**Student Search & Sort System**

**Description**  
A Java-based sorting and searching system that allows users to sort students by name and score, search for students, and benchmark sorting algorithms. It reads student details from a CSV file and supports multiple sorting and searching options.

**Pre-requisites**  
Ensure a ***student.csv*** file is present in the same directory as `Main.java`

**Clone**  
https://github.com/nankam1803/LDSProject01.git

**Changing the Dataset**

If you want to change the dataset being used:

1. Add the new CSV dataset file to the project folder (same location as Main.java).
2. Modify all occurrences of "student.csv" in Main.java to match the new filename.

**Where to Change the Dataset Filename (student.csv)**

Modify the filename in the following locations inside Main.java:

| **Line Number** | **Location in Code** | **Modification Needed** |
| --- | --- | --- |
| **Inside loadCSV()** | BufferedReader br = new BufferedReader(new FileReader("student.csv")); | Change "student.csv" to your new filename |
| **Inside benchmarkSortingPerformance()** | List<Student> tempList = new ArrayList<>(students); | Ensure the dataset is loaded before benchmarking |

**CSV File Format (student.csv)**

The program reads student data from a CSV file. Ensure it follows this format:

ID, Name, Score

1, Alice, 89.5

2, Bob, 76.8

3, Charlie, 92.3

4, David, 85.0

If your dataset has a different structure, you may need to modify the loadCSV() method in Main.java to correctly parse the data.

**General Application Execution**

After running the program, you will see the following menu:

Welcome to the Student Search & Sort System!

1. Sort students by name

2. Sort students by score

3. Search for a student by name

4. Compare sorting performance

5. Exit

Enter your choice:

**Option 1: Sort Students by Name**

Choose an algorithm:

1. Selection Sort

2. Merge Sort

3. Quick Sort

* The sorted student names will be displayed.

**Option 2: Sort Students by Score**

Choose an algorithm:

1. Selection Sort

2. Merge Sort

3. Quick Sort

4. Radix Sort (Scores converted to integers)

* If **Radix Sort** is selected, scores will be **converted to nearest integers** before sorting.
* The sorted scores will be displayed.

**Option 3: Search for a Student by Name**

Choose a search method:

1. Linear Search

2. Binary Search (only works on sorted data)

* If the dataset is **not sorted** and you choose **Binary Search**, it will show:

***The dataset is not sorted. Please sort the data before trying binary search.***

* If found, the student's name and index will be displayed.

**Option 4: Benchmark Sorting Performance**

* The program will display sorting times for:

Selection Sort

Merge Sort

Quick Sort

Radix Sort (scores)

* Example Output:

Sorting Performance Benchmark (Time in ms):

---------------------------------------------------

Algorithm Time (ms)

---------------------------------------------------

Selection Sort 12

Merge Sort 3

Quick Sort 1

Radix Sort 0

**Option 5: Exit**

* Exits the program.

**Class Explanations**

**Main.java - Application Controller**

This is the **main entry point** of the program. It:

* Loads student data from student.csv
* Displays the **user menu**
* Handles **sorting, searching, and benchmarking** user choices
* Ensures **invalid inputs are handled properly**

**Key Methods:**

| **Method** | **Description** |
| --- | --- |
| main() | Starts the application and shows the menu |
| loadCSV(String filename) | Loads students from the CSV file |
| sortByName(Scanner scanner) | Sorts students by name using user-selected algorithm |
| sortByScore(Scanner scanner) | Sorts students by score (Radix Sort option available) |
| searchStudent(Scanner scanner) | Searches for a student using Linear or Binary Search |
| benchmarkSortingPerformance() | Compares sorting times for different algorithms |

**SortAlgorithms.java - Sorting Implementations**

This class contains **sorting algorithms** for **names and scores**.

**Sorting Algorithms Included:**

| **Algorithm** | **Description** |
| --- | --- |
| **Selection Sort** | Finds the smallest element and swaps it |
| **Merge Sort** | Recursively divides the array and merges sorted parts |
| **Quick Sort** | Picks a pivot and partitions the array |
| **Radix Sort** | Used **only for scores** by converting doubles to integers |

**SearchAlgorithms.java - Searching Implementations**

This class provides **searching algorithms**.

**Searching Methods:**

| **Method** | **Description** |
| --- | --- |
| linearSearch(String[] arr, String key) | Searches each element sequentially |
| binarySearch(String[] arr, String key, int low, int high) | Searches in a sorted array using divide-and-conquer |

**Binary Search Requirements:**

* The dataset **must be sorted** before using Binary Search.
* If it is not sorted, the program will display:

***The dataset is not sorted. Please sort the data before trying binary search.***

**Student.java - Student Data Model**

This is a **simple class that represents a Student** with:

* String name
* double score
* The toString() method is overridden to **display student details properly**.

**Troubleshooting**

**Error: "student.csv not found"**

Ensure student.csv is in the same directory as Main.java.

**Error: "Invalid input! Please enter a valid number."**

Enter **only numbers** when selecting menu options.

**Binary Search Not Working**

**Sort the dataset first** before using **Binary Search**.