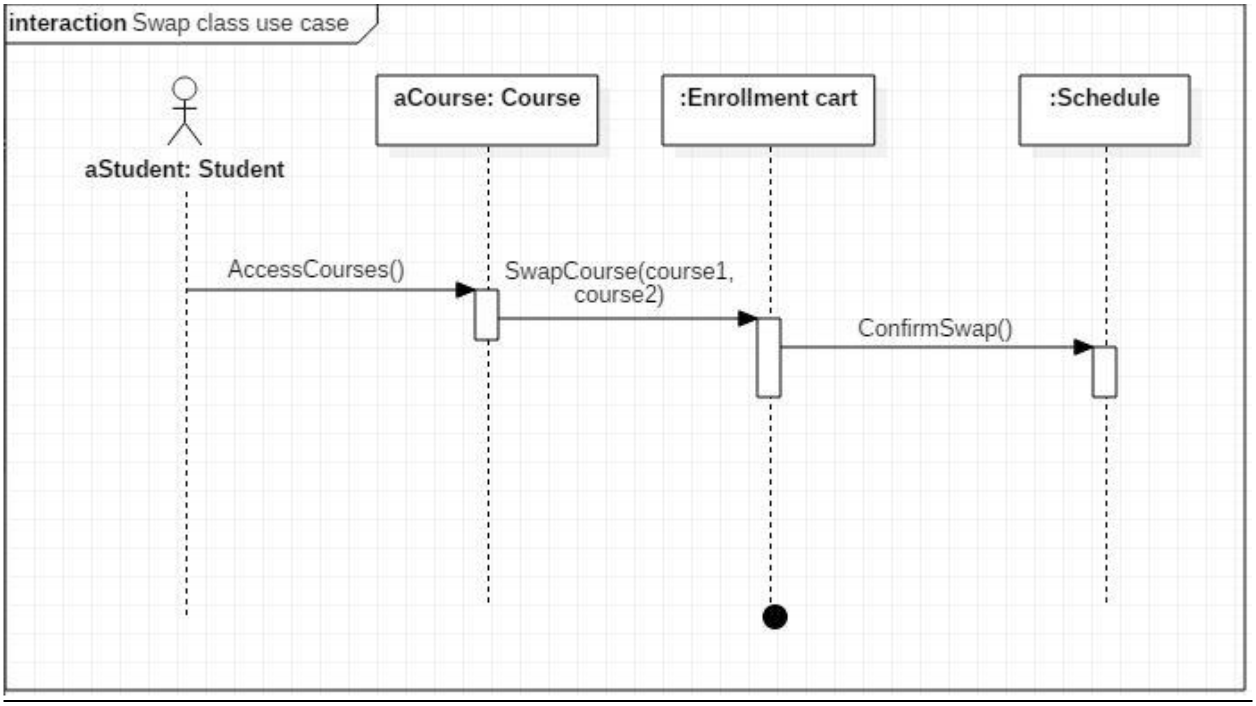
# Class Diagram

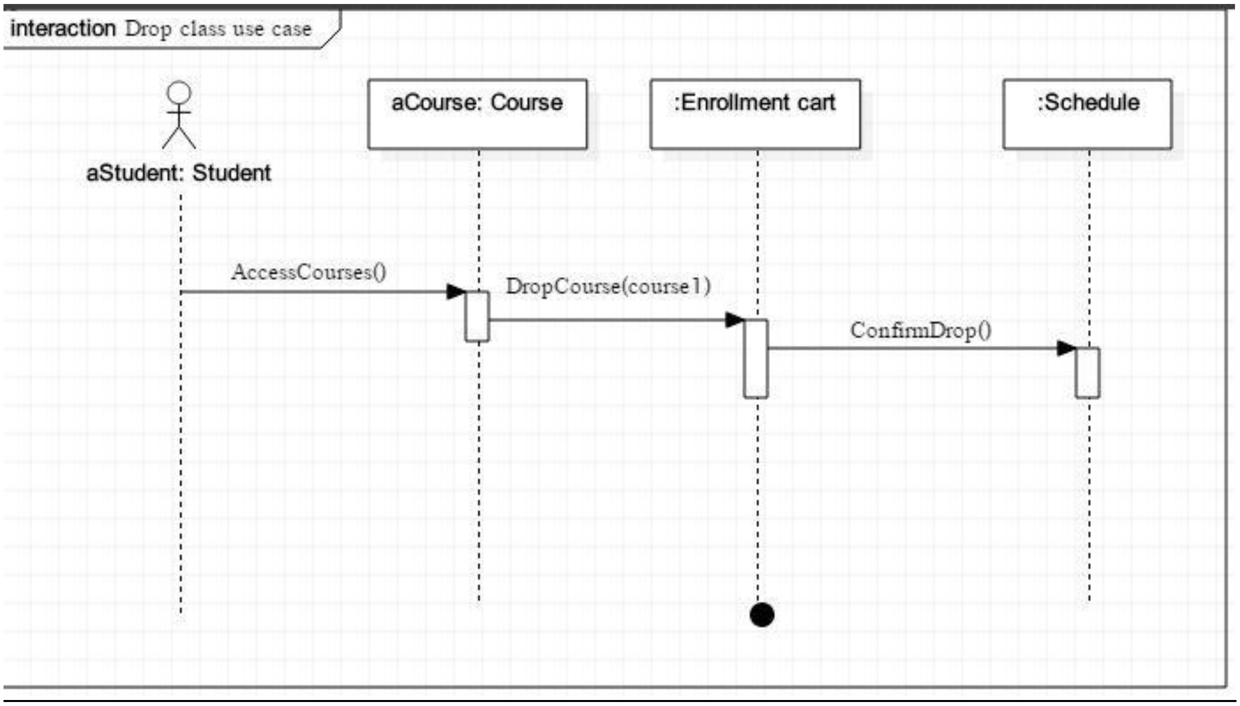


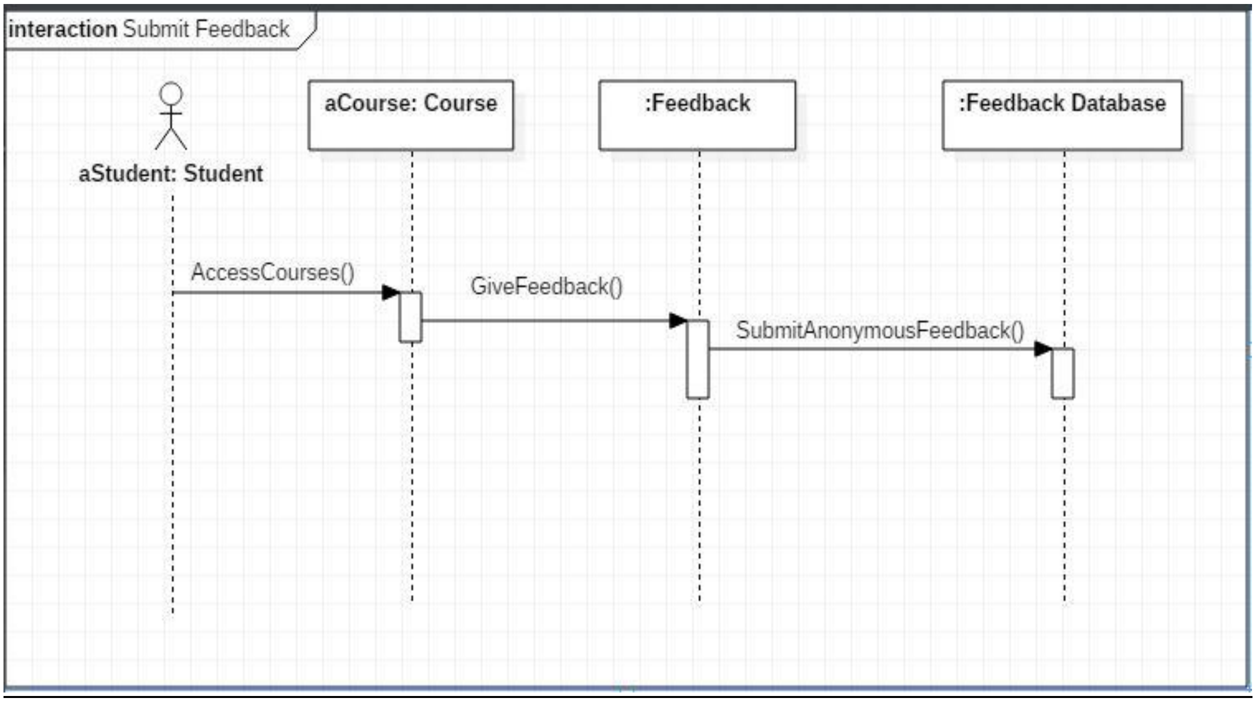
## 8. Behavioral model

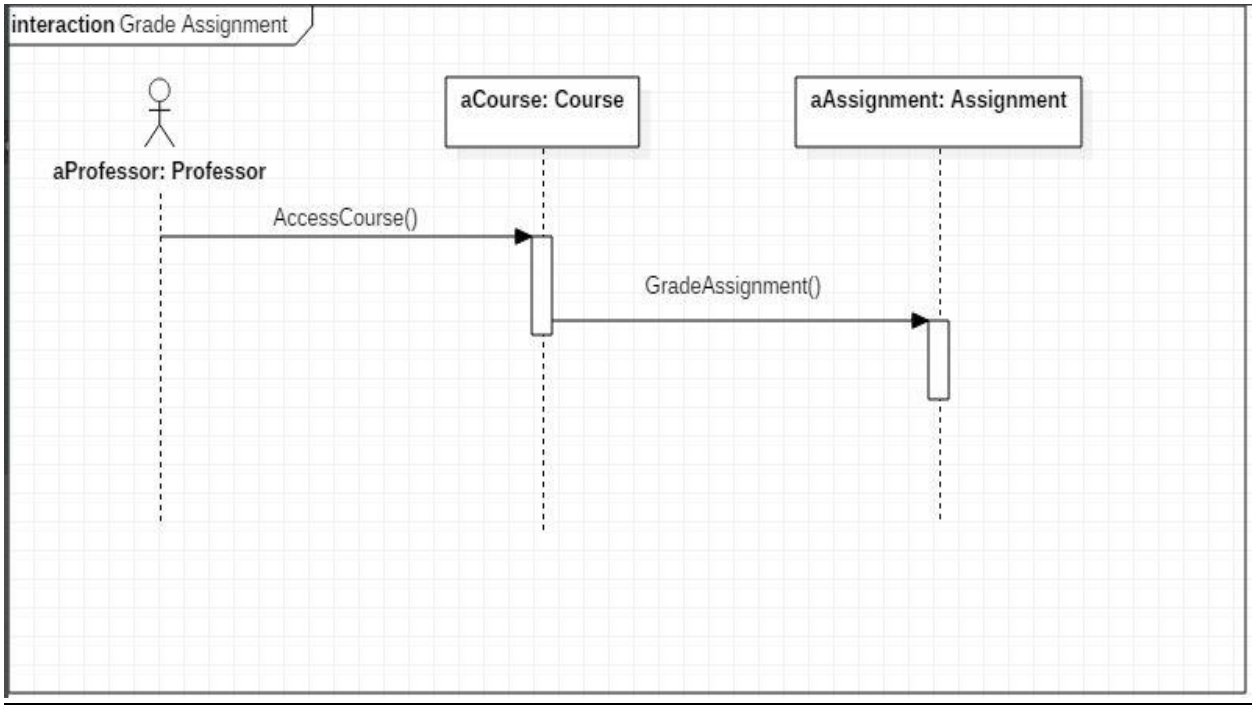
Sequence diagrams

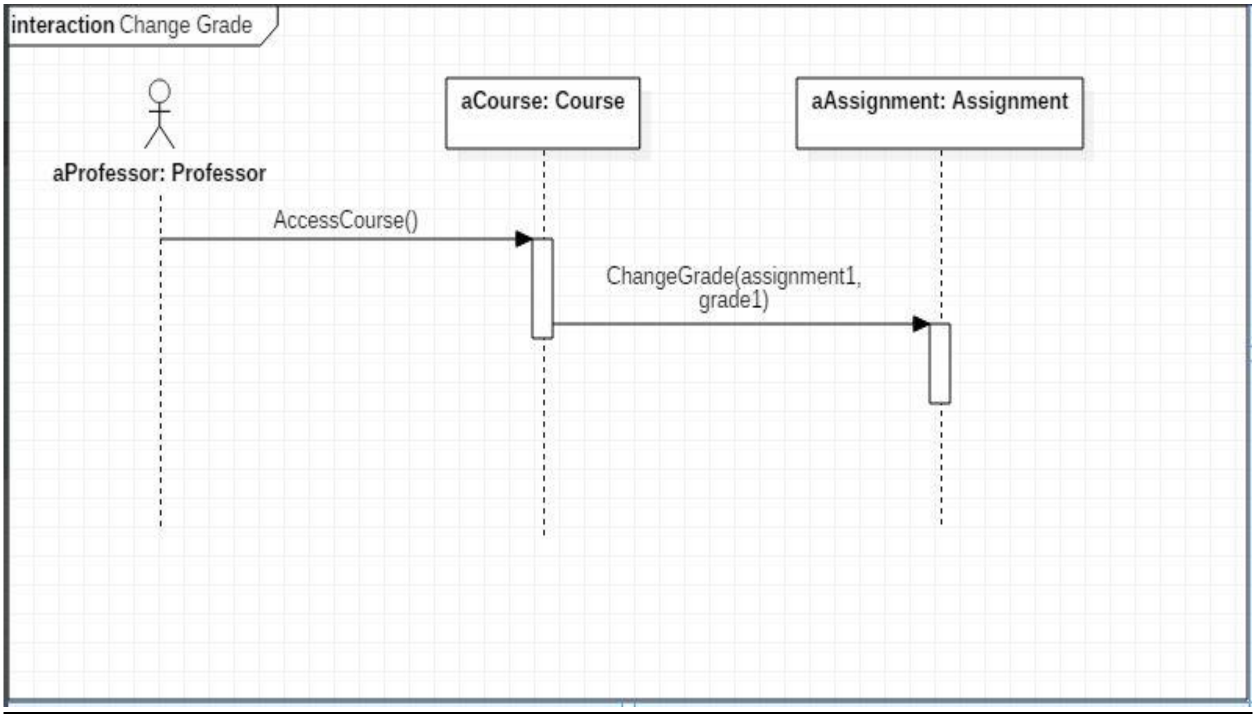


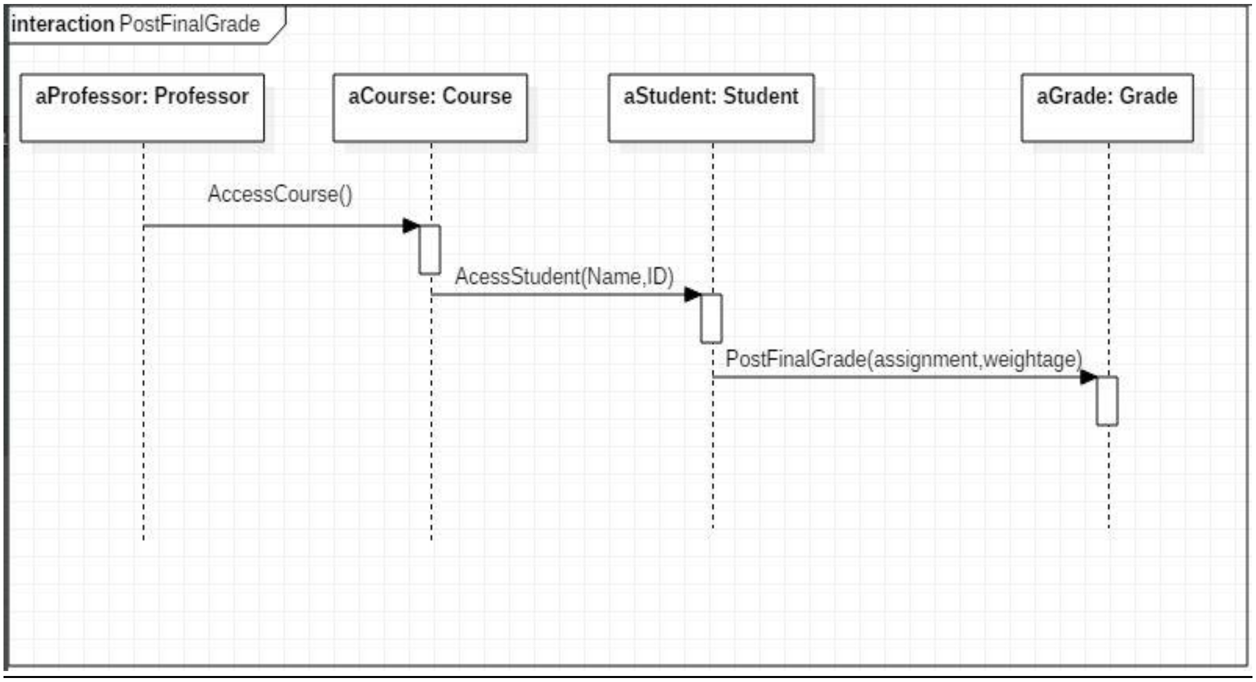


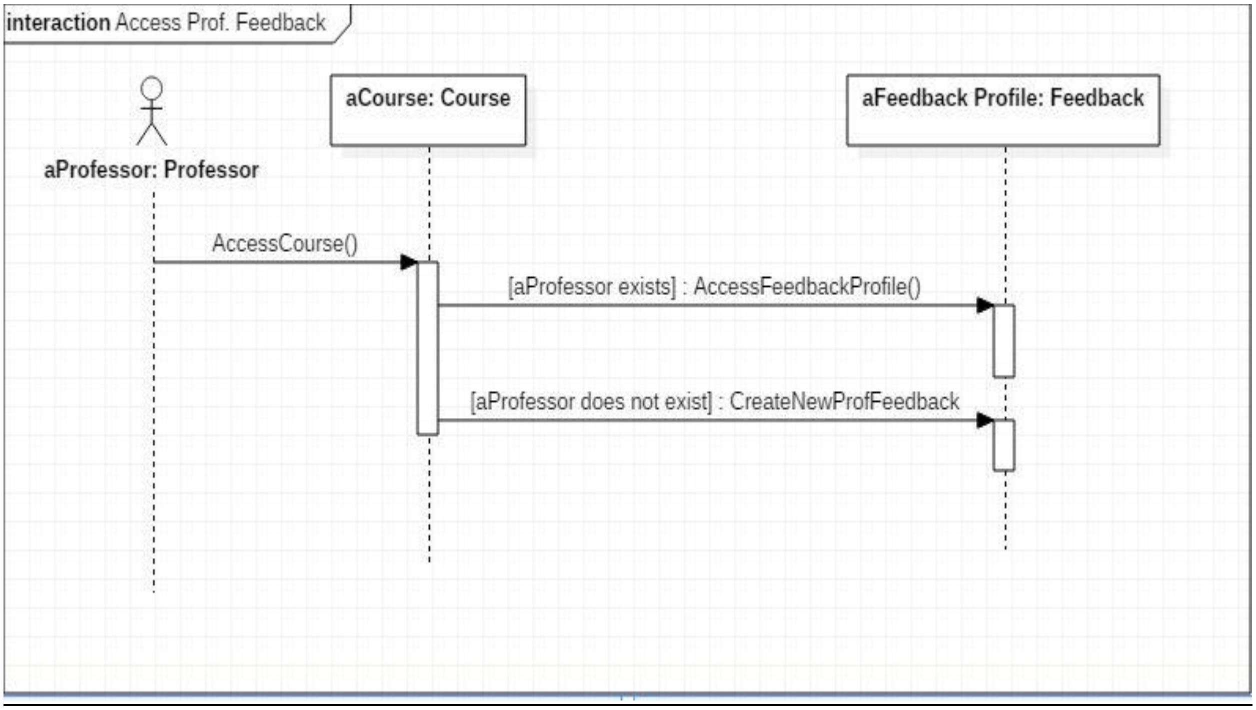

interaction Drop class use case
aStudent: Student
aCourse: Course
:Enrollment cart
:Schedule
AccessCourses()
DropCourse(course1)
ConfirmDrop()


interaction Submit Feedback
aStudent: Student
aCourse: Course
:Feedback
:Feedback Database
AccessCourses()
GiveFeedback()
SubmitAnonymousFeedback()

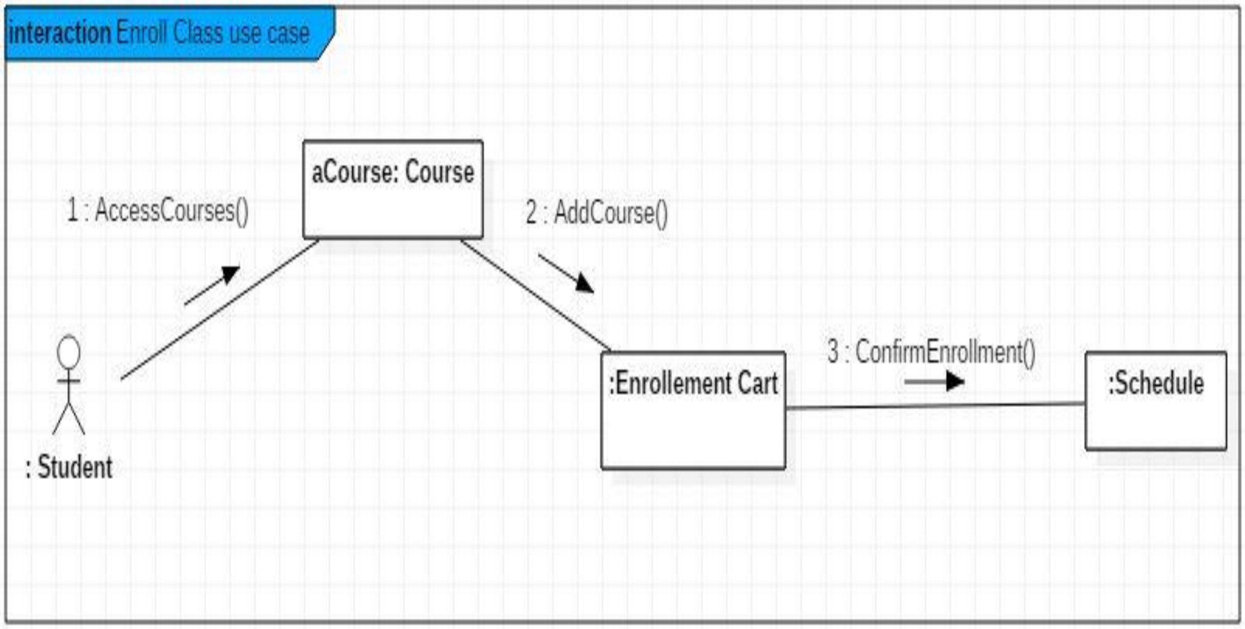

interaction Grade Assignment
aProfessor: Professor
aCourse: Course
aAssignment: Assignment
AccessCourse()
GradeAssignment()

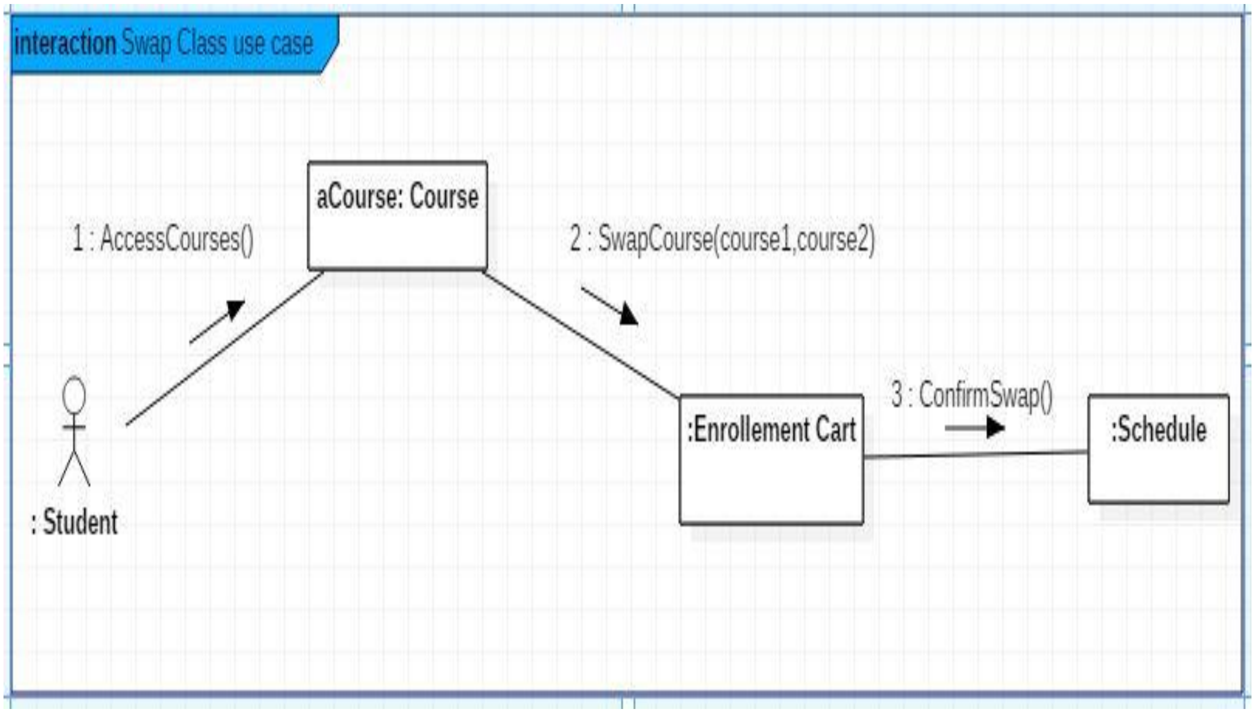

interaction Change Grade
aProfessor: Professor
aCourse: Course
aAssignment: Assignment
AccessCourse()
ChangeGrade(assignment1,
grade1)

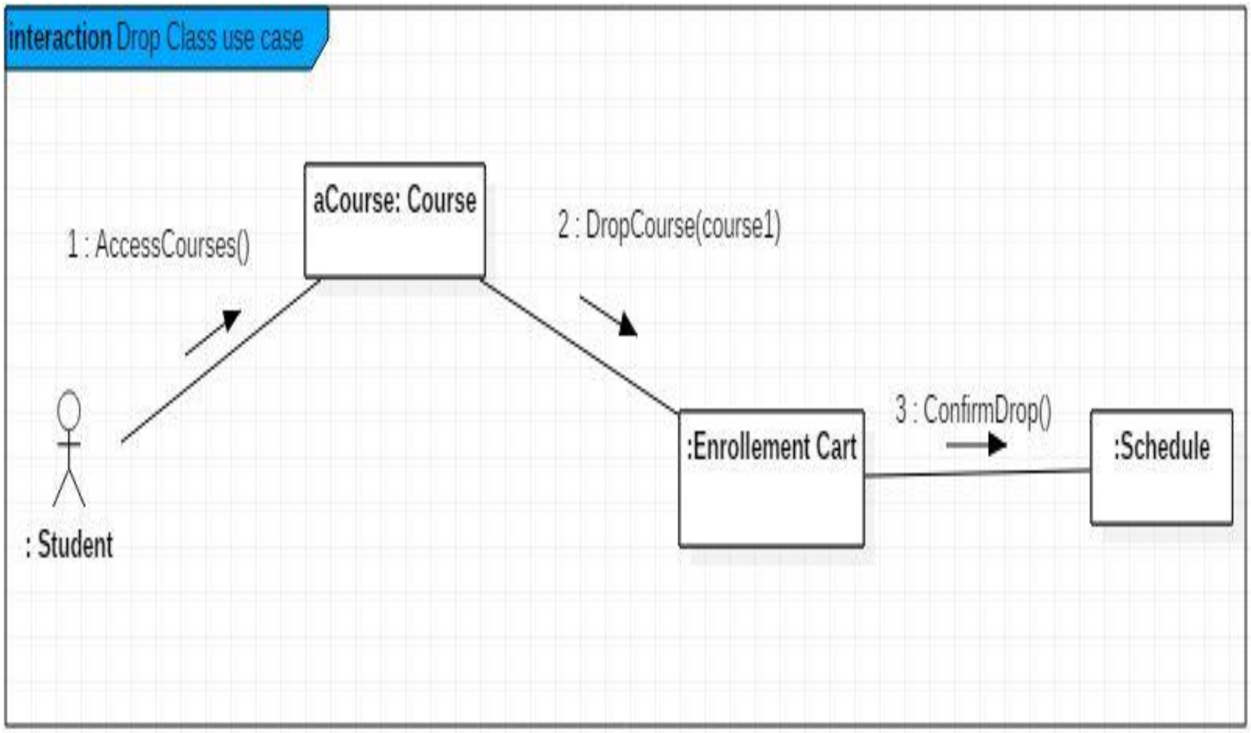

interaction PostFinalGrade
aProfessor: Professor
aCourse: Course
aStudent: Student
aGrade: Grade
AccessCourse()
AcessStudent(Name,ID)
PostFinalGrade(assignment,weightage)

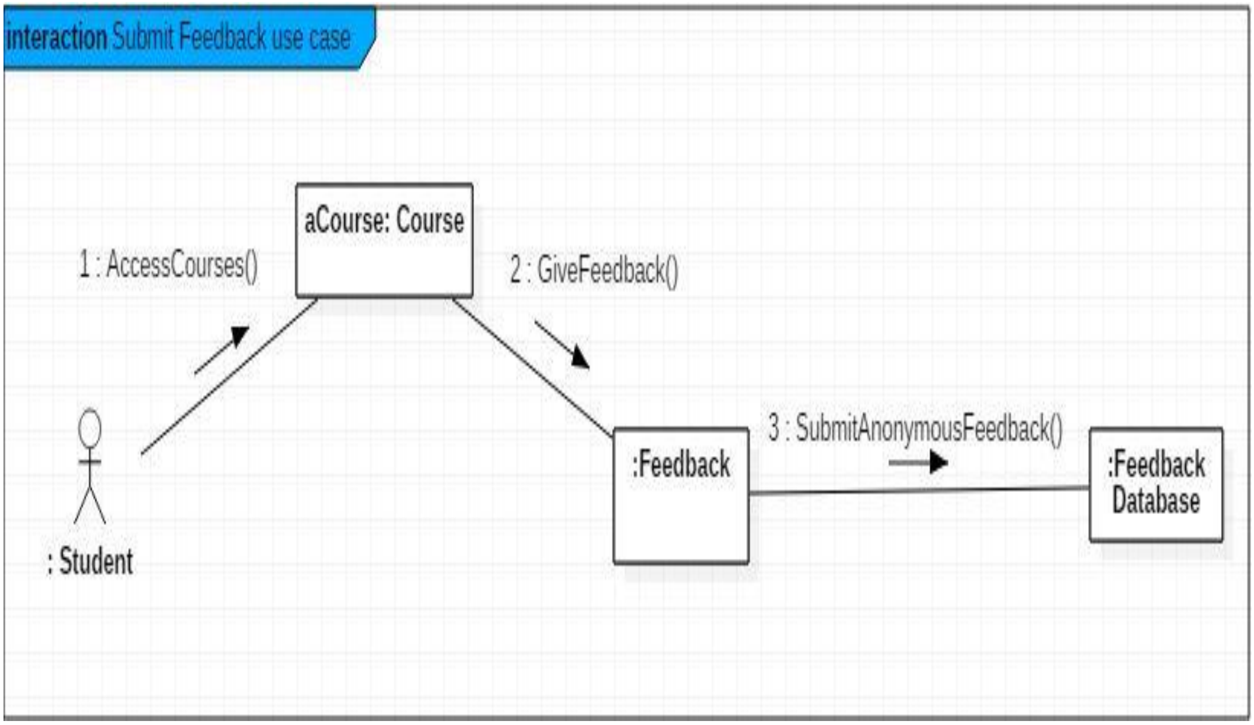

interaction Access Prof. Feedback
aProfessor: Professor
aCourse: Course
aFeedback Profile: Feedback
AccessCourse()
[aProfessor exists] : AccessFeedbackProfile()
[aProfessor does not exist] : CreateNewProfFeedback

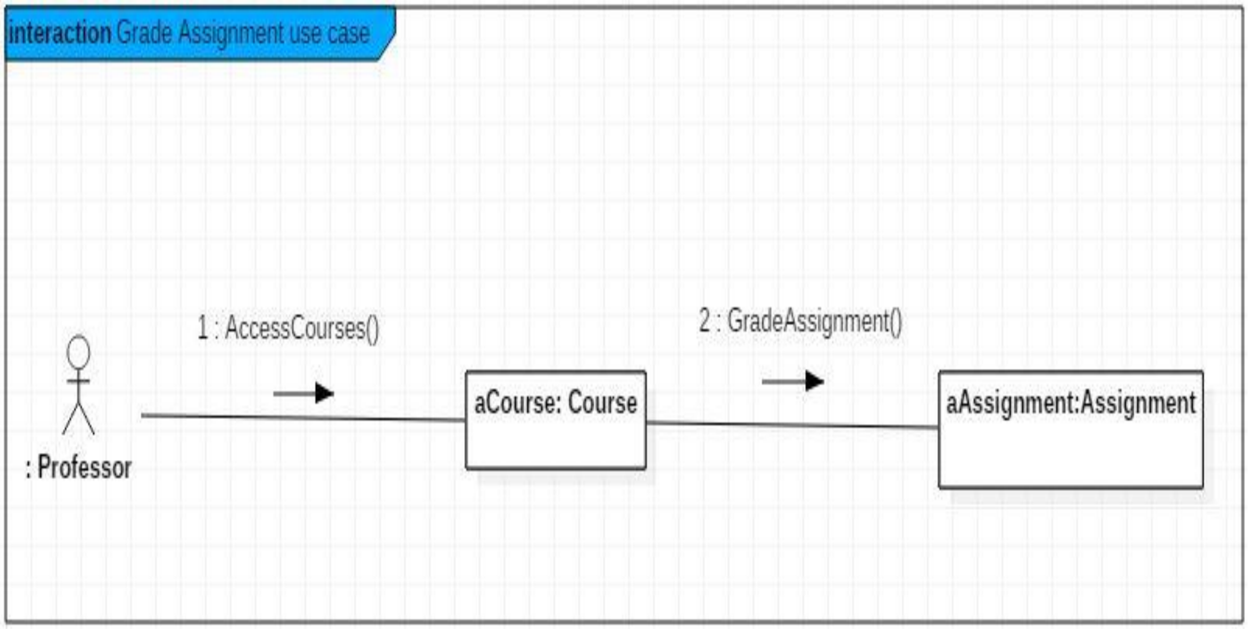
Communication Diagrams

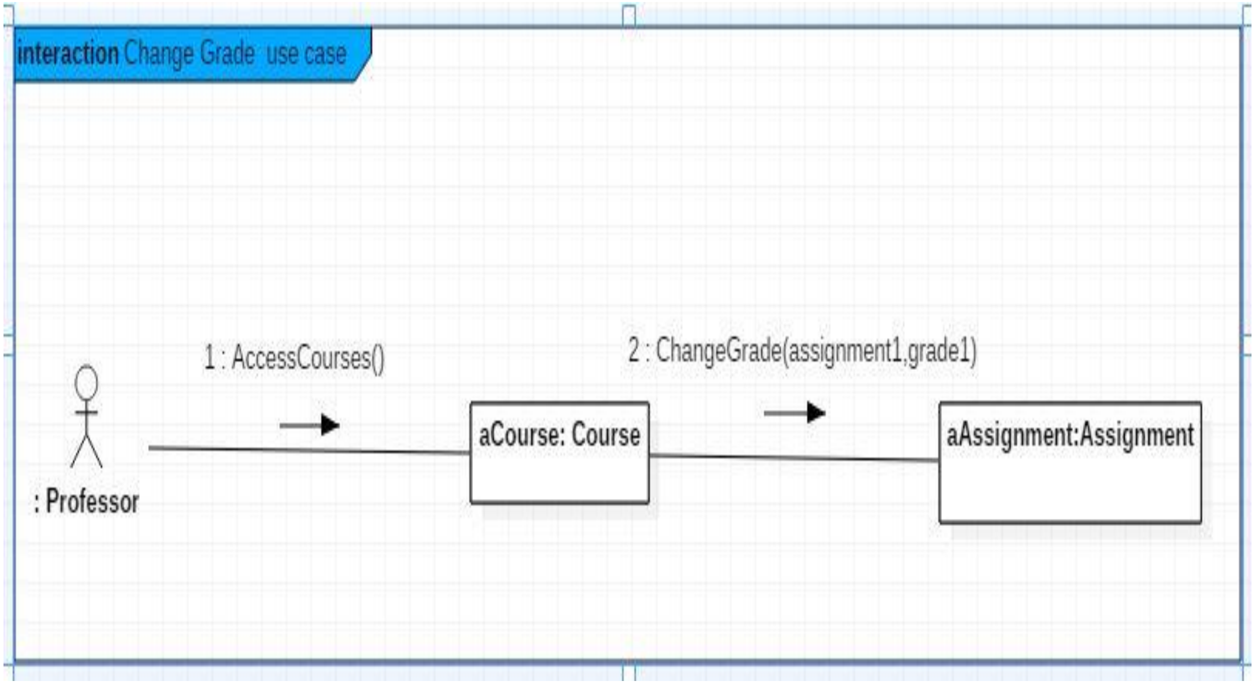


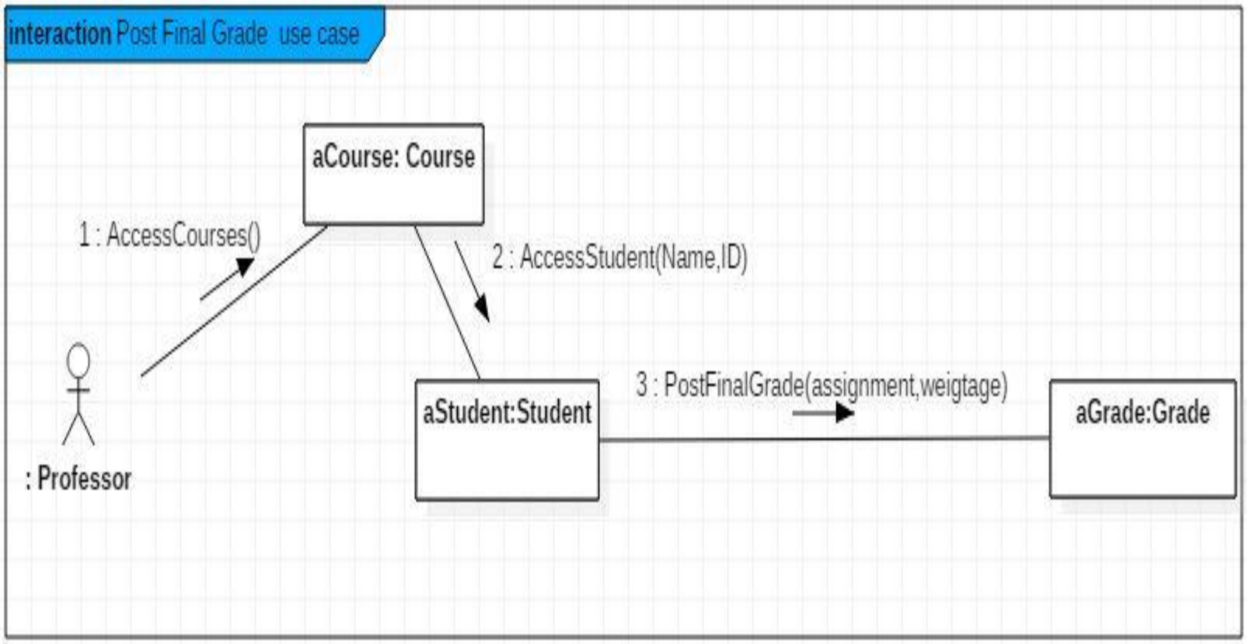



interaction Drop Class use case
aCourse: Course
1 : AccessCourses()
2 : DropCourse(course1)
: Student
:Enrollement Cart
3 : ConfirmDrop()
:Schedule


interaction Submit Feedback use case
aCourse: Course
1 : AccessCourses()
2 : GiveFeedback()
: Student
:Feedback
3 : SubmitAnonymousFeedback()
:Feedback Database


interaction Grade Assignment use case
1 : AccessCourses()
2 : GradeAssignment()
: Professor
aCourse: Course
aAssignment:Assignment

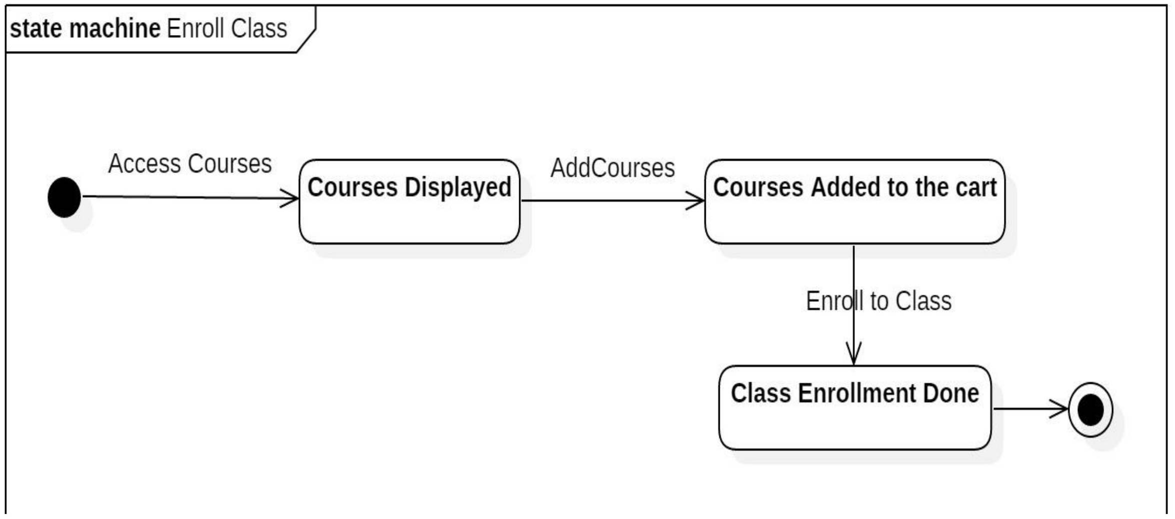

interaction Change Grade  use case
1 : AccessCourses()
2 : ChangeGrade(assignment1,grade1)
: Professor
aCourse: Course
aAssignment:Assignment

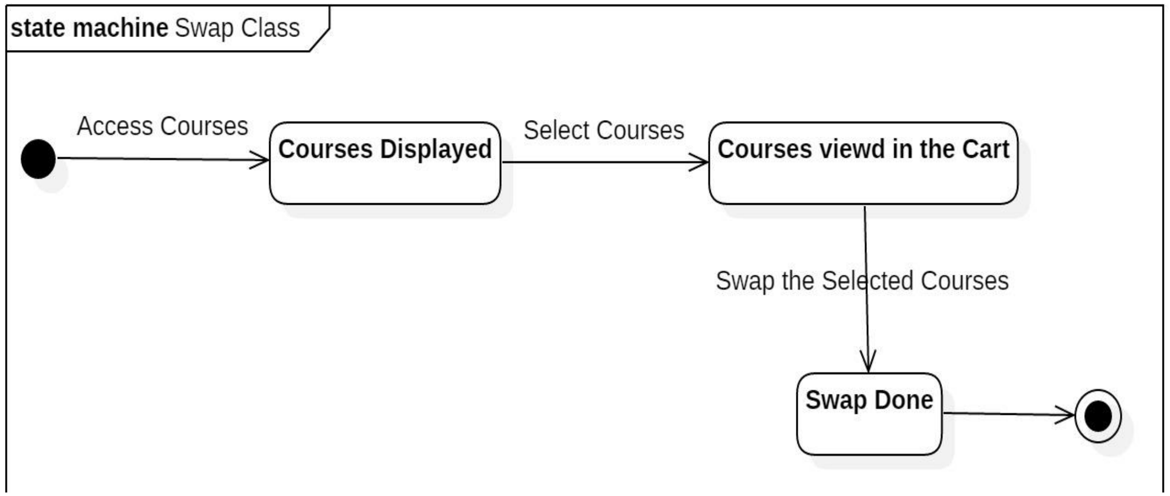

interaction Post Final Grade use case
aCourse: Course
1 : AccessCourses()
2 : AccessStudent(Name,ID)
: Professor
aStudent:Student
3 : PostFinalGrade(assignment,weigtage)
aGrade:Grade

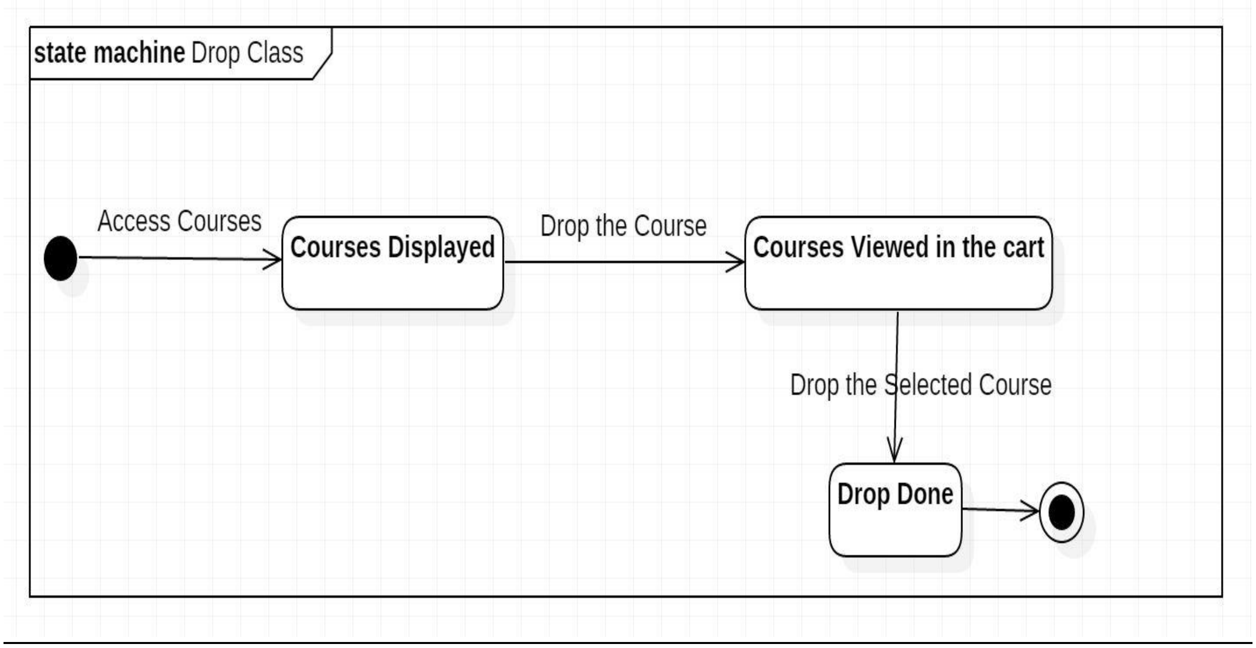
## CRUDE Matrix:

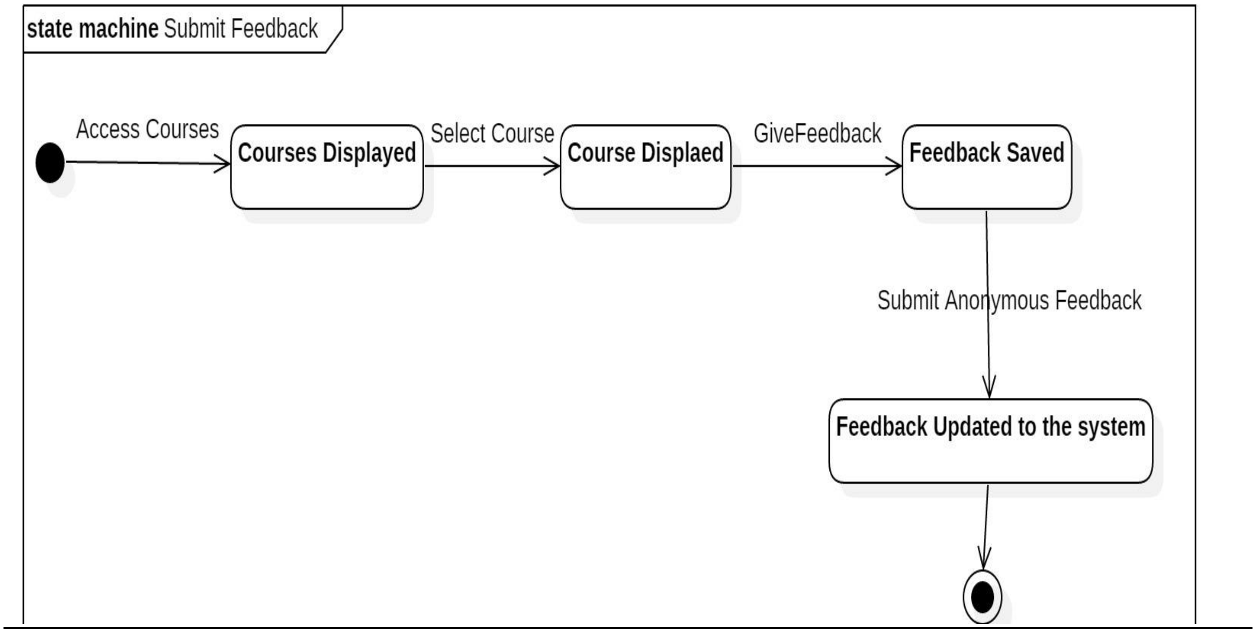
| | Student | New professor | Old Professor | System | Assignment | Grade Weightage | Feedback |
|---|---|---|---|---|---|---|---|
| *Student* | | | | CRUD | R | R | C |
| *New Professor* | | | | CUD | U | UD | R |
| *Old Professor* | | | | CUD | U | UD | RU |
| *System* | CRUD | | | | CRUD | | CRUD |
| *Assignment* | | | | U | | | |
| *Grade Weightage* | | | | U | | | |
| *Feedback* | | | | U | | | |

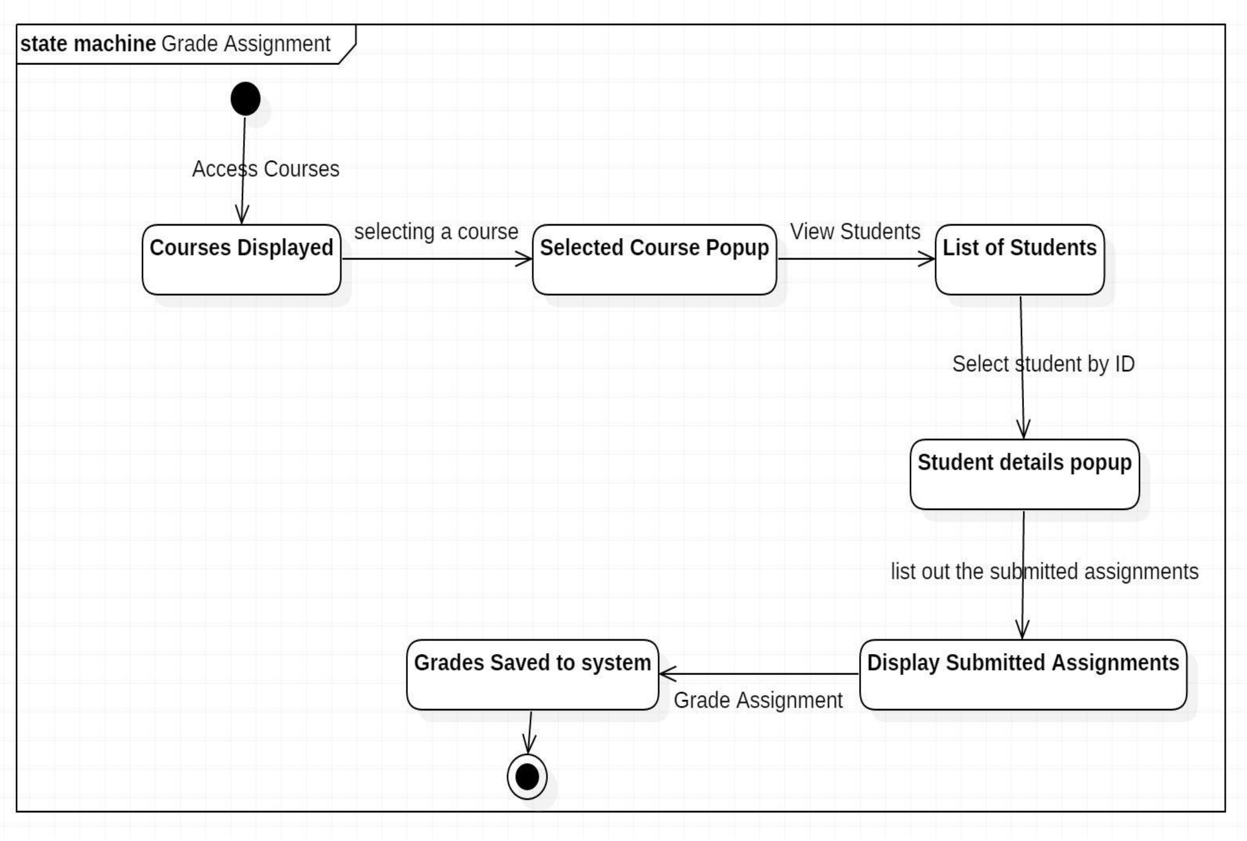
## Behavioral state diagrams

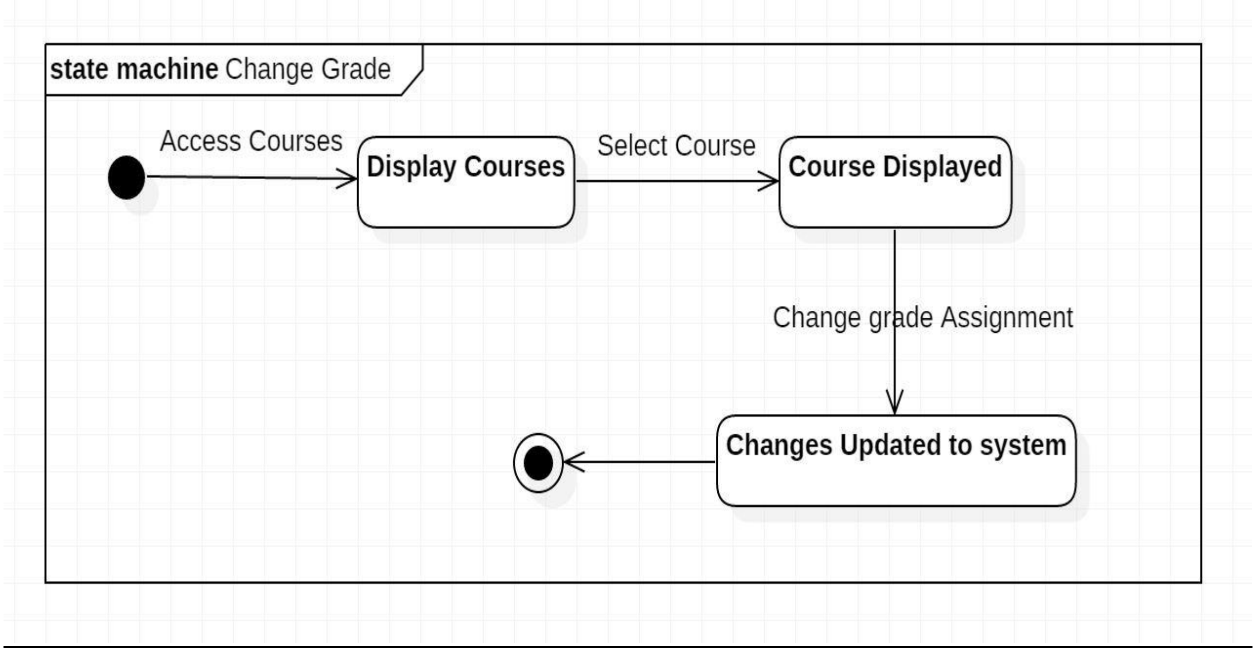


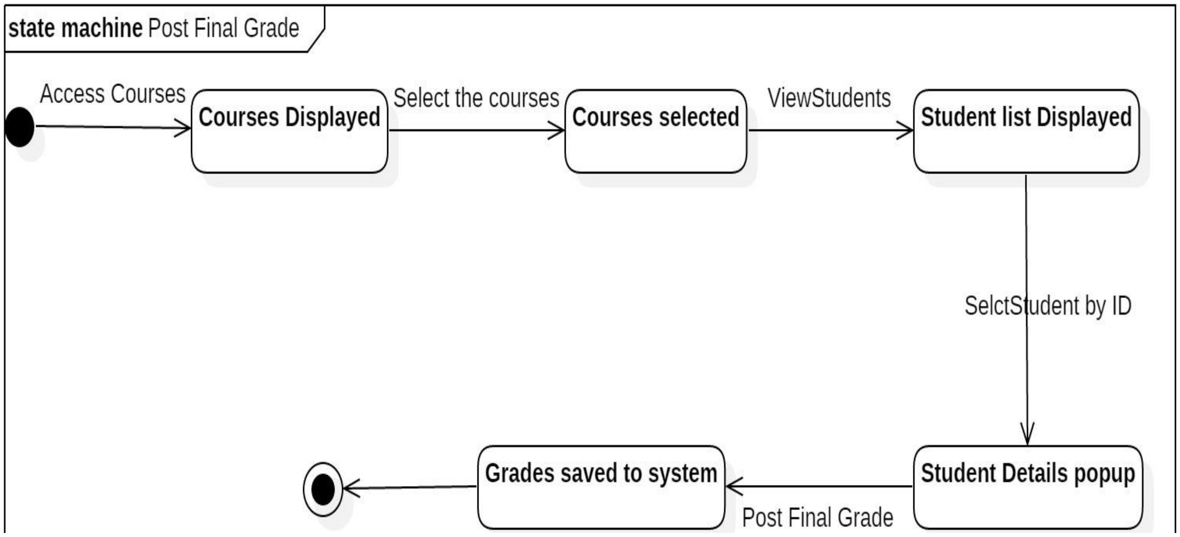


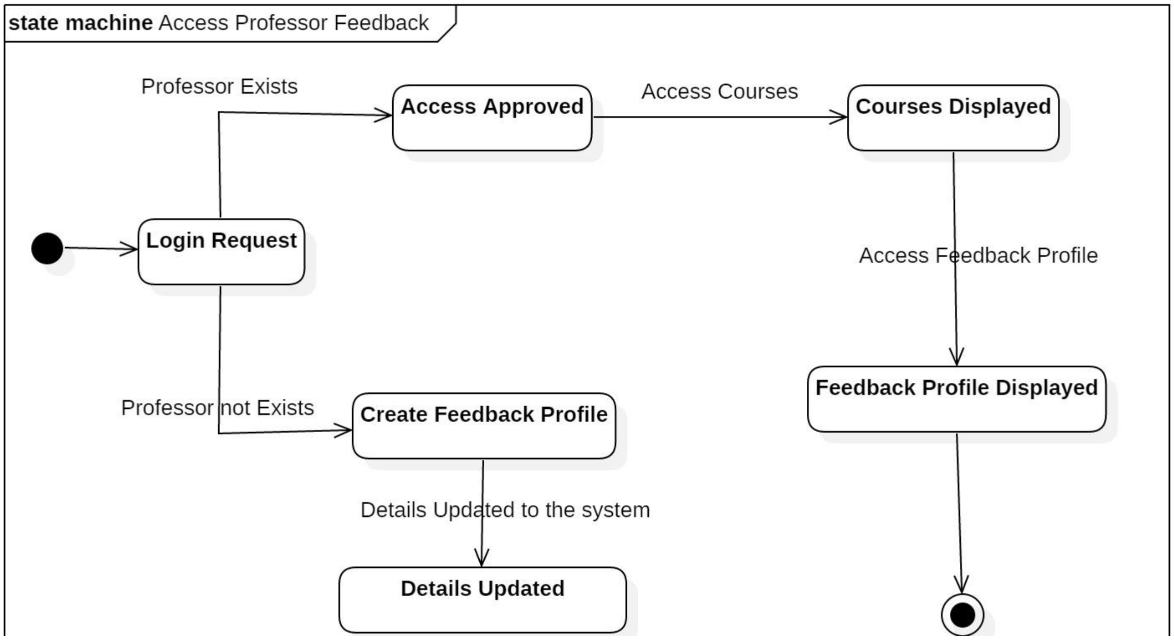

state machine Drop Class
Access Courses
Courses Displayed
Drop the Course
Courses Viewed in the cart
Drop the Selected Course
Drop Done


state machine Submit Feedback
Access Courses
Courses Displayed
Select Course
Course Displaed
GiveFeedback
Feedback Saved
Submit Anonymous Feedback
Feedback Updated to the system


state machine Grade Assignment
Access Courses
Courses Displayed
selecting a course
Selected Course Popup
View Students
List of Students
Select student by ID
Student details popup
list out the submitted assignments
Display Submitted Assignments
Grade Assignment
Grades Saved to system


state machine Change Grade
Access Courses
Display Courses
Select Course
Course Displayed
Change grade Assignment
Changes Updated to system


state machine Post Final Grade
Access Courses
Courses Displayed
Select the courses
Courses selected
ViewStudents
Student list Displayed
SelctStudent by ID
Student Details popup
Post Final Grade
Grades saved to system


state machine Access Professor Feedback
Professor Exists
Access Approved
Access Courses
Courses Displayed
Login Request
Access Feedback Profile
Feedback Profile Displayed
Professor not Exists
Create Feedback Profile
Details Updated to the system
Details Updated

Appendices:

1. <u>Functional Model (Verification and Validation results):</u>

Verification and Validation of Use-case diagram, Use-case descriptions and Activity Diagram

FOR User Group 7
BY Developer Group 9

1) At least one event recorded in the normal flow of events, sub-flows, or alternative/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Missing events in activity diagram
Access Old Prof. Feedback
Access New Prof. Feedback
Missing events in Use Case-diagram
N.A.

2) All objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, sub-flows, or alternative/exceptional flows of the use-case description:

No changes required

3) Sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram:

**Needs to be added for the sequential order in both use-case description and activity diagram**

Access Old Prof. Feedback
Access New Prof. Feedback
Creating feedback profile
Required changes have been made and new diagrams have been added in the final report.

**4) When comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa:**

The number of use-cases in the use-case diagram matches the number of use-case descriptions and vice-versa

**5) All actors listed in a use-case description must be portrayed on the use-case diagram:**

All actors listed in the use-case description are portrayed on the use-case diagram

**6) Include the stakeholders listed in the use-case description as actors in the use-case diagram:**

All stake holders are listed in the use-case description as actors in the use-case diagram.

**7) All other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram:**

All the relationships were accurately depicted in the use case diagram.

**8) Diagram-specific requirements that must be enforced:**

Activity Diagrams must be split for events with the flow of events. Additionally, when we examined the three functional models of our developer group, we found that the use case diagram, activity diagram and the use case description reflect the functional requirements and the business requirements in the system request match the functional requirements.

Suggested changes have been successfully incorporated by splitting the activity diagram.

2. **Structural Model (Verification and Validation Results):**

**Verification and Validation of Structural Model**
**(Performed by User Group # 9 on Developer Group # 7)**

- Every CRC card should be associated with a class on the class diagram, and vice versa:

  Class Diagram is missing the following classes
  - Feedback
  - Grade Weightage

  **Response from Developer Group:** Necessary changes has been made. Feedback Class and Grade Weightage Class has been added to the class diagram.

- The responsibilities listed on the front of the CRC card must be included as operations in a class on a class diagram, and vice versa.

  **No Change is required for this rule.**

- Collaborators on the front of the CRC card imply some type of relationship on the back of the CRC card and some type of association that is connected to the associated class on the class diagram.

**No Change is required for this rule.**

- Attributes listed on the back of the CRC card must be included as attributes in a class on a class diagram, and vice versa.

  Attributes are missing for the following classes in the class diagram

  - Student
  - Assignment

  **Response from the Developer Group:** Necessary changes has been made. Attributes for Student and Assignment class has been added in class diagram.

- The object type of the attributes listed on the back of the CRC card and with the attributes in the attribute list of the class on a class diagram implies an association from the class to the class of the object type.

  **No change is required for this rule.**

- The relationships included on the back of the CRC card must be portrayed using the appropriate notation on the class diagram.

  **No change is required for this rule.**

- An association class should be created only if there is indeed some unique characteristic (attribute, operation, or relationship) about the intersection of the connecting classes.

  **No change is required for this rule.**

- Specific representation rules must be enforced.

  **No change is required for this rule.**

3. Behavioral Model (Verification and Validation Results):

**Verification and Validation of Behavioral state machines and CRUDE Matrices (By User group 9 for Developer group 7)**

1) **Every actor and object included on a sequence diagram must be included as an actor**

**and an object on a communication diagram, and vice versa.**

Actors: Student, Professor and related objects that are included on the sequence diagram also appear in the communication diagram and vice versa

No changes required.

2) **If there is a message on the sequence diagram, there must be an association on the communications diagram, and vice versa.**

All messages that appear in the sequence diagram have an association included in the communication diagram and vice versa

No changes required.

3) **Every message that is included on a sequence diagram must appear as a message on an association in the corresponding communication diagram, and vice versa.**

All messages included on the sequence diagram appear on an association in the corresponding communication diagram, and vice versa

No changes required.

4) **If a guard condition appears on a message in the sequence diagram, there must be an equivalent guard condition on the corresponding communication diagram, and vice versa.**

All messages with a guard condition in the sequence diagram have an equivalent guard condition on the corresponding communication diagram and vice versa

No changes required.

5) **The sequence number included as part of a message label in a communications diagram implies the sequential order in which the message will be sent. Therefore, it must correspond to the top-down ordering of the messages being sent on the sequence diagram**

Sequence numbers included as part of message labels in the communication diagram correspond to the top-down ordering of messages being sent on the sequence diagram.

No changes required.

6) **All transitions contained in a behavior state machine must be associated with a message being sent on a sequence and communication diagram, and it must be classified as a (C)reate, (U)pdate, or (D)elete message in a CRUDE matrix.**

All transitions contained in a behavior state machine are associated with a message being sent on a sequence and communication diagram, and is classified as a (C)reate, (U)pdate, or (D)elete message in a CRUDE matrix.

No changes required.

7) **All entries in a CRUDE matrix imply a message being sent from an actor or object to another actor or object. If the entry is a (C)reate, (U)pdate, or (D)elete, then there must be an associated transition in a behavioral state machine that represents the instances of the receiving class.**

   All associated transitions exist. No changes required.

4. **Balancing the Models (Functional, Structural and Behavioral):**

- **<u>Balancing Functional and Structural Models:</u>**

1. **Every class on a class diagram and every CRC card must be associated with at least one use-case, and vice versa.**

   All classes on the class diagram, all CRC cards are associated with at least one use-case and vice-versa.

   No changes required.

2. **Every activity or action contained in an activity diagram and every event contained in a use-case description should be related to one or more responsibilities on a CRC card and one or more operations in a class on a class diagram and vice versa.**

Every activity or action contained in the activity diagram and every event contained in a use-case description are related to one or more responsibilities on a CRC card and one or more operations in a class on a class diagram and vice versa.

No changes required.

3. **Every object node on an activity diagram must be associated with an instance of a class on a class diagram (i.e., an object) and a CRC card or an attribute contained in a class and on a CRC card.**

   Every object node on an activity diagram is associated with an instance of a class on a class diagram (i.e., an object) and a CRC card or an attribute contained in a class and on a CRC card.

   No changes required.

4. **Every attribute and association/aggregation relationships contained on a CRC card (and connected to a class on a class diagram) should be related to the subject or object of an event in a use-case description.**

   Every attribute and association/aggregation relationships contained on a CRC card (and connected to a class on a class diagram) is related to the subject or object of an event in a use-case description.

   No changes required.

- <u>**Balancing Functional and Behavioral Models:**</u>

1. **The sequence and communication diagrams must be associated with a use case on the use-case diagram and a use-case description.**

   The sequence and communication diagrams are associated with a use case on the use-case diagram and a use-case description.

No changes required.

2. **Actors on sequence diagrams, communication diagrams, and/or CRUDE matrices must be associated with actors on the use-case diagram or referenced in the use case description, and vice versa.**

   Actors on sequence diagrams, communication diagrams, and/or CRUDE matrices are associated with actors on the use-case diagram or referenced in the use case description, and vice versa.

   No changes required.

3. **Messages on sequence and communication diagrams, transitions on behavioral state machines, and entries in a CRUDE matrix must be related to activities and actions on an activity diagram and events listed in a use-case description, and vice versa.**

   Messages on sequence and communication diagrams, transitions on behavioral state machines, and entries in a CRUDE matrix are related to activities and actions on an activity diagram and events listed in a use-case description, and vice versa.

   No changes required.

4. **All complex objects represented by an object node in an activity diagram must have a behavioral state machine that represents the object's lifecycle, and vice versa.**

   All complex objects represented by an object node in an activity diagram must have a behavioral state machine that represents the object's lifecycle, and vice versa.

   No changes required.

- Balancing Functional and Behavioral Models:

1. **Objects that appear in a CRUDE matrix must be associated with classes that are represented by CRC cards and appear on the class diagram, and vice versa.**

   Objects that appear in a CRUDE matrix are associated with classes that are represented by CRC cards and appear on the class diagram, and vice versa. No changes required.

2. **Because behavioral state machines represent the life cycle of complex objects, they must be associated with instances (objects) of classes on a class diagram and with a CRC card that represents the class of the instance.**

   Because behavioral state machines represent the life cycle of complex objects, they are associated with instances (objects) of classes on a class diagram and with a CRC card that represents the class of the instance.

   No changes required.

3. **Communication and sequence diagrams contain objects that must be an instantiation of a class that is represented by a CRC card and is located on a class diagram.**

   Communication and sequence diagrams contain objects that are an instantiation of a class that is represented by a CRC card and is located on a class diagram.

   No changes required.

4. **Messages contained on the sequence and communication diagrams, transitions on behavioral state machines, and cell entries on a CRUDE matrix must be**

**associated with responsibilities and associations on CRC cards and operations in classes and associations connected to the classes on class diagrams.**

Messages contained on the sequence and communication diagrams, transitions on behavioral state machines, and cell entries on a CRUDE matrix are associated with responsibilities and associations on CRC cards and operations in classes and associations connected to the classes on class diagrams.

No changes required.

5. **The states in a behavioral state machine must be associated with different values of an attribute or set of attributes that describe an object.**

   The states in a behavioral state machine are associated with different values of an attribute or set of attributes that describe an object.

   No changes required.