Matthew Abney

CSC 440 – Applied software engineering

Individual Project: grade and gpa calculator and degree progress system

September 14, 2019

Final report

Table of Contents

[Table of Figures 2](#_Toc27002269)

[I. Introduction 3](#_Toc27002270)

[A. Problem Statement 3](#_Toc27002271)

[B. Proposal 3](#_Toc27002272)

[II. System Description 3](#_Toc27002273)

[III. System Requirements 4](#_Toc27002274)

[A. Functional Requirements 4](#_Toc27002275)

[B. Nonfunctional Requirements 17](#_Toc27002276)

[IV. Use Case Diagram 18](#_Toc27002277)

[V. DFD 19](#_Toc27002278)

[VI. Structure Chart 21](#_Toc27002279)

[VII. Data Store Design 22](#_Toc27002280)

[VIII. Algorithm Design 22](#_Toc27002281)

[IX. Conclusion 23](#_Toc27002282)

[X. Data Dictionary 23](#_Toc27002283)

# Table of Figures

[Figure 1 Adding a Grade 4](#_Toc27002284)

[Figure 2 Add grade form with invalid input 5](#_Toc27002285)

[Figure 3 course page with added assignment 5](#_Toc27002286)

[Figure 4 Performing What-If Grade Calculation 6](#_Toc27002287)

[Figure 5 Viewing What-If Results 6](#_Toc27002288)

[Figure 6 Confirmation window for delete grade 7](#_Toc27002289)

[Figure 7 Class page with assignment we want to change 7](#_Toc27002290)

[Figure 8 Modify grade form with values of grade 8](#_Toc27002291)

[Figure 9 Grade with modified values 8](#_Toc27002292)

[Figure 10 Assignment table with changed grade 9](#_Toc27002293)

[Figure 11 Course list with new class button 9](#_Toc27002294)

[Figure 12 New course form 10](#_Toc27002295)

[Figure 13 New course form with invalid field 11](#_Toc27002296)

[Figure 14 Class list with new class 11](#_Toc27002297)

[Figure 15 Course page for CSC 496 with delete class button 12](#_Toc27002298)

[Figure 16 Confirmation window for delete class 12](#_Toc27002299)

[Figure 17 Modify course form with values added 13](#_Toc27002300)

[Figure 18 Modify course with valid changes 14](#_Toc27002301)

[Figure 19 View progress form with GPA 15](#_Toc27002302)

[Figure 20 View progress after hours have been calculated 15](#_Toc27002303)

[Figure 21 Page header with links to view classes by status 16](#_Toc27002304)

[Figure 22 List of current classes 16](#_Toc27002305)

[Figure 23 List of assignments for a chosen course 17](#_Toc27002306)

[Figure 24 Use Case Diagram 18](#_Toc27002307)

[Figure 20 DFD Context Level 19](#_Toc27002308)

[Figure 21 DFD Level 1 20](#_Toc27002309)

[Figure 22 DFD Level 2 What-if GPA 20](#_Toc27002310)

[Figure 23 DFD Level 2 What-if Grade 21](#_Toc27002311)

[Figure 24 DFD Level 2 Modify Grades 21](#_Toc27002312)

[Figure 25 Structure Chart 21](#_Toc27002313)

# Introduction

## Problem Statement

Computer science students at Eastern Kentucky University need to find out their grades in certain classes. They will have their grades on several completed assignments, with more assignments to come. They need to know what grade(s) they will need on future assignment(s) in order to receive, say, 89.5% overall in a course. Perhaps they would like to know what grade they will get overall in a course, assuming they make, say, 75% on remaining assignments and/or tests.

Additionally, computer science students needto keep track of their GPA. They may want to know their GPA in different areas, such as their major GPA, overall GPA, and GPA in supporting courses. And, before a semester is over, a computer science student would like to know what effects different final grades will have on the student’s overall GPA.

One final piece of information that computer science students would like to keep track of is their progress towards their concentration specific C.S. degree. C.S. students would like to track which classes they need to take (general education courses, supporting courses, core courses, etc.).

## Proposal

My solution to the needs of computer science students at Eastern Kentucky University is a grade and GPA calculator, with added functionality to keep track of a student’s progress towards degree completion. The grade calculator would allow students to record grades on assignments/tests and perform “what-if” scenarios, showing them what grades they would need on remaining coursework in order to receive a certain final grade overall (such as 88%) and what grade they would end up with in a course if they received a specified grade on remaining coursework.

For the GPA portion of my application, the application would calculate a student’s overall GPA as they enter different final grades. The app would show a student what GPA he/she would have if certain final grades were achieved in current or future courses.

Lastly, this app would track a student’s progress towards a degree concentration. It is tailored specifically for computer science students at Eastern Kentucky University, taking one of EKU’s C.S. concentrations.

# System Description

The project is to build a system that allows students to input grades in for individual classes to help calculate the grade in that class, also allowing to calculate GPA, and know how close they are to fulfilling their majors’ requirements. The system shall give the student options to add or remove grades as needed while doing the same for classes current or completed. The system shall allow the student to modify the grades or classes. The system shall allow the student to perform what-if scenarios on grades and overall GPA.

# System Requirements

The system is required to give information to EKU computer science students regarding their degree progress, GPA, and grades in current classes. The system tracks the student’s GPA as he/she submits final grades, and allows a student to see what GPA he/she would have if he/she received certain final grades in future classes (that is, this system can perform “what-if” calculations). The system uses the information about completed classes to show a user his/her progress towards the different C.S. concentrations that EKU offers (users may view their progress under the different concentrations, so a user is not “stuck” viewing progress for merely one concentration). Users may submit their grades for assignments, quizzes, tests, etc. in current classes and view their overall grade in the courses. Then, they may ask the calculator what grade they would need on remaining coursework in order to finish with a certain grade overall.

## Functional Requirements

R1. The system shall allow a user to submit grades for completed assignments/tests.

* 1. After the user has chosen a course to view (see R12), they shall click the “Add Grade” button.
  2. The system shall render a form that the user can place the grade information into.

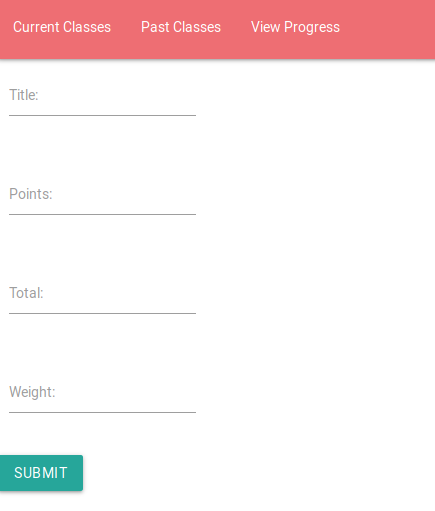


Figure 1 Adding a Grade

* 1. The user shall enter the assignment/test description (e.g. “Assignment 1”), the grade received on the assignment (as a percentage), the number of points the assignment is worth and the assignment’s weight (as a percentage).
  2. The user shall press the “Submit Changes” button.
  3. The system shall validate the user’s inputs to check if they match the required format
     1. The system shall mark the proper inputs as red if the information contained is not valid

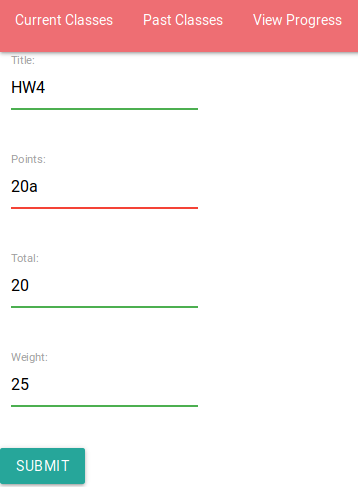


Figure 2 Add grade form with invalid input

* + 1. If the inputs are valid, the system shall send a POST request to the server to add the grade.
  1. The system shall redirect to the assignment list of the course.

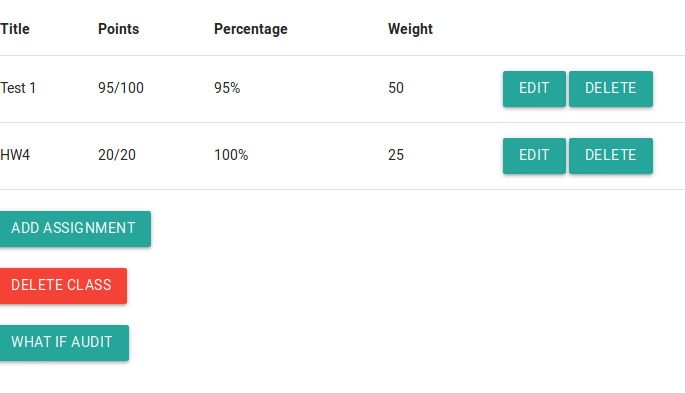


Figure 3 course page with added assignment

R2. The system shall allow a user to figure out what grades will be needed on remaining assignments in order to receive a desired overall grade in a course. Additionally, the system shall tell the user what grade he/she will have in the course, given a certain grade on remaining coursework.

* 1. After a user has chosen a course to view (see R12), they shall click the “What-If Audit” button
  2. The user shall enter their estimated grade for their remaining course work in the first input and their desired overall grade in the second as a percentage.

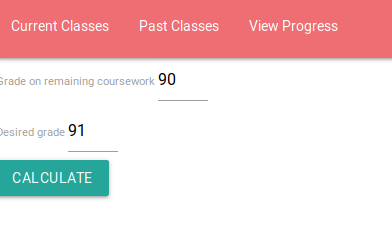


Figure 4 Performing What-If Grade Calculation

* 1. The user shall click the Submit button.
  2. The system shall validate the user’s inputs to ensure they match a positive number.
     1. If an input is invalid, the system shall mark the input as invalid and prevent the form from submitting.
     2. If the inputs are valid, the system shall perform the calculations and display the result.

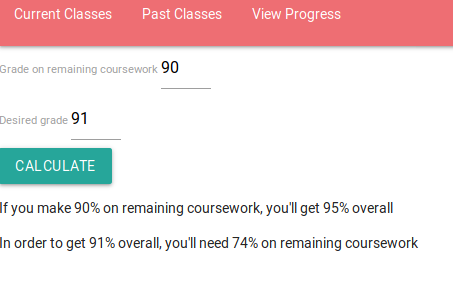


Figure 5 Viewing What-If Results

R3. The system shall allow a user to delete a grade.

* 1. The user shall navigate to the page of the course that the grade is part of.
  2. The user shall press the “Delete” button of the grade they wish to delete.
  3. The system shall render a confirmation window.

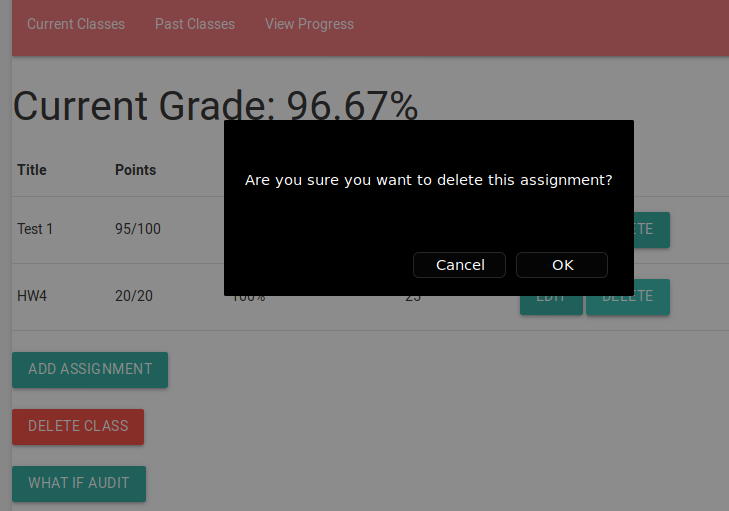


Figure 6 Confirmation window for delete grade

* 1. The user shall click the ‘OK’ button.
  2. The system shall send a delete request to the server.
     1. If the delete request was successful, the system shall reload the course page.

R4. The system shall allow a user to modify a grade.

* 1. The user shall navigate to the page of the course that contains the grade.

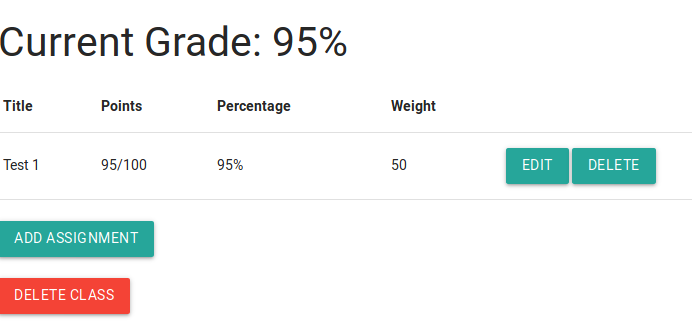


Figure 7 Class page with assignment we want to change

* 1. The user shall press the “Edit” button on the row of the grade they want to modify.
  2. The system shall redirect to the modify grade page.
     1. The system shall render a form containing the current values of the grade.

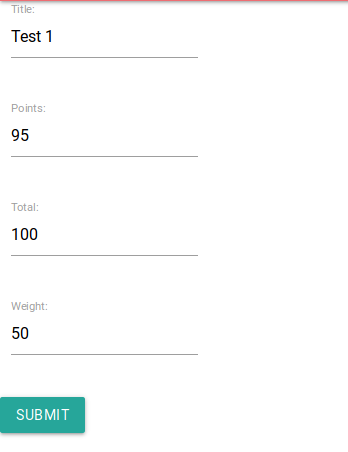


Figure 8 Modify grade form with values of grade

* 1. The user shall make their desired changes.
  2. The user shall press the “Submit” button once they are finished.
  3. The system shall validate the inputs.
     1. If an input is invalid, the system shall mark it as red and prevent the submission of the changes.

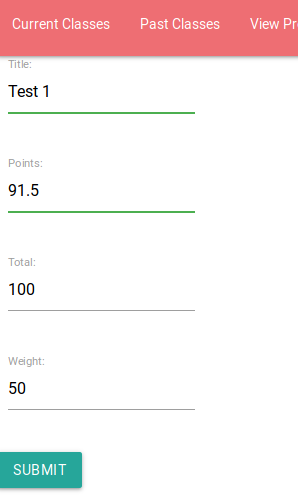


Figure 9 Grade with modified values

* 1. The system shall submit the changes to the server and redirect to the course’s page if successful.

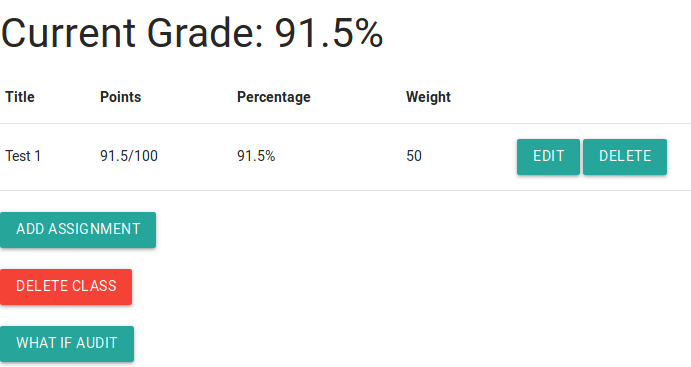


Figure 10 Assignment table with changed grade

R5. The system shall allow a user to add a class.

* 1. The user shall navigate to any list of courses (R9)

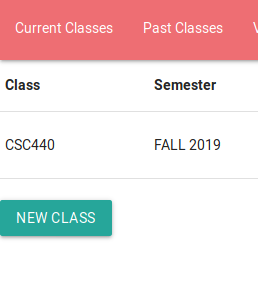


Figure 11 Course list with new class button

* 1. The user shall press the ‘Add Course’ button.
  2. The system shall redirect to the add course page and render a form for the course prefix, course number, credit hours, final grade (optional), semester, year, and status.

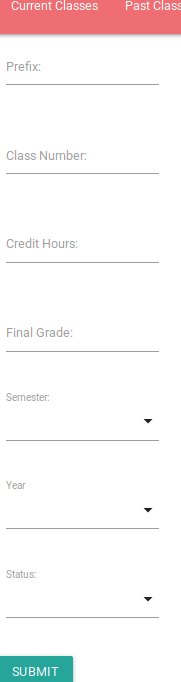


Figure 12 New course form

* 1. The user shall enter the above information and press ‘Submit’.
  2. The system shall validate the user’s input.
     1. If an input is invalid, the system shall mark the field as invalid and prevent the form from submitting.

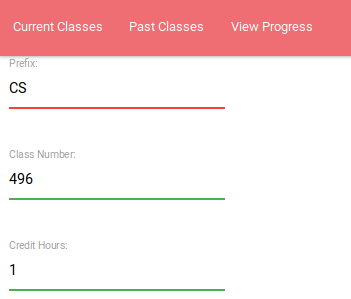


Figure 13 New course form with invalid field

* + 1. If the inputs are valid, the system shall add the new course to the database.
  1. The system shall redirect to the list of classes.

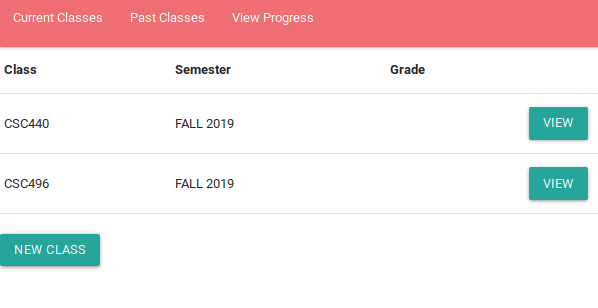


Figure 14 Class list with new class

R6. The system shall allow a user to delete a class.

* 1. The user shall navigate to the class’ page R10.

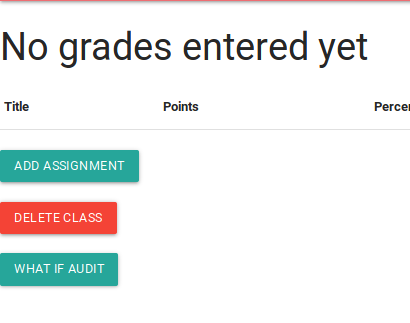


Figure 15 Course page for CSC 496 with delete class button

* 1. The user shall press the ‘Delete Class’ button.
  2. The system shall display a confirmation window.

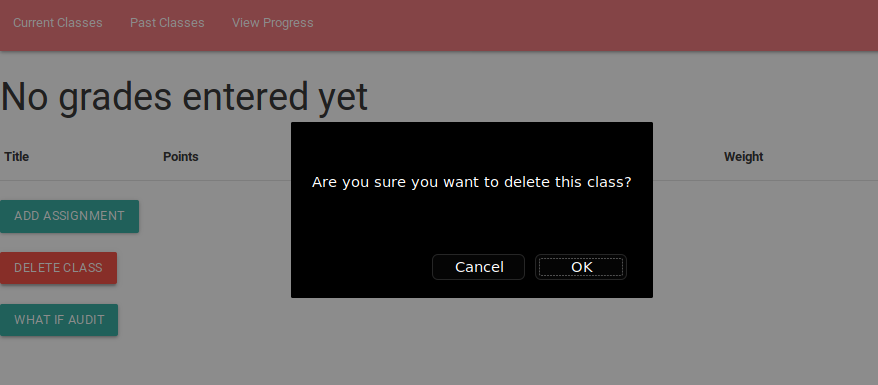


Figure 16 Confirmation window for delete class

* 1. The user shall press the ‘OK’ button.
  2. The system shall delete the class from the database.
     1. The system shall redirect back to a list of courses if the deletion was successful.

R7. The system shall allow a user to modify a class

* 1. The user shall navigate to course list containing the class.
  2. The user shall press the ‘Edit’ button corresponding to the class they want to edit.
  3. The system shall redirect to the modify class page and fill in the current values of the class in the fields.

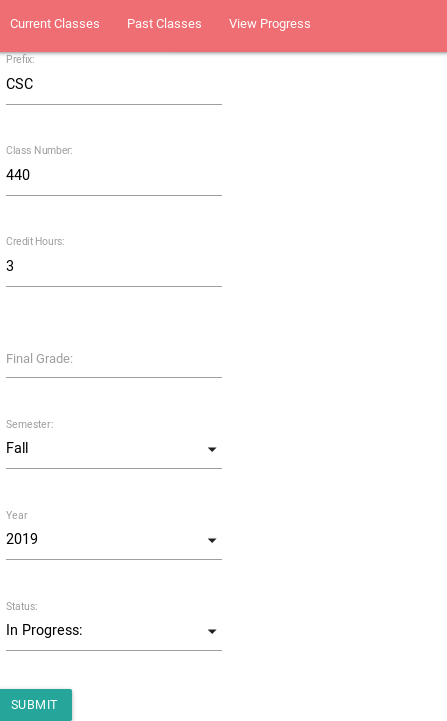


Figure 17 Modify course form with values added

* 1. The user shall modify the information they want to change.
  2. The system shall validate the changed information.
     1. If the input is invalid, the input field should be marked red and submission must be disabled.

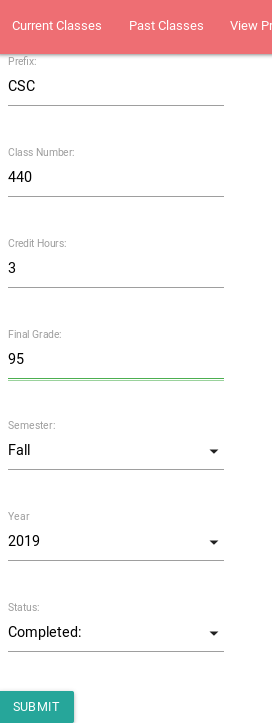


Figure 18 Modify course with valid changes

* 1. The user shall press the ‘Submit’ button.
  2. The system shall update the information in the database and redirect to the list of classes.

R8. The system shall track a user’s degree progress.

* 1. The user shall press ‘View Progress’ on the navigation header.
  2. The system shall redirect to a page containing a menu for the concentration
     1. The system shall retrieve the user’s class list and calculate the current GPA.

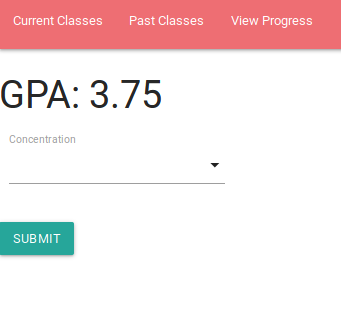


Figure 19 View progress form with GPA

* 1. The user shall select their concentration and press ‘Submit’
  2. The system shall calculate and display the user’s current progress for that concentration.

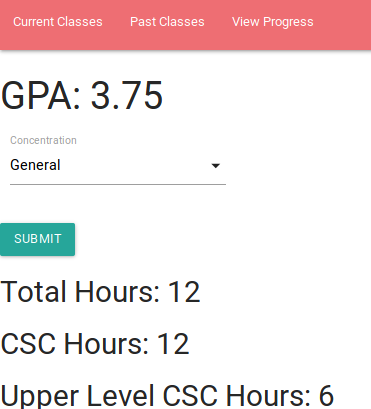


Figure 20 View progress after hours have been calculated

R9. The system shall display the classes marked with a given status.

* 1. The user shall press ‘Current Classes’ or ‘Past Classes’.

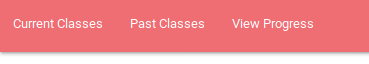


Figure 21 Page header with links to view classes by status

* 1. The system shall load the list of classes page.
  2. The system shall create a table with all the classes marked with the given status with the class name, semester, and final grade (if applicable).

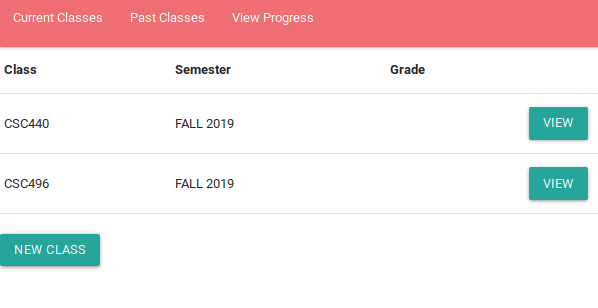


Figure 22 List of current classes

R10. The system shall display the grade and list of grades for a given course.

* 1. The user shall navigate to a list of classes (R9).
  2. The user shall press the ‘View’ button corresponding to the desired class.
  3. The system shall load a page containing the information for the class.
     1. The system shall calculate the current grade of a class and display it at the top.
     2. The system shall render a table of the grades entered for the course with the title, points, percentage and weight of the grade.

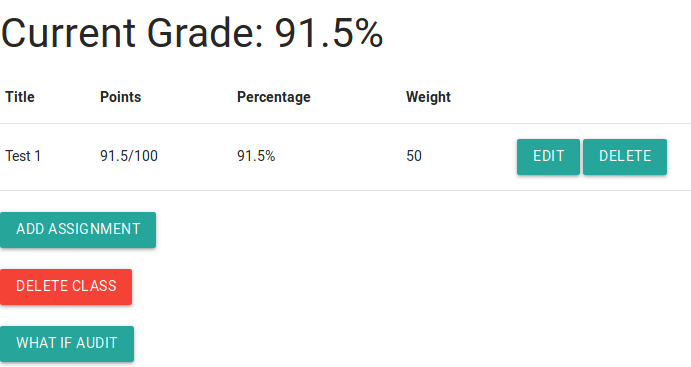


Figure 23 List of assignments for a chosen course

## Nonfunctional Requirements

NR1. All buttons must have a purpose

* 1. When a button is pressed it will have an outcome of some sort, whether it be visible or not.

NR2. System shall not crash under invalid data input.

2.1. When the user enters data into the system that is wrong data type or just invalid the system will error check and inform the user.

NR3. Any information in the database should be secure.

3.1. Only those who have access to that information should receive that information.

NR4. The functions of this system should be consistent and reliable.

4.1. The system should always output the same content given the same input.

# Use Case Diagram

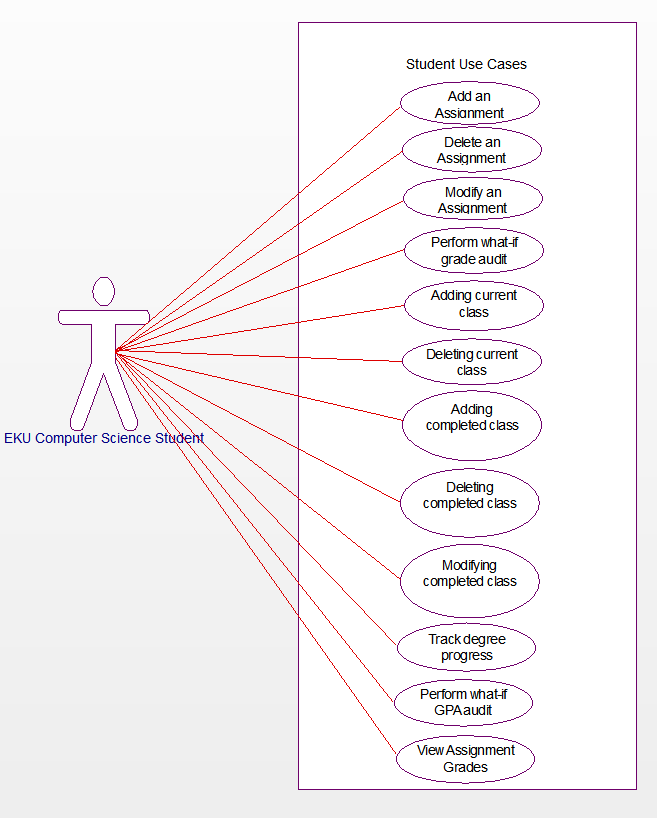


Figure 24 Use Case Diagram

Adding a grade – Use Case to add a grade for a class to the system.

Deleting an assignment grade – Use Case to delete an assignment/test/quiz grade from the system.

Modifying an assignment grade – Use Case to modify a grade in the system.

Perform what-if grade audit – Use Case for a student to see what his/her grade will be given certain grades on future assignments.

Delete a current class – Use Case to delete a current class.

Adding a current class – Use Case to add a current class. Once a current class is added, user can submit grades for that class.

Adding a completed class – Use Case to add information about a completed class (such as class name, final grade, etc.).

Deleting a completed class – Use Case to delete a completed class.

Modifying a completed class – Use Case to modify information about a completed class (e.g., class name).

Track degree progress – Use Case to see how close you are to graduating.

Perform what-if GPA audit – Use Case to see what your GPA will be given certain final grades.

View Current Classes – Use Case for the student to see what their current classes are.

# DFD



Figure 20 DFD Context Level



Figure 21 DFD Level 1



Figure 22 DFD Level 2 What-if GPA



Figure 23 DFD Level 2 What-if Grade



Figure 24 DFD Level 2 Modify Grades

# Structure Chart



Figure 25 Structure Chart

# Data Store Design

Class(CourseID, Credits)

Grades(GradeID, CourseID)

Courses(CourseID, isCompleted)

# Algorithm Design

1. Calculate What-If GPA

This algorithm will retrieve all the grades from the system and calculate the overall GPA by multiplying hours by grade and adding them all together and divide by hours completed.

Public void whatIf{

Gpa = Get gpa

Fg = Get fake grades

Fgw = Get fake grade weight

Gpa = (gpa+fg)/(fgw+1)

}

1. Add Grade

This algorithm will add a grade to the system.

Public void addGrade{

Grade = get grade to add

Add grade to system

}

1. Delete Grade

This will remove a grade from the system.

Public void deleteGrade{

Grade = get grade to delete

Delete grade from system

}

1. Delete Current Class

This removes a current class from the system.

Public void deleteClass{

Class = get class to delete

Delete class from system

}

1. Delete Completed Class

This removes a completed class from the system.

Public void deleteCurrentClass{

Class = get current class to delete

Delete current class from system

}

# Conclusion

In conclusion this system will be helpful to any student at EKU that is currently in a Computer Science concentration by allowing them to see how grades will impact them and allow an easy way to keep track of any classes you need and what GPA these classes could give you.

# Data Dictionary

**EKU C.S. Student/EKU CS Student** – A Computer Science Student at Eastern Kentucky University.

**GPA** – The overall Grade Point Average of the student’s grades.

**What-if grade audit** – This allows the student to input a fake grade to see how that grade would impact them.

**Current Courses** – Courses the student is currently taking.

**Completed Courses** – Courses the student has completed.

**Concentration** – The major or set of classes the student is enrolled in.