A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Verónica Sánchez Rendón |
| *Student Number* | 2024358 |
| *Module Title* | Higher Diploma in Science in Computing |
| *Assessment Title* | System Modelling & Build |
| *Lecturer/Supervisor* | Muhammad Iqbal |
| *Assessment Due Date* | Saturday 10th, May 2025 |
| *Date of Submission* | Week of 7th April 2025 |

**Use of AI Tools**

I have not used any AI tools or technologies in the preparation of this assessment.

**Declaration**

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

**Contents**

[Justification for Sorting Algorithm 3](#_Toc197597081)

[Justification for Searching Algorithm 3](#_Toc197597082)

[GitHub Link 4](#_Toc197597083)

## Justification for Sorting Algorithm

For sorting the list of names from dummydata.csv alphabetically, using “merge sort” is the most appropriate recursive sorting algorithm. Merge Sort is a stable algorithm, this means that if two applicants have the same name, their original order will be preserved in the sorted list. Unlike Quick Sort, Merge Sort always runs in O(n log n) time, that is why is essential for this project since it deals with randomly ordered real-world form data. It also handles large data sets more predictably divides the data before sorting, reducing the memory overhead and avoiding deep recursion on sorted input. Merge Sort is easy to implement in Java and keeps the logic clean and maintainable, especially when working with lists or arrays of String, which is being used in this project. Compared to recursive Quick Sort, Merge Sort uses balanced recursion, which makes it less likely to hit stack overflow errors when dealing with long lists.

Merge Sort was chosen because it balances performance, stability and safety; making it the most appropriate recursive algorithm for sorting real-world, text-based applicant data in this Java project.

## Justification for Searching Algorithm

For efficiently finding a person by name in the sorted list of applicants, Binary Search was chosen because is the most appropriate algorithm. The list of names in the project is already sorted alphabetically when using a sorting algorithm, Binary Search takes advantage of this, reducing the number of comparisons compared to linear approaches. This makes it ideal for large applicant list.

It doesn’t matter if there are 100 or 10,000 names, Binary Search scales efficiently. So, if the program is expected to grow with more applicants or departments in the future, this algorithm ensures fast response times. Unlike Linear Search, Binary Search cuts the search space in half at each step, making it far more optimal for already-sorted name lists.

Overall, Binary Search was chosen because is the most efficient, scalable and appropriate searching algorithm the sorted applicant list. It provides fast results and integrates with the Java program, making it the best fit over linear or other complex alternatives.

## GitHub Link