**A blue and white logo

Description automatically generated**

CSC 463 – Advanced Software Development

Final Project Documentation

Marven Younes

&  
Andrew Fakhry

ID: 20211528

ID: 20212423

Table of Contents

[Student Grades Management System (GMS) 2](#_Toc185443953)

[1. Overview 3](#_Toc185443954)

[2. Development Process and Prototyping 3](#_Toc185443955)

[3. User Guide 4](#_Toc185443956)

[Input Fields 4](#_Toc185443957)

[Action Buttons 5](#_Toc185443958)

[Menu Options 5](#_Toc185443959)

[4. Implementation Details 5](#_Toc185443960)

[5. Validation 5](#_Toc185443961)

[6. Advanced Usage 5](#_Toc185443962)

[7. User Interface Showcase 5](#_Toc185443963)

# Student Grades Management System (GMS)

**This document provides comprehensive instructions, technical details, and step-by-step guidance for using the GMS application.**

## Overview

The Student Grades Management System (GMS) is a Windows Forms application developed using C# and .NET Framework. It enables instructors to manage student data, calculate statistics, and apply grade adjustments. This tool is essential for academic institutions, simplifying grade management and providing clear insights into student performance.

## Development Process and Prototyping

The development of the **Grade Management System (GMS)** was carried out using a structured and iterative approach to ensure a robust, scalable, and user-friendly application. The process emphasized rapid prototyping, careful planning, and continuous testing to meet the project’s objectives.

**1. Planning and Requirements Analysis**

The first phase involved identifying the primary objectives and user needs. The key goal of the GMS is to manage and analyze student grades, automate grade adjustments, and generate class statistics. The system was designed to cater to instructors and administrators who require efficient, reliable methods of handling student data, performing calculations, and generating reports.

Key functionalities included:

* Grade entry and management
* Calculation of class statistics (average, highest, lowest grades)
* Grade adjustment functionality
* Validation of data input
* Error handling for common input mistakes

**2. Prototyping**

In this phase, a low-fidelity **prototype** was developed to visualize the application’s layout and interactions. The prototype was used to demonstrate the core features, such as the grade entry forms, statistical displays, and the user interface for generating reports.

Key steps included:

* **Wireframing**: Initial sketches and wireframes were created to map out the structure of the application, focusing on the user interface elements and the flow of interactions.
* **Iterative Refinements**: the prototype underwent several iterations to fine-tune the UI elements, layout, and functionality.

**3. Development**

From the prototype phase, the development of the full application began. Key considerations during this phase included:

* **Choice of Technology**: Visual Studio was used (as required by our instructor) for the application development, with C# as the programming language. The application was built to be both flexible and scalable, allowing easy modifications and updates.
* **Database Integration**: A simple CSV-based data storage system was implemented to store and manage student grade data. The application’s backend logic was designed to ensure efficient data retrieval and processing.

**4. Testing and Iteration**

After initial development, the system underwent comprehensive testing to ensure accuracy in grade calculations, data storage, and error handling. Several rounds of testing were conducted:

* **Unit Testing**: Individual components and functions were tested to ensure they worked as expected.
* **Bug Fixes and Refinements**: Based on multiple tests, bugs were identified and resolved, and improvements were made to enhance the overall system performance and user experience.

**5. Deployment and Finalization**

After testing and final refinements, the system was ready for use. User manuals and documentation were provided to assist users in navigating the system. The final application was ready for implementation, ensuring that it met the specified requirements for grade management and analysis.

## User Guide

### Input Fields

- \*\*Name\*\*: Enter the full name of the student. This field accepts text input.  
- \*\*Semester\*\*: Select the semester of enrollment from a predefined dropdown list.  
- \*\*Grade\*\*: Enter the student's grade as a numeric value between 0 and 100.

### Action Buttons

- \*\*Save Data\*\*: Saves the current entries into a CSV file for persistence.  
- \*\*Load Data\*\*: Loads student records from an existing CSV file for analysis.  
- \*\*Calculate Stats\*\*: Displays statistical metrics like average, maximum, and minimum grades.  
- \*\*Apply Raise\*\*: Applies a user-defined adjustment to all grades, ensuring grades do not exceed 100.  
- \*\*Pass/Fail Count\*\*: Calculates and displays the number of students passing and failing.

### Menu Options

- \*\*File > New File\*\*: Clears all data to start a new session.  
- \*\*File > Open File\*\*: Opens a dialog to select and load an existing CSV file.  
- \*\*File > Save File\*\*: Saves the current data into a CSV file.

## Implementation Details

The GMS application is built using Windows Forms in C# with the following key components:  
- \*\*Data Storage\*\*: In-memory DataTable for managing student records.  
- \*\*CSV Operations\*\*: Handles file input/output for saving and loading data.  
- \*\*Statistics Calculation\*\*: Leverages LINQ for efficient data analysis.

## Validation

Validation ensures data integrity and prevents errors. Specific checks include:  
- Grades must be numeric and within the range of 0 to 100.  
- Adjustment values must be numeric and applied only when valid.

## Advanced Usage

The application supports advanced features such as:  
- Maintaining an adjusted grade column for comparison with original grades.

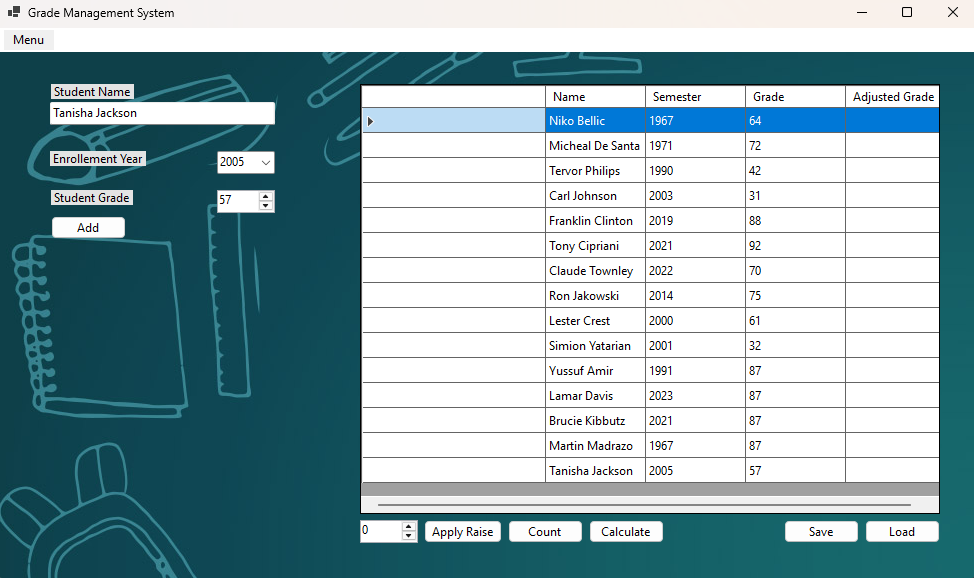
## User Interface Showcase

Main GUI Design (not compiled)

A screenshot of a computer

Description automatically generated

GUI after compiling and Data implementation



Save Data as CSV Dialog

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedResults Saved in the CSV File

Applying Raise to all students

A screenshot of a computer

Description automatically generated