# Software Documentation

## for

# RelativeEase:

## A Relative Grading Software for Professors

**Prepared by Arshad Ahmed, Sanjana Gummuluru, and Harshel Swain**
**CSPE41 project, Spring 2024**
**National Institute of Technology, Tiruchirappalli**

# Table of Contents

# Problem Identification and Requirements Elicitation

## Motivation

The RelativeEase software is a relative grading software aimed at university professors. The software allows the user to assign grades with ease, satisfying their specific criteria pertaining to their individual university guidelines.

This software has been developed by team ‘Mastery’, whose members are:

1. Arshad Ahmed (106122016)
2. Harshel Swain (106121049)
3. Sanjana Gummuluru (106122106)

This software was developed as the project for the CSPE41 Software Engineering course handled by Dr. Oswald C during the Spring 2024 semester.

The major motivation for this project came after a discussion with Dr. Oswald, who explained the monotonous, time-consuming process of assigning grades to students. After speaking to a few more faculty members at NIT-Tiruchirappalli, it was evident that a software that efficiently automated this process would be of great help to the professors.

## Objective

The intended functions that the software wishes to fulfil are:

1. Rough relative grading based on user input of a .xlsx file
2. Customization based on user requirements of
    a. Grades as per university (for example, S-F scale or AA-FF scale)
    b. ‘floor’ or ‘ceil’ marks, i.e. roundoff
    c. Percentage allowed per grade as per university requirements
    d. Pushing or pulling students to adjacent grades
    e. Manipulation of marks of individual records
3. Outputs:
    a. A downloadable .xlsx file of the records of students, marks and the assigned grade computed
    b. A report of the number and percentage of students assigned each grade, along with the marks cutoff for each grade

Note: the Software Requirements Specification document can be separately found in the same folder. This also contains the UML diagrams.

## Future scope

This software has been developed over the course of 10 weeks, and currently includes not all, but most of the previously listed features. Our intention is to continue working on this to release a fully-functioning software with additional features, so it can be widely used across several universities. A few of these features include:

1. Working with universities: This will allow the software to be used easily with university login and will already contain required data for the grading system. It will also allow easy submission from professors to the administration and will be able to hold records for all courses, easing the work of both the professor as well as the office.
2. A more interactive UI: This will allow even professors with extremely minimal computer use to easily use the software.
3. Graphs showing statistics of the grades assigned.

# Process

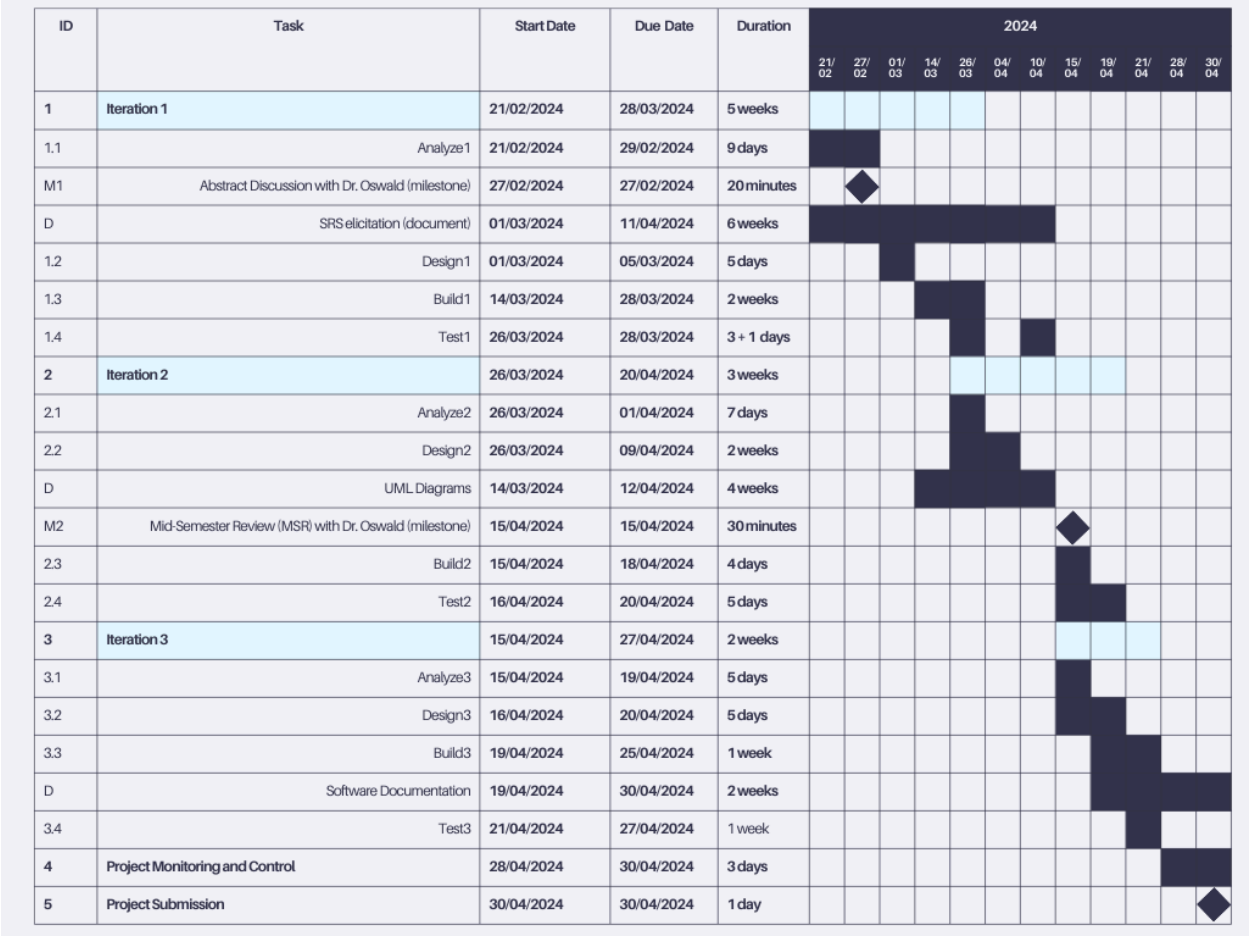
This software was developed using the Agile model. Due to the short timeframe for this project, we felt that the Agile model would best suit it. The major reason for this is the nature of our work, i.e. other university classes and responsibilities. The major working hours for this project were between 18:00-21:00 and 23:00-01:00. Thus, it seemed most appropriate to work in iterations of sprints of 2 weeks, with weekend meetings where planning and next-stage design took place. This also allowed us to work around sudden schedule changes such as CTs/exams, immediate assignment deadlines, extra classes or labs, and other club/council/project meetings.

## Applications of Agile methods

(methods stated from slides created by Dr. Oswald C.)

1. Customer Involvement: It was ensured that at every step of the way, we communicated with Dr. Oswald as and when decisions were made and doubts came into picture, to ensure that the base of this project was to satisfy professor requirements.
2. Incremental delivery: The software was developed in increments. Each increment included its own Analyse-Design-Build-Test cycle.
3. People not process: The development team each performed tasks to their individual strengths as well as in the way that suited their process best. This includes both technically (frontend, backend, diagrams, design, documentation) and non-technically (time of work, style of work, duration of work).
4. Embrace change: During the course of the project, i.e. after each review with the professor (client as well as end user in our case), the necessary changes were taken as top-priority consideration for next steps.
5. Maintain simplicity: As students who learnt through the course of the project, both product and process were kept simple and succinct.

# Timeline Chart



| ID | Task | Start Date | Due Date | Duration |
|---|---|---|---|---|
| 1 | **Iteration 1** | 21/02/2024 | 28/03/2024 | 5 weeks |
| 1.1 | Analyze1 | 21/02/2024 | 29/02/2024 | 9 days |
| M1 | Abstract Discussion with Dr. Oswald (milestone) | 27/02/2024 | 27/02/2024 | 20 minutes |
| D | SRS elicitation (document) | 01/03/2024 | 11/04/2024 | 6 weeks |
| 1.2 | Design1 | 01/03/2024 | 05/03/2024 | 5 days |
| 1.3 | Build1 | 14/03/2024 | 28/03/2024 | 2 weeks |
| 1.4 | Test1 | 26/03/2024 | 28/03/2024 | 3 + 1 days |
| 2 | **Iteration 2** | 26/03/2024 | 20/04/2024 | 3 weeks |
| 2.1 | Analyze2 | 26/03/2024 | 01/04/2024 | 7 days |
| 2.2 | Design2 | 26/03/2024 | 09/04/2024 | 2 weeks |
| D | UML Diagrams | 14/03/2024 | 12/04/2024 | 4 weeks |
| M2 | Mid-Semester Review (MSR) with Dr. Oswald (milestone) | 15/04/2024 | 15/04/2024 | 30 minutes |
| 2.3 | Build2 | 15/04/2024 | 18/04/2024 | 4 days |
| 2.4 | Test2 | 16/04/2024 | 20/04/2024 | 5 days |
| 3 | **Iteration 3** | 15/04/2024 | 27/04/2024 | 2 weeks |
| 3.1 | Analyze3 | 15/04/2024 | 19/04/2024 | 5 days |
| 3.2 | Design3 | 16/04/2024 | 20/04/2024 | 5 days |
| 3.3 | Build3 | 19/04/2024 | 25/04/2024 | 1 week |
| D | Software Documentation | 19/04/2024 | 30/04/2024 | 2 weeks |
| 3.4 | Test3 | 21/04/2024 | 27/04/2024 | 1 week |
| 4 | **Project Monitoring and Control** | 28/04/2024 | 30/04/2024 | 3 days |
| 5 | **Project Submission** | 30/04/2024 | 30/04/2024 | 1 day |



An incremental Agile model was followed for the process, which involved 3 iterations of the Analyze-Design-Build-Test cycle.

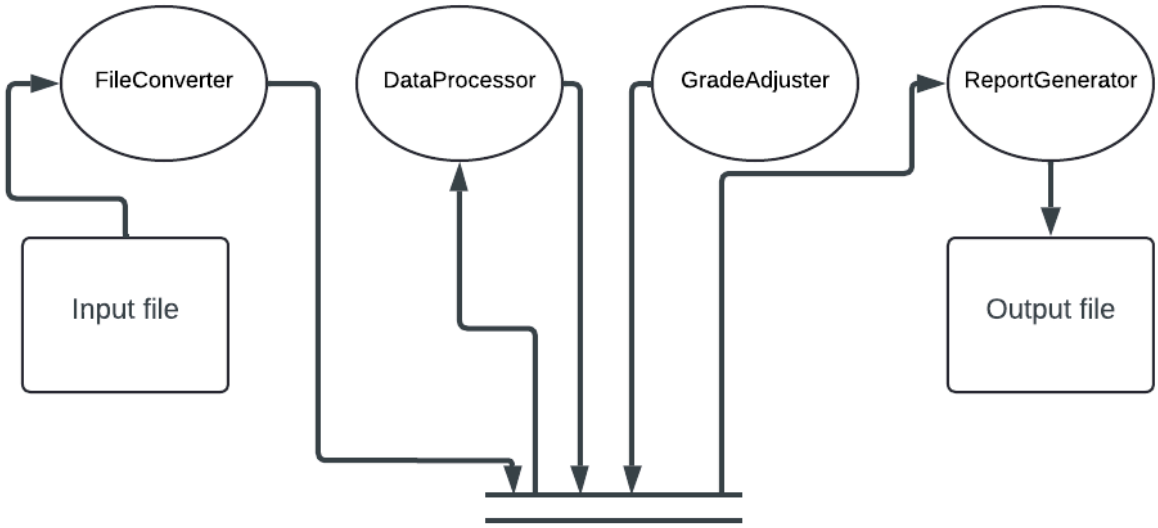
1. Basic Architectural Design with functionality as reviewed for the MSR (15/04/2024)
   Duration: 5 weeks
   Problems faced / reasons for delay: CTs/exams, other course activity

2. Fully-functioning software capable of being run in terminal
   Duration: 3 weeks
   Problems faced / reasons for delay: small break taken to prepare for MSR

3. User Interface
   Duration: 2 weeks
   Problems faced / reasons for delay: other course activity, T&P internship activity

# Software design

## Data flow diagram




## Component Design

The following principles were adhered to:

- Open - Closed Principle (OCP)
    - Modules should be open for extension but closed for modification
    - Example: the GradeAdjuster Class was added in the 2nd incremental cycle. This class is a part of the functionality of data processing. However, adhering to this principle, it was made an extension of DataProcessor so as to not modify existing functionality.

- Dependency Inversion Principle (DIP)
    - Abstractions rather than concretions
    - Example: after regular meetings as well as the mid-semester review with Dr. Oswald, changes were implemented due to the abstract nature of the code, rather than concrete code that would have had to be rewritten from scratch.

- The following principles were kept in mind, but did not come into the picture as of now. As we continue to work on this project even after the semester closure, they are of the highest importance.

    - Interface Segregation Principle (ISP): As mentioned above in the Future Scope section, this principle is the base for creating separate interfaces for different clients (here, universities).
    - Liskov Substitution Principle (LSP): Similarly, due to the primitive nature of this software, it does not currently contain as many classes for this principle to be applicable. As it grows, however, this principle forms the basis for the structure of modules to support several interfaces.

## Coupling and Cohesion

The desirable combination of high cohesion and low coupling has been adhered to. Following are the types of cohesion and coupling present in this software.

- Cohesion:
    - Functional: all elements within each module contribute to performing the single task it is defined to achieve.
    - Sequential: in a module, the output of one element serves as the input to the next.
    - Communicational: within the module, elements share common data (here, the student records database).
    - Temporal and Procedural: elements are grouped as they are executed sequentially but within a certain time period associated with that module.

- Coupling:
    - Common: modules share global data.

As evident, the presence of low coupling reduces error probability and time spent on correction, and increases readability, maintainability and modularity.

# Code

```
#pragma once
ref class InputHandler
{
public:
    int getUserInput();
};
```

```
se                                                    (Global Scope)
#pragma once
#include <string>
ref class FileConverter
{
private:
public:
    void convertCSVtoDatabase(const std::string& filename);
};
```

```
#pragma once
ref class DataProcessor
{
public:
    void calculateGrades();
};
```

```
#pragma once
ref class GradeAdjuster
{
public:
    void displayTableByGrade();
    void adjustStudentGrade(double targetMark, char targetGrade);
};
```

```
#pragma once
#include <string>
ref class ReportGenerator
{
public:
    void generateReport();
    void writeCSVFile(const std::string& filename);
};
```

## Tech Stack

- Languages:
    - C++
    - Javascript
    - HTML
    - CSS
    - SQL
- Frameworks:
    - React JS
    - Oracle MySQL
    - Visual Studio
    - C++ - SQL connector (API)
    - Creately
    - IBM Rational Rose

## Do’s and don’ts followed

- Single entry, single exit
    - As can be seen from the main function (commented as well), the control flow adheres to this rule.
    - Control flows as objects are created and functions are called, with no arbitrary nesting, infinite loops, etc.

- Exception handling
    - try-catch blocks are used at every necessary location to ensure no errors are overlooked or generate confusion for the user

- No goto statements have been used.

- Indentation and comments are as per the required standards.

- Variables, functions and classes have been named such that their use and functionality is immediately evident.

# Testing

## Test Plan

- Scope
    - Functional testing of the algorithm
    - Integration testing with the external systems and connections
    - Usability testing of the interface

- Approach
    - Functional testing: make sure that the algorithm assigns the correct grades based on the criteria specified
    - Integration testing: ensuring server connectivity, dependencies and other integrations
    - Usability testing: assessing the user interface for intuitiveness and ease of use

- Schedule
    - Testing was conducted with respect to the incremental agile model followed.
    - Round 1: 26/03/2024 - 28/03/2024, and again on 10/04/2024 in preparation for the mid-semester review. This round of testing involved functionality for the initial model and integration of the backend.
    - Round 2: 16/04/2024 - 20/04/2024. This round involved functionality and integration testing of the entire backend model. After this round, the software was fully capable of running in the terminal.
    - Round 3: 21/04/2024 - 27/04/2024. This round involved usability testing of the interface.

## Requirements Traceability Matrix

| **Serial No.** | **Requirement Description** | **Test Case Description** | **Result** |
|---|---|---|---|
| 1 | The system shall allow the user to input student data | Verify that user can input the student record file as per the given format | Pass |
| 2 | The system shall calculate grades | Verify that grades are calculated | Pass |

| | | | |
|---|---|---|---|
| | based on relative performance compared to peers | accurately using the grading algorithm | |
| 3 | The system shall provide a user friendly interface for professors | Verify that the user interface is intuitive, easy to navigate, and adheres to UI principles | Pass |
| 4 | The system shall generate reports of student grades | Verify that the software generates accurate grade reports | Pass |

## Test Scenario

1. Preconditions:
    a. The software is installed and accessible
    b. Student performance data is available

2. Inputs
    a. Student performance data **as per the stated format**
    b. Grade gaps, methods, y/n for push/pull, target marks and grades

3. Steps:
    a. Login
    b. Input student performance data
    c. Input other guidelines
    d. Verify that the system calculates grades based on the relative performance of the student as compared to their peers
    e. Verify the accuracy of push/pull
    f. Review the calculated grades for accuracy and consistency
    g. Repeat with different sets of data to ensure robustness

4. Expected outcomes:
    a. The system assigns grades accurately
    b. The grade distribution reflects the variation in student performance
    c. The calculated grades align with the user requirements

5. Postconditions:
    a. The grading data is saved and can be accessed for further review and analysis

## Test Cases and Data

### Iteration 1:

Functionalities implemented:

1. Mathematical calculation of Standard Deviation

| Sl no. | Test Case | Expected Output | Actual Output | Result |
|---|---|---|---|---|
| 1 | {10, 12, 23, 23, 16, 23, 21, 16} | 4.899 | 4.899 | Pass |
| 2 | {5,6,78,91,101,17,26,11,10,15} | 36.164 | 36.164 | Pass |
| 3 | {4,4,4,4,4,4,4,4} | 0 | 0 | Pass |
| 4 | {0,-1,9,4,17,2} | Please enter positive integers | Please enter positive integers | Pass |

2. Statistical assignment of grades

| Sl no. | Test Case | Expected Output | Actual Output | Result |
|---|---|---|---|---|
| 1 | {10, 12, 23, 23, 16, 23, 21, 16} | {E, D, A, A, B, A, A, B} | {E, D, A, A, B, A, A, B} | Pass |
| 2 | {5,6,78,91,101,17,26,11,10,15} | {F, F, A, S, S, F, E, F, F, F} | {F, F, A, S, S, F, E, F, F, F} | Pass |
| 3 | {4,4,4,4,4,4,4,4} | {S,S,S,S,S,S,S,S} | {S,S,S,S,S,S,S,S} | Pass |
| 4 | {0,-1,9,4,17,2} | Please enter positive integers | Please enter positive integers | Pass |

## Iteration 2:

Functionality implemented: Pushing and pulling grades

This functionality enables the user to push and pull several records above/below a certain grade to an adjacent one.

| Sl no. | Test Case | Expected Output | Actual Output | Result |
|---|---|---|---|---|
| 1 | {(12,A),(11,B),(5,D)}, 11, A | {(12,A),(11,A),(5,D)} | {(12,A),(11,A),(5,D)} | Pass |
| 2 | InputFile.csv (in the zip file) | OutputFile.csv (in the zip file) | OutputFile.csv (in the zip file) | Pass |
| 3 | InputFile2.csv (in the zip file) | OutputFile2.csv (in the zip file) | OutputFile2.csv (in the zip file) | Pass |
| 4 | {(20,A),(19,B),....,(2,F)}, 25, S | {(20,A),(19,B),....,(2,F)} (no change) | {(20,A),(19,B),....,(2,F)} (no change) | Pass |

## Iteration 3:

Functionality implemented: Front-end user interface

The User Interface was designed by allowing the prototyped interface to be looked at by 4 of our classmates, first without any explanation or guidance, and the second time after the usage was demonstrated.

The lesser the difference in time taken to explain the runthrough from start to finish between these 2 tries, the more usable the software.

The benchmark we tried for was a maximum of a 45 second difference.

| Sl no. | Test Case | Expected Output | Actual Output | Result |
|---|---|---|---|---|
| 1 | 106122017, Arushi Chauhan | < 45 | 23 | Pass |
| 2 | 106122129, Thomas Mampilli | < 45 | 41 | Pass |
| 3 | 106122105, Samrddhi | < 45 | 33 | Pass |

| | | | | |
|---|---|---|---|---|
| | Srinivasan | | | |
| 4 | 106122109, Saranya Shree M S | < 45 | 37 | Pass |

## Cover Sheet

Iteration 1
Functionality: Standard deviation, statistical grade assignment
Programmer: Sanjana Gummuluru

| Section | Contents | Due date | Completion date | Review date |
|---|---|---|---|---|
| 1 | Rqmts. | 29/02/2024 | 29/02/2024 | 04/03/2024 |
| 2 | Arch. design | 05/03/2024 | 05/03/2024 | 07/03/2024 |
| 3 | Detail design | 05/03/2024 | 05/03/2024 | 07/03/2024 |
| 4 | Test plan | 14/03/2024 | 14/03/2024 | 20/03/2024 |
| 5 | Source code | 28/03/2024 | 28/03/2024 | 30/03/2024 |
| 6 | Test Results | 30/03/2024 | 30/03/2024 | 30/03/2024 |
| 7 | Change Requests | - | - | - |
| 8 | Notes | - | - | - |

Release date: 30/03/2024

Iteration 2
Functionality: Pushing and pulling grades
Programmer: Sanjana Gummuluru

| Section | Contents | Due date | Completion date | Review date |
|---|---|---|---|---|
| 1 | Rqmts. | 01/04/2024 | 01/04/2024 | 03/04/2024 |
| 2 | Arch. design | 08/04/2024 | 09/04/2024 | 11/04/2024 |

| | | | | |
|---|---|---|---|---|
| 3 | Detail design | 09/04/2024 | 09/04/2024 | 11/04/2024 |
| 4 | Test plan | 10/04/2024 | 13/04/2024 | 15/04/2024 |
| 5 | Source code | 17/04/2024 | 18/04/2024 | 20/04/2024 |
| 6 | Test Results | 20/04/2024 | 20/04/2024 | 20/04/2024 |
| 7 | Change Requests | - | - | - |
| 8 | Notes | - | - | - |

Release date: 20/04/2024

Iteration 3
Functionality: Frontend user interface
Programmer: Harshel Swain

| **Section** | **Contents** | **Due date** | **Completion date** | **Review date** |
|---|---|---|---|---|
| 1 | Rqmts. | 19/04/2024 | 22/04/2024 | 25/04/2024 |
| 2 | Arch. design | 20/04/2024 | 27/04/2024 | 27/04/2024 |
| 3 | Detail design | 20/04/2024 | 28/04/2024 | 29/04/2024 |
| 4 | Test plan | 22/04/2024 | 28/04/2024 | 28/04/2024 |
| 5 | Source code | 25/04/2024 | 01/05/2024 | 02/05/2024 |
| 6 | Test Results | 27/04/2024 | 02/05/2024 | 02/05/2024 |
| 7 | Change Requests | - | - | - |
| 8 | Notes | - | - | - |

Release date: 02/05/2024