1. Requirements:

The software to be designed will act as a sub-system of a learning management software application for the administration, documentation, tracking, reporting, and delivery of educational courses for an educational institution.

This software application will only hold one semester’s worth of information.

This software application will store and maintain the basic information of a student, faculty, courses for that semester, and administrator for the current semester.

This software application will allow have two accessing modes: administrator and user.

This software application will be managed by administrators. The administrator will be able to insert, update, and delete records in all the tables included with the software application. When a user logs in the administrator should be able to view the details of a student logged in, but cannot insert, update, and/or delete information on the student’s profile.

1. User Interface
   1. This software application will implement the two user interfaces with a “standard GUI” style application.
   2. A user interface like the following might be adopted. Not include is a File menu with New, Open, Close, Save, Save As…, Print, and Quit options. For the “Edit” and “Delete” buttons, the user must first select a person in the scrolling list of names, and then can click the appropriate button to edit/delete that person.

A screenshot of a cell phone

Description automatically generated

1. **Use Cases**
   1. In general, the following users should be able the following:

A close up of a map

Description automatically generated

* 1. Student must be able to:
     1. Login to user console
     2. View Classes by Name
     3. Open a Class table by Name – to see their grades
     4. View GPA for the Semester
     5. Logout of application
  2. Faculty must be able to:
     1. Login to user console
     2. View Classes by Name
     3. Open a Class Table by Name
     4. Edit a Class Table by Name
     5. Logout of application
  3. Admin must be able to:
     1. A picture containing text

        Description automatically generatedAdd a student
     2. Edit a student
     3. Delete a student
     4. Add a faculty
     5. Edit a faculty
     6. Delete a faculty
     7. Add a course
     8. Edit a course
     9. Delete a course

Admin wants to add a student record

1. Admin logs into admin console
2. Admin clicks on student table icon
3. Admin opens student table
4. Admin is able to sort student table by student\_last\_name and/ or student\_id
5. Admin adds new student onto student table
6. Admin saves changes to student table

Admin wants to edit a student record

1. Admin logs into admin console
2. Admin clicks on student table icon
3. Admin opens student table
4. Admin is able to sort student table by student\_last\_name and/ or student\_id
5. Admin selects desired student on student table
6. Admin edits student record
7. Admin saves changes to student table

Admin wants to delete a student record

1. Admin logs into admin console
2. Admin clicks on student table icon
3. Admin opens student table
4. Admin is able to sort student table by student\_last\_name and/ or student\_id
5. Admin selects desired student on student table
6. Admin deletes selected student record
7. Admin saves changes to student table

Admin wants to add a faculty record

1. Admin logs into admin console
2. Admin clicks on faculty table icon
3. Admin opens faculty table
4. Admin is able to sort student table by faculty\_last\_name and/ or staff\_id
5. Admin adds new faculty onto faculty table
6. Admin saves changes to faculty table

Admin wants to edit a faculty record

1. Admin logs into admin console
2. Admin clicks on faculty table icon
3. Admin opens faculty table
4. Admin is able to sort student table by faculty\_last\_name and/ or staff\_id
5. Admin selects desired faculty on faculty table
6. Admin edits faculty record
7. Admin saves changes to faculty table

Admin wants to delete a faculty record

1. Admin logs into admin console
2. Admin clicks on faculty table icon
3. Admin opens faculty table
4. Admin is able to sort student table by faculty\_last\_name and/ or staff\_id
5. Admin selects desired faculty on faculty table
6. Admin deleted selected faculty record
7. Admin saves changes to faculty table

Admin wants to add a course record

1. Admin logs into admin console
2. Admin clicks on course icon
3. Admin opens course table
4. Admin is able to sort student table by course\_name and/ or course\_id
5. Admin adds new course onto course table
6. Admin saves changes to course table

Admin wants to edit a course record

1. Admin logs into admin console
2. Admin clicks on course icon
3. Admin opens course table
4. Admin is able to sort student table by course\_name and/ or course\_id
5. Admin selects desired course on course table;
6. Admin edits selected course record
7. Admin saves changes to course table

Admin wants to delete a course record

1. Admin logs into admin console
2. Admin clicks on course icon
3. Admin opens course table
4. Admin is able to sort student table by course\_name and/ or course\_id
5. Admin selects desired course on course table;
6. Admin deletes selected course record
7. Admin saves changes to course table
8. **1st Iteration of CRC Cards // i used the appendix g of the text book to create these**

|  |  |
| --- | --- |
| **Module name** | StudentClass :: addNewStudent |
| **Module type** | method |
| **Return type** | **void** |
| **Input parameter** | none |
| **Output parameter** | none |
| **Error messages** | none |
| **Filess accessed** | STUDENT\_RECORDS\_TABLE\_TEMP, STUDENT\_RECORDS\_TABLE |
| **Files changed** | STUDENT\_RECORDS\_TABLE |
| **Modules called** |  |
| **Narrative** | Inserts a student object in alphabetical order (by Student Last Name) or by ascending order (by student id number) into the STUDENT\_RECORDS \_TABLE. This operation is performed by first copying STUDENT\_RECORDS\_TABLE to STUDENT\_RECORDS\_TABLE\_TEMP. The new object is then inserted into the correct location while STUDENT\_RECORDS\_TABLE\_TEMP is copied back to STUDENT\_RECORDS\_TABLE record by record. |

|  |  |
| --- | --- |
| **Module name** | FacultyClass :: addNewFaculty |
| **Module type** | method |
| **Return type** | **void** |
| **Input parameter** | none |
| **Output parameter** | none |
| **Error messages** | none |
| **Filess accessed** | FACULTY\_RECORDS\_TABLE\_TEMP, FACULTY\_RECORD\_TABLE |
| **Files changed** | STUDENT\_RECORD\_TABLE |
| **Modules called** |  |
| **Narrative** | Inserts a faculty object in alphabetical order (by Student Last Name) or by ascending order (by faculty id number) into the FACULTY\_RECORDS\_TABLE. This operation is performed by first copying FACULTY\_RECORDS\_TABLE to FACULTY\_RECORDS\_TABLE\_TEMP. The new object is then inserted into the correct location while FACULTY\_RECORDS\_TABLE\_TEMP is copied back to FACULTY\_RECORDS\_TABLE record by record. |

|  |  |
| --- | --- |
| **Module name** | CourseClass :: addNewCourse |
| **Module type** | method |
| **Return type** | **void** |
| **Input parameter** | none |
| **Output parameter** | none |
| **Error messages** | none |
| **Filess accessed** | COURSE\_RECORDS\_TABLE\_TEMP, COURSE\_RECORDS\_TABLE |
| **Files changed** | COURSE\_RECORDS\_TABLE\_TEMP, |
| **Modules called** |  |
| **Narrative** | Inserts a course object in alphabetical order (by course name) or by ascending order (by course id number) into the COURSE\_RECORDS\_TABLE. This operation is performed by first copying COURSE\_RECORDS\_TABLE to COURSE\_RECORDS\_TABLE\_TEMP. The new object is then inserted into the correct location while COURSE\_RECORDS\_TABLE\_TEMP, is copied back to COURSE\_RECORDS\_TABLE\_TEMP, record by record. |