**COIT20258-Software Engineering**

**Assessment 1**

Grades Management System Phase 1: Design Document

Version 1.0

Student Software

August 2020

Robustness Analysis

Student Name: Bishal Budhathoki

Student ID: 12116421

Lecturer: Dennis Jarvis

Tutor: Dennis Jarvis

Unit Coordinator: Dennis Jarvis

CQUniversity, Sydney

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 9/2 | 02/09/2020 | Bishal Budhathoki | Complete phase 1 documentation with appendix |

Table of Contents

[Revision History 2](#_Toc49946205)

[1. Introduction: 4](#_Toc49946206)

[2. Requirements: 5](#_Toc49946207)

[3. Architecture: 6](#_Toc49946208)

[4. Database / Data Access Design: 7](#_Toc49946209)

[5. GUI Design 8](#_Toc49946210)

[6. Class Diagram 10](#_Toc49946211)

[7. Sequence Diagrams: 11](#_Toc49946212)

[Appendix 14](#_Toc49946213)

[Appendix A: Use Case Diagram 14](#_Toc49946214)

[Appendix B: Communication Diagram: 18](#_Toc49946215)

[Appendix C: Class Diagram: 19](#_Toc49946216)

# Introduction:

Student Software has taken the project to build a Grade Management System for the unit coordinators for the allocation of grades of every individual student. This desktop application should be developed the latest technologies like NetBeans, Java, Java Database, Swing, JavaFX can be applied. Here in this document Robust analysis of the system is also performed. This report document visualises the present state of the design for the stage 1 system.

# Requirements:

Use cases are not duplicated due to the simpler functional requirements which is same case for Student design document.

The aim of the desktop system is to help unit coordinators to compute and allocate the grades of every individual student. Three phase development process is implemented where calculation of grades and displaying on text area are then deferred to upcoming phase. First two phases is more oriented on calculating grades, updating an entry and displaying it.

Using Java Swing or JavaFX a desktop application is to be developed whereas the communication between the software and the database is handled by JDBC. Here the IDE used will be IntelliJ Idea for the development and Java DB will be used for the database.

Early functional requirements are:

1. Start the software and make connection with the database
2. Do verification using username and password to use application
3. Display home GUI that is attached in figure 2 that accepts studentID, Assessment and exam marks.
4. Display the required result / grade in text area.
5. Update the DB as required.
6. Alert the warnings and notices.
7. Exit the application and terminate the database connection.

The application also must follow MVP architecture. Database design along with sample data are available in section 4. There are no data validation during this stage, but some criteria are to be fulfilled and if not satisfied message required are to be displayed. These criteria are available in Appendix A.

# Architecture:

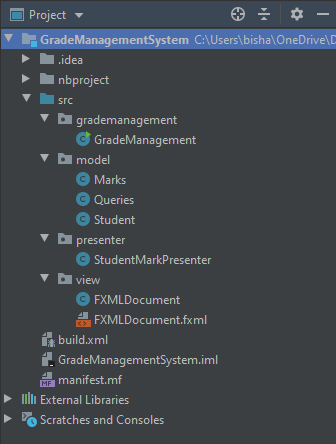
MVP architecture demands 3-layered architecture for the system software and those layers are demonstrated as Packages. The package structure for the system software is displayed in figure 1.

Figure 1. Package Structure

# Database / Data Access Design:

SQL script for creating the database for GRADES table is as follow:

DROP TABLE MARKS;

CREATE TABLE MARKS

(

STUDENTID VARCHAR (8) NOT NULL,

ASSIGNMENT1 INT NOT NULL,

ASSIGNMENT2 INT NOT NULL,

EXAM INT NOT NULL,

TOTAL INT NOT NULL,

GRADE VARCHAR (4) NOT NULL,

PRIMARY KEY (STUDENTID)

);

INSERT INTO MARKS

(STUDENTID, ASSIGNMENT1, ASSIGNMENT2, EXAM, TOTAL, GRADE)

VALUES

('S01', 20, 0, 25, 45, '?'),

('S02' ,0, 0, 0, 0, '?'),

('S03', 15, 0, 0, 15, '?');

Due to only one table, ERD for this is not given.

Data access is done using JDBC and prepared statements. Database here to use must be Java DB.

In software design document the queries are specified and then connect them with the methods provided in GradesQueries. Queries formulation is also a work for implementers for this project. Grades valuation is done only 3 entries including 2 assessment and 1 exam are provided and if not provided will be assigned 0 by default.

# GUI Design

Below is the purposed GUI for the system and developers can edit and make changes to it using NetBeans / IntelliJ Idea GUI Builder or any other relative tools. Developers can also use their own separate layout. Due to the simplicity in the requirements, only this screenshot of GUI is provided which are needed for Student Design document. To display the result one can, use Grade.toString() to have own formatted result of each student.

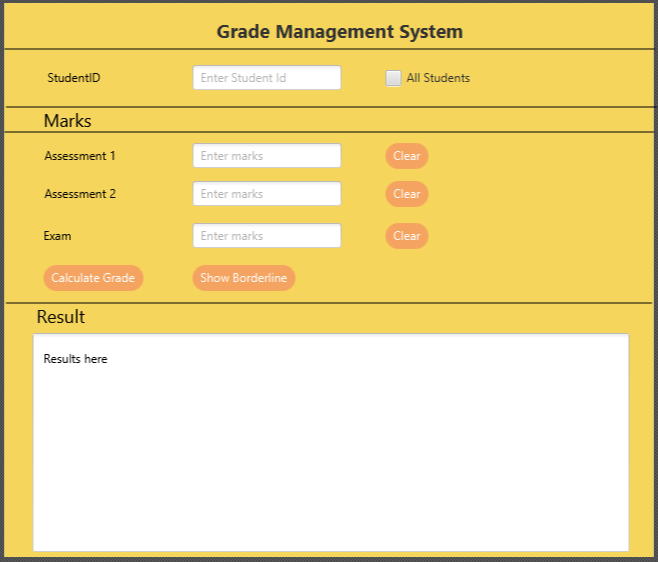


Figure 2: Purposed GUI

|  |  |
| --- | --- |
| **Functionality** | **JavaFX Components** |
| Output | TextArea |
| Input | TextField, CheckBox |
| Operations | Button |

**Table 1.** Mapping of GUI functionality to Swing component types.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Widgets** | **Inputs Required** |
| 1 | n/a | n/a |
| 2 | Exit (Button) | None |
| 3 | StudentID (TextFIeld) | StudentID |
| 4 | All Students (CheckBox) | n/a |
| 5 | Assessment1 (TextFIeld) | Assessment 1 Marks |
| 6 | Assessment 2 (TextFIeld) | Assessment 2 Marks |
| 7 | Assessment 3 (TextFIeld) | Assessment 3 Marks |
| 8 | Clear (Button) | n/a |
| 9 | Calculate Grade (Button) | n/a |
| 10 | Show Borderline (Button) | n/a |

**Table 2.** Mapping of requirements to actions

For requirement 4, selecting the checkbox allows the calculate grades for all students. Requirement 8 is for three separate buttons namely Clear that Clears input of their individual text field. The input in the text field does not accept negative marks and treat it as an error. Validation is done by GradesView class. Show Borderline class button check for range of values and a set of validation condition and if satisfies then it will be graded as Passed with 50% grade.

# Class Diagram

For Grade Management System application, below is the class diagram:

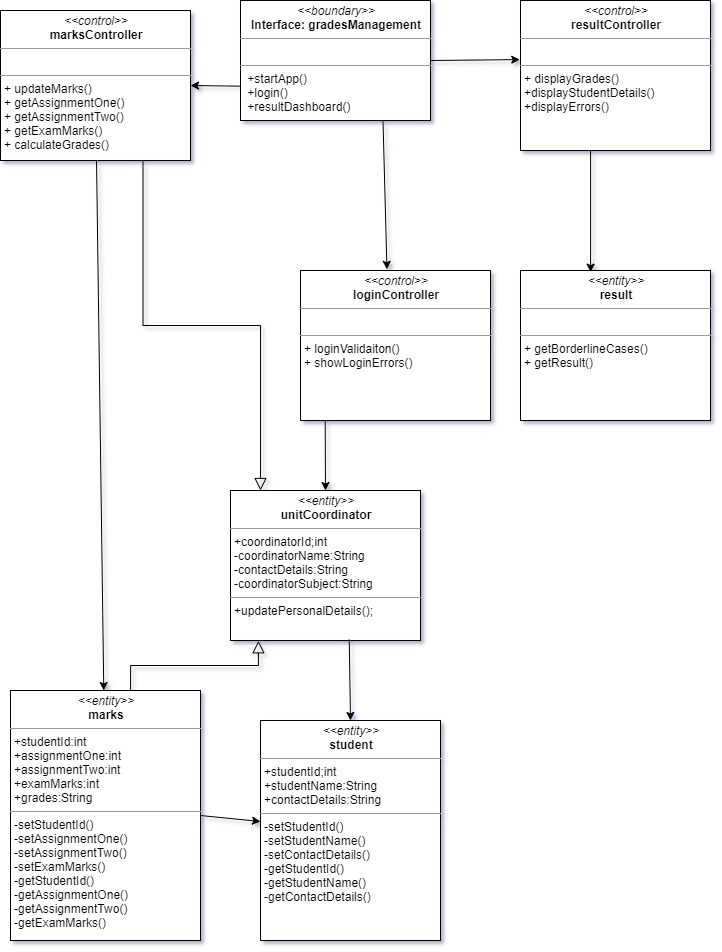


Figure 3: Class Diagram of Grade Management System

Note: Please zoom the image for in detail view of the class diagram.

Above is the refactored class structure in accordance with the NetBeans / IntelliJ Idea GUI builder and here the is created is an individual main class which is automatically moved to the Grades class. Here the interaction between the JDBC and GradesQueries are not displayed which is OK for class diagram. Also, formal UML syntax is also not followed here. Every individual classes and the method are also properly mentioned above.

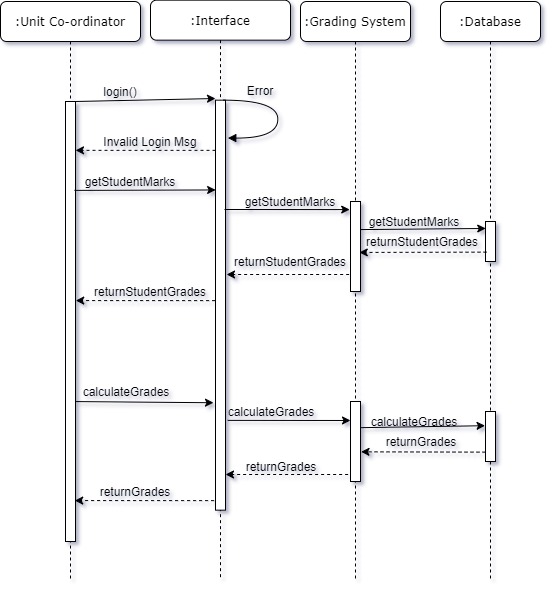
# Sequence Diagrams:

There are objects in the system that needs to communicate properly in order to do the task. Sequence diagram describes how this object communicate sequentially. Below sequence diagram is a basic diagram. Sequence diagram is separated according to the given 3 use cases.

Following are the sequence diagram according to the given use cases:

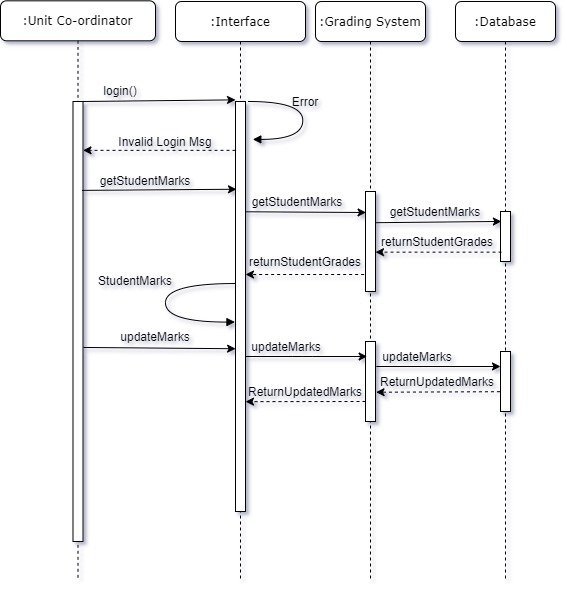
1. Calculate Grades for an individual student or all student and display the result

After Unit co-ordinator login in the system successfully interacting with the interface, s/he either gives student ID of the specific student or select the check box for all student and click Calculate Grade button. Then the grades for individual or all student is calculated done by the grading system which is collected from the database and then displayed in the given text area in the interface.



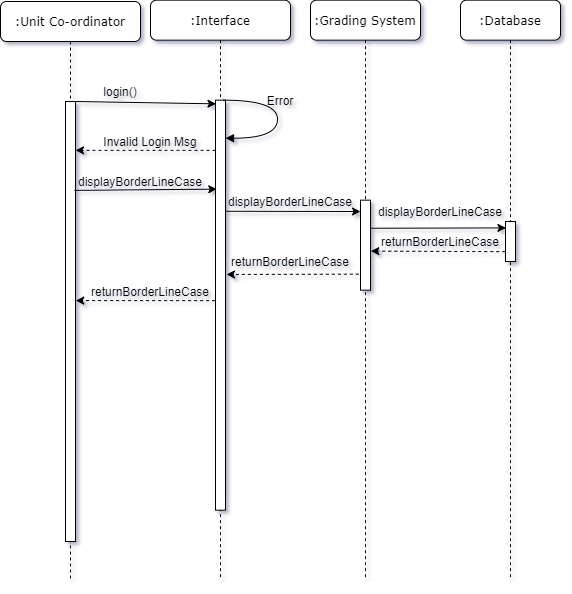
1. Update an entry for a student for one or more assessment and display the updated record

Here after successful login, Student marks are updated for assessment(one or more) in the interface and then those marks are taken to the database by the Grading system and when the updating is completed in the DB then Grading System takes the updated marks from the DB and calculate the grade and then that output is displayed in the text area.



1. Find borderline cases and display in text area

In this sequence diagram, after unit co-ordinator did the successful login, when clicks the Show Borderline Button, Grading system request the required data from the database to find the boundaries cases and then applies tolerance of 2 rules on those boundaries and then it is updated in DB which is already occurred in case a. Then those borderline case are returned to interface and then obtained by co-ordinator.



# Appendix

# Appendix A: Use Case Diagram

Here, the actors in the diagram interact with the system depending on multiple use cases scenario.

**Use Case 1: Calculate Grades for individual or all student and display in the text area**

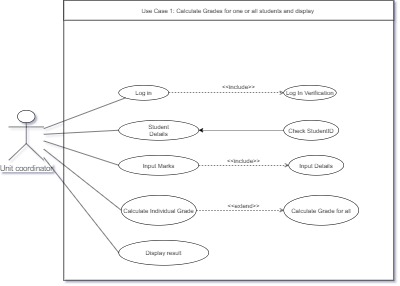
Unit Co-ordinators is the actor as the grades are provided only by unit co-ordinators and the person to interact with the system application is Unit Co-Ordinator. And for this below is the use case diagram:

Figure: Use case of Students Grades

**Use Case 2: Update an entry for a student and display the updated record in text area**

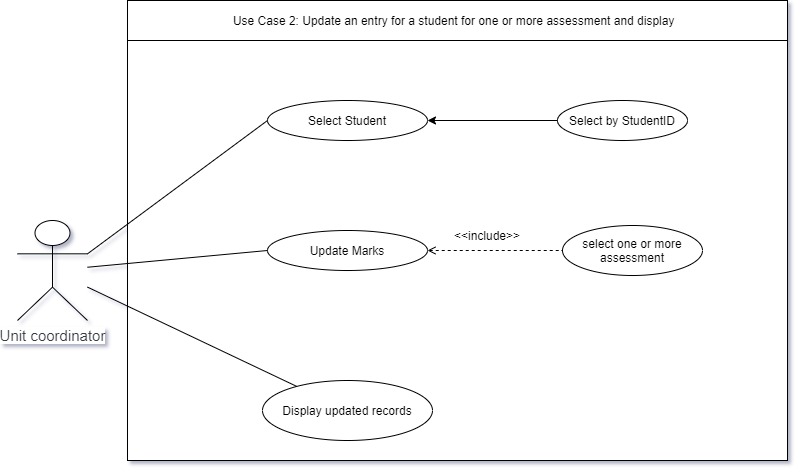


Figure: Update an entry for a student

**Use Case 3: Find Borderline Cases**

Here student marks are obtained then boundaries range is selected and then tolerance rule is applied and then it is updated in the database and then displayed.

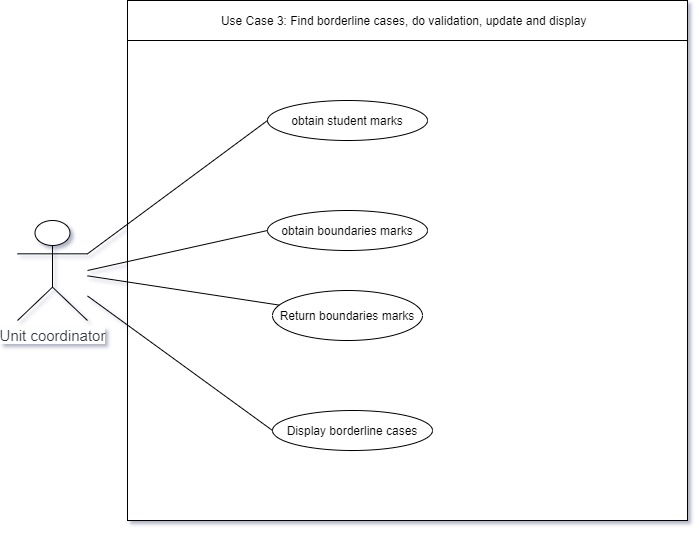


Figure: For border line cases

Use cases can also be presented textually known as Use Case Specification. Use Case Specification for above use cases are presented as follow:

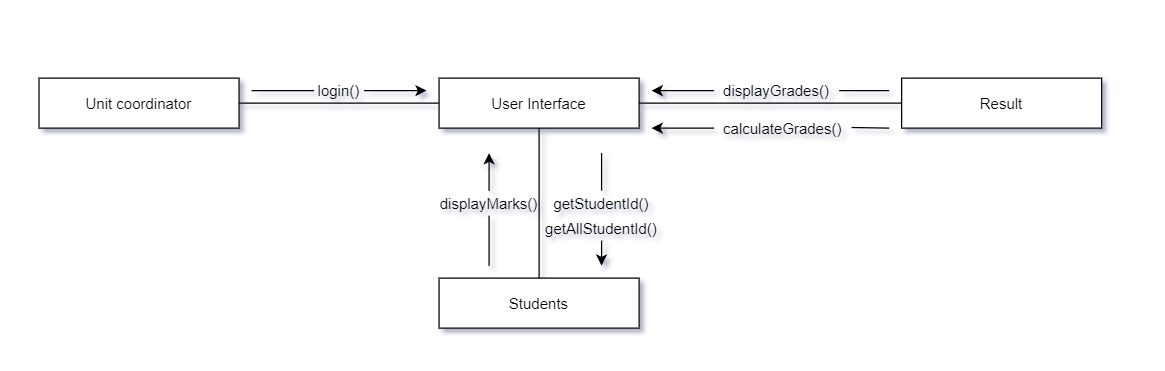
|  |  |
| --- | --- |
| Title | Grades Management System |
| Description | The booking system helps co-ordinator to insert grades for every individual student, calculate grades and display the grades. |
| Primary Actor | Unit Co-ordinator |
| Preconditions | * The connection to database is established. * The system has no flaws and works perfectly. |
| Post Conditions | * Grades are calculated. * Entries are updated * Result / changes displayed in text area * Confirmation on result / changes. |
| Cases | Case1: Calculate grades and display in text area  Case2: Update an entry for a student or all student and display updated record in text area  Case3: Find border line cases and display in text area the border line cases |
| Success scenario | * Case 1: The text is filled with calculated grades and displayed. * Case 2: An entry for a student is made and update is displayed in text area. * Case 3: Analyse border line cases and make changes accordingly and display in text area |
| Extensions | Case 1:   * Display alert for not found marks in database. * Display alert if negative marks are given and above max marks is given.   Case 2:   * Display alert if student ID is not found. * Alert if no update marks is provided for student and use the previous marks.   Case 3:   * Find borderline case and apply tolerance of 2 rule * Display alert message if no borderline case present |
| Frequency of use | The Grading system is used once after every submission due date is over and exam is over and also when student applies for revaluation of the marks. |
| Special Requirements | All students’ marks are to be filled initially along with student detail. |

Figure: Use case Specification description

# Appendix B: Communication Diagram:

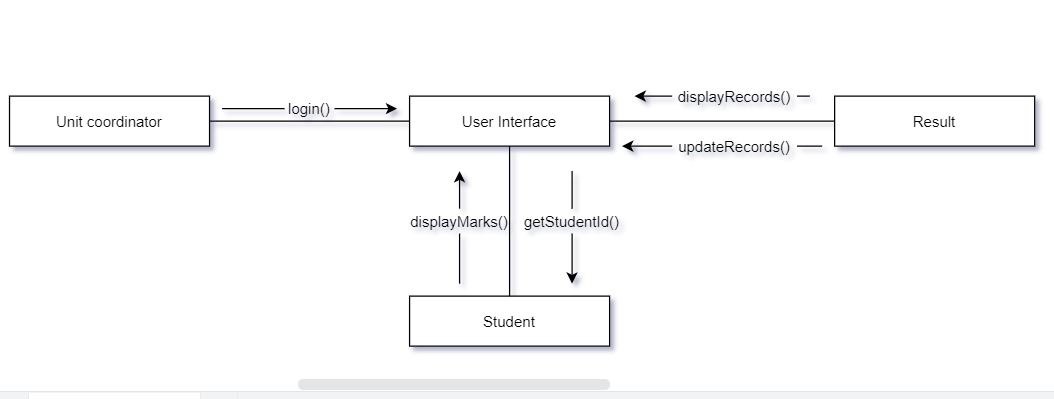
It represents exchange of data between objects. For different use cases, 3 different communication diagrams are as follow:

1. Calculate grades for individual or all student and display grades

Figure: Communication Diagram for Use Case 1

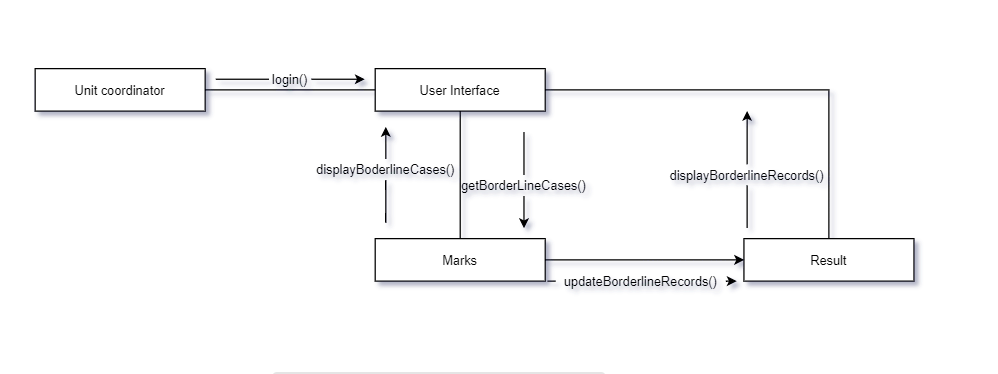
After successful login in the system using interface, unit co-ordinator request for Grades for one or all students and students entity provide marks and Result after calculation of grades display the grades in the interface.

1. Update an entry for a student and display updated record

Figure: Communication diagram for Use Case 2

When an entry for a student needs to be done, after entering studentID, Unit co-ordinator provides marks required and then the updated records is displayed from Result in the interface. This is for individual student so for more process, it can be tedious to perform.

1. Find borderline case and apply validation and update rule and display the borderline cases.

Figure: Communication Diagram for Use Case 3

After login, Unit Co-ordinator selects Show borderline button which then request for getBorderLineCases() method and after applying the methods it then update Result. Result then display the updated borderline cases with the updated marks and grades.

# Appendix C: Class Diagram:

Class diagram according to 3 different use cases are as follow:

Display grades of individual or all student after calculating grades

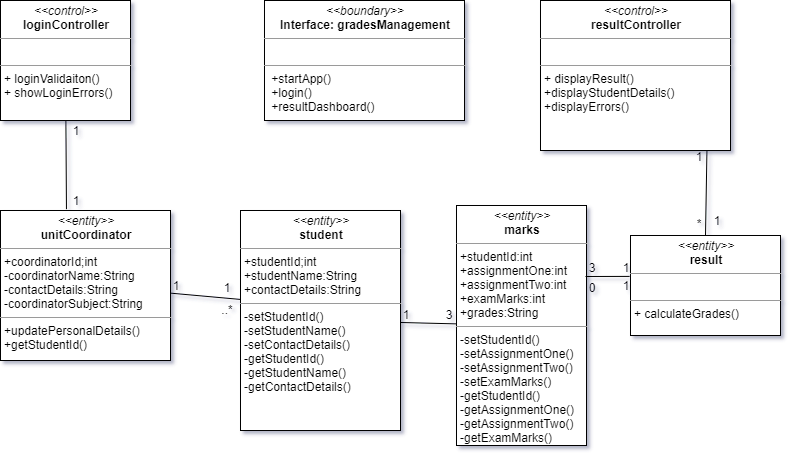


Figure: Class diagram to display grades after grade calculation

Display records after successfully updating marks of one or more assessment of individual student

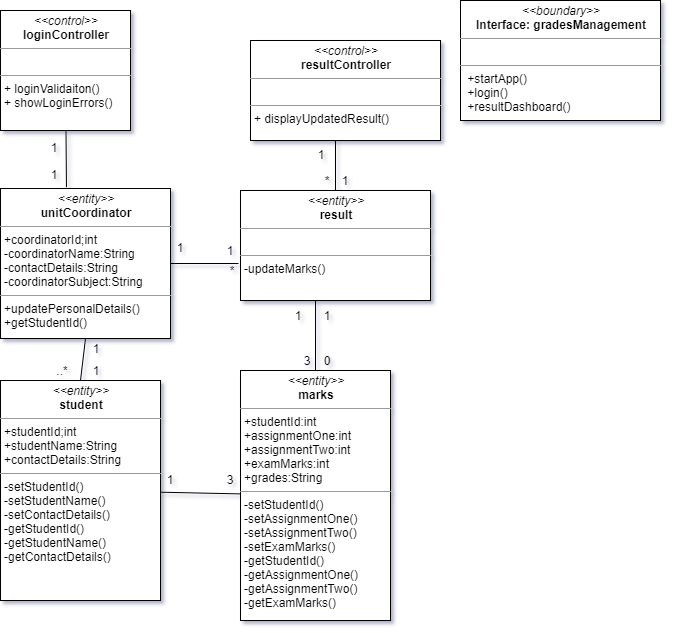


Figure: Class Diagram for individual student marks update

Display borderline cases after validation and update using tolerance of 2 rule

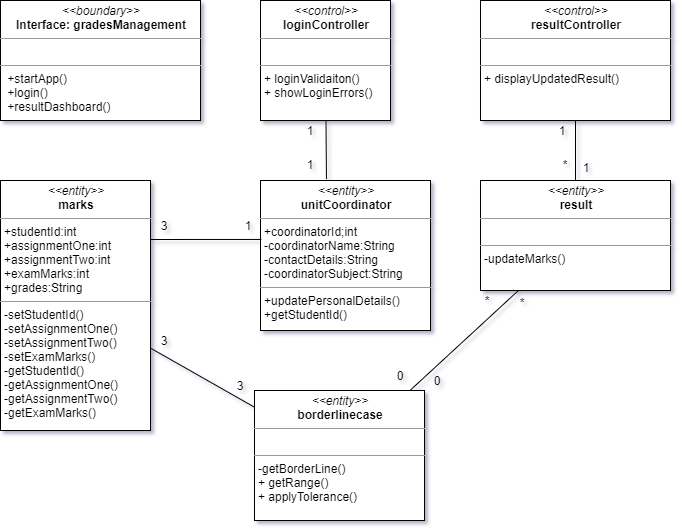


Figure: Class Diagram For borderline cases

From above 3 cases of class diagram for Grading Management System, below is the final class diagram:

**Final Class Diagram**

From above three class diagram below final class diagram is made with consideration of every connection and entities, class along with multiplicity in mind. Below is not an in-depth class diagram. For more in-depth class diagram and more detail it is present in Section 6 of design document. Below is class diagram according to the use cases.

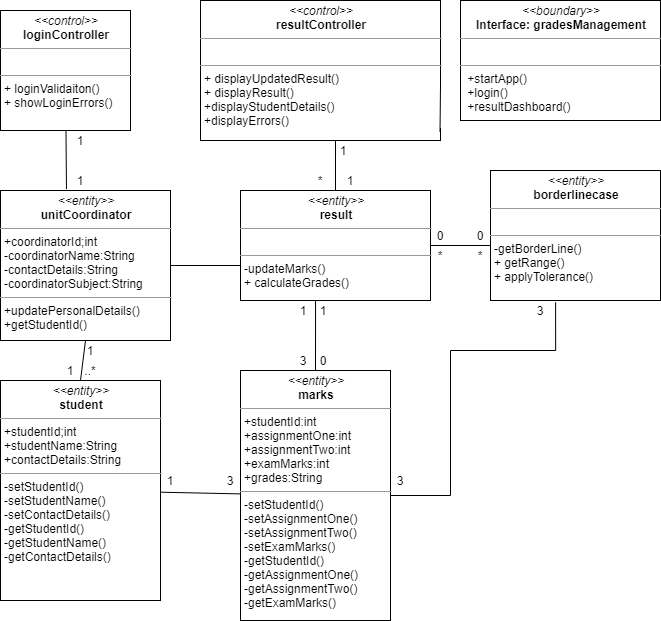


Figure: Class Diagram for Grades Management System

(Class Diagram is displayed in-depth in design document.)

Thank you