Student Record Management System

Analytical Minds Group
Daniel O. Igwe, Joseph Inaku, Osasumwen Osazuwa, Isaiah Emmanuella, Chukwuemeka Onaegbu

March 17, 2025

1 Algorithm for Student Record Project

1.1 Program Initialization

- Start the program.
- Display a welcome message and prompt the user to enter their name.
- Validate the user's name using `cleanStringInput` to ensure it contains only letters and spaces.
- Display the main menu with the following options:
  1. Display Student Records
  2. Add Student Record
  3. Modify Student Record
  4. Delete Student Record
  5. Search for a Student
  6. Calculate Average Mark
  7. Sort Student Record
  8. Save Record To File
  9. Load Record From File
  10. Exit

1.2 Main Menu Loop

- Prompt the user to enter their choice.
- Validate the user's choice using `cleanNumericInput` to ensure it is a valid integer between 1 and 10.
- Switch based on the user's choice:
  - Case 1: Display Student Records

  - Case 2: Add Student Record
  - Case 3: Modify Student Record
  - Case 4: Delete Student Record
  - Case 5: Search for a Student
  - Case 6: Calculate Average Mark
  - Case 7: Sort Student Record
  - Case 8: Save Record To File
  - Case 9: Load Record From File
  - Case 10: Exit Program
- Repeat the main menu loop until the user chooses to exit.

1.3 Functionality Breakdown

- **Display Student Records:** Checks if records exist; if not, notifies the user.
- **Add Student Record:** Validates input and ensures unique roll numbers.
- **Modify Student Record:** Allows changes to student details based on roll number.
- **Delete Student Record:** Removes a student and shifts remaining records.
- **Search for a Student:** Uses BST-based search.
- **Calculate Average Mark:** Computes and displays averages.
- **Sort Student Record:** Uses quicksort based on average scores.
- **Save/Load Record:** Handles file I/O for student data.
- **Exit:** Frees allocated memory and terminates the program.

1.4 Key Functions

- **inputStudentData:** Handles input and validation for student details.
- **isRollNumberUnique:** Checks if a roll number is unique.
- **cleanStringInput:** Validates string input.
- **cleanNumericInput:** Validates numeric input.
- **quickSort:** Sorts students by average score.
- **searchBST:** Searches for a student in the BST.
- **freeBST:** Frees memory allocated for the BST.

1.5 Pseudocode for Key Functions

inputStudentData Function

```
function inputStudentData(student, students, count, currentRollNumber):
    do:
        prompt for student name
        validate name using cleanStringInput
    while name is invalid

    do:
        prompt for roll number
        validate roll number using cleanNumericInput
        check if roll number is unique using isRollNumberUnique
    while roll number is invalid or not unique

    do:
        prompt for number of scores
        validate number of scores using cleanNumericInput
    while number of scores is invalid

    allocate memory for scores
    for each score:
        do:
            prompt for score
            validate score using cleanNumericInput
        while score is invalid
```

isRollNumberUnique Function

```
function isRollNumberUnique(rollNumber, students, count, currentRollNumber):
    for each student in students:
        if student.roll_number == rollNumber and rollNumber != currentRollNumber:
            return false
    if file "students.txt" exists:
        read file line by line
        if line contains "Roll Number: ":
            extract existingRollNumber
            if existingRollNumber == rollNumber and rollNumber != currentRollNumber:
                return false
    return true
```