**Actor: Admin**

*Name:* Add Class

1. Admin enters the ‘Add’ tab and selects ‘Class’.
2. Admin inserts a class name, department, class times, room number, and a staff number into a query.
3. System creates a unique ID for the class (the section umber).

*Input*: CourseID, time, roomNumber

*Output:* A message stating completion, and shows the new section number

*Name*: Modify Class

1. Admin enters the ‘Modify’ tab, and selects ‘Class’
2. Admin inputs the section number of the class, and the system opens all data on that class.
3. Admin selects any data within that class and modifies it as needed. All changes are cascaded to its dependencies.

*Input*:  a section number

*Output:* A dialogue box that relays all modifications to the database.

*Name*: Remove Class

1. Admin enters the ‘Delete’ tab and selects ‘Class’
2. Admin inputs a section number into a new dialogue box. The system asks for confirmation.
3. The system deletes the class and modifies all dependent data to NULL

*Input*: a section number

*Output:* a box stating the deletion.

*Name*: Add Teacher

1. Admin enters the ‘Add’ tab and selects ‘Teacher’.
2. Admin inputs a first name, last name, and department that teacher is joining in.
3. System generates a unique staff ID for the new addition.

*Input*: A first name, last name, and a department

*Output:* a message of completion.

*Name*: Modify Teacher

1. Admin enters the ‘Modify’ tab, and selects ‘Teacher’
2. Admin inputs the teacher’s staff ID, and the system opens all data on that teacher.
3. Admin selects any data within and modifies it as needed. All changes are cascaded to its dependencies.

*Input*:  a staff ID

*Output:* A dialogue box that relays all modifications to the database.

*Name*: Remove Teacher

1. Admin enters the ‘Delete’ tab and selects ‘Teacher’.
2. Admin inputs a staff ID into a new dialogue box. The system asks for confirmation.
3. The system deletes the class and modifies all dependent data to NULL.

*Input*: a section number

*Output:* a box stating the deletion.

*Name*: Add Student

1. Admin enters the ‘Add’ tab and selects ‘Student’.
2. Admin enters in the first and last name into the dialogue box.
3. The system generates a new student ID.

*Input*: A first name and last name

*Output:* a box stating the completion, and shows the new student ID.

*Name*: Modify Student

1. Admin enters the ‘Modify’ tab, and selects ‘Student’
2. Admin inputs the student’s ID, and the system opens all data on that student.
3. Admin selects any data within and modifies it as needed. All changes are cascaded to its dependencies.

*Input*:  a student ID

*Output:* A dialogue box that relays all modifications to the database.

*Name*: Remove Student

1. Admin enters the ‘Delete’ tab and selects ‘Student’.
2. Admin inputs a student ID into a new dialogue box. The system asks for confirmation.
3. The system deletes the student data modifies all dependent data to NULL

*Input*: a student ID

*Output:* a box stating the deletion.

**Actor: Teacher**

*Name*: Add Assignment

1. Teacher selects button to “Add an Assignment”.
2. The system opens a dialogue box, and the teacher enters a name for the assignment, and enters the assignment instructions into a second box, and a due date. Additionally, the teacher may attach any documents to the assignment.
3. The system sends this to the database and creates a connection to any created documents (somehow).

*Input*: An assignment name, assignment instructions, a due date, and optionally one or multiple documents.

*Output:* a message of completion.

*Name*: Grade Assignment

1. Teacher searches a Class then Assignments\_AssignmentNo
2. Teacher grades assignments

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class), Grade(Assignments)

*Output:* Grade(Assignments) has an altered value, completion message

*Name*: Modify Assignment

1. Teacher searches a Class then Assignments\_AssignmentNo
2. Teacher modifies Assignments of that assignmentNo

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class)

*Output:* General modifications are made to assignment, completion/verification message

*Name*: Modify Grade

1. Teacher searches a Class then Assignments\_AssignmentNo, or Class then Student\_StudentID
2. Teacher changes Grade(Assignments)

*Input*:  SectionNo(Class), Assignments\_assignmentNo(Class) or  SectionNo(Class), Student\_StudentID(Class)

*Output:* Grade for a specific assignment is changed, completion message/ save changes message appears.

*Name*: Remove Assignment

1. Teacher searches a Class then Assignments\_AssignmentNo
2. Teacher removes assignment

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class)

*Output:* Assignment is removed from class, a save changes message/ completion message displays

*Name*: View Class Roster

1. Teacher searches a Class then Student\_StudentID

*Input*: SectionNo(Class), Student\_StudentID(Class)

*Output:* Displays students in searched class

*Name*: Upload File

1. Teacher searches a Class then Assignments\_AssignmentNo
2. Create Class Material, documentNo is assigned, rest of properties are entered, file is uploaded

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class), name(Class Material), docType(Class Material), Class)sectionNo(Class Material)

*Output:* Displays uploaded file, save changes message/completion message displays

*Name*: Delete File

1. Teacher searches a Class then Assignments\_AssignmentNo then Class Material\_documentNo(Assignments)
2. delete Class Material

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class), Class Material\_documentNo(Assignments)

*Output:* save changes message/completion message displays

**Actor: Student**

*Name*: View Personal Assignment

1. Student searches their studentID union all assignments
2. shows all assignments for all classes for that

*Input*: StudentID(Student)

*Output:* list of all assignments for that student

*Name*: View Uploaded File

1. Student searches a Class then Assignments\_AssignmentNo then Class Material\_documentNo(Assignments)

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class), Class Material\_documentNo(Assignments)

*Output:* Displays uploaded file searched for

*Name*: Submit Assignment

1. Student searches a Class then Assignments\_AssignmentNo
2. Student makes a submission

*Input*: SectionNo(Class), Assignments\_assignmentNo(Class), make submission

*Output:* save changes message/completion message displays

*Name*: Check Grades

1. Student searches a Class then Assignments\_AssignmentNo or student searches all assignments with their studentID
2. Student views grade(Assignments)

*Input*:  SectionNo(Class), Assignments\_assignmentNo(Class)  or StudentID

*Output:* Individual assignment is displayed, or all assignments are displayed. Student can view grades.

*Name*: View Report

1. Actor inserts class name, department, and class times into a query.
2. System creates a unique ID for the class.

*Input*:

*Output:*

*Name:* Sign up to Classes

1. Actor inserts class name, department, and class times into a query.
2. System creates a unique ID for the class.

*Input*:

*Output:*