A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Mario Rubén Arturo Vera Rodríguez |
| *Student Number* | 2024570 |
| *Module Title* | Algorithms and Constructs |
| *Assessment Title* | System Modelling & Build |
| *Lecturer/Supervisor* | Dr. Muhammad Iqbal |
| *Assessment Due Date* | Saturday 10th May 2025 |
| *Date of Submission* | Saturday 10th May 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.

Table of Contents

[System Modelling & Build 3](#_Toc197811076)

[Sorting and Searching Algorithm Justification 3](#_Toc197811077)

[GitHub Repository 3](#_Toc197811078)

[Link to access repo 3](#_Toc197811079)

# **System Modelling & Build**

## Sorting and Searching Algorithm Justification

For this project, Merge Sort was chosen as the most appropriate sorting algorithm due to several practical and educational factors. One key reason is that it satisfies the requirement for a recursive algorithm, which was explicitly stated in the assignment brief.

In terms of suitability, Merge Sort performs well on datasets that may vary in size or order, and it does not degrade significantly under worst-case scenarios. This contrasts with simpler alternatives like Bubble Sort or Insertion Sort, which may be easier to implement but offer poor performance as data grows, and are not recursive by nature.

While Quick Sort is also a powerful recursive algorithm, it can suffer from performance issues in the worst case if the data is nearly sorted or contains many duplicates. Merge Sort, in contrast, offers more predictable behaviour and was easier to implement correctly within the project’s timeframe.

The decision to use Merge Sort was also influenced by its ability to handle complex objects such as the Applicant class, allowing the program to sort by surname in a stable and consistent manner. It integrates smoothly with object-oriented design and produces a reliable order, which was crucial for displaying the top 20 applicants as specified.

Overall, Merge Sort was selected as the best fit because it met all project criteria, offered consistent performance, was safer than Quick Sort for general use, and more appropriate than non-recursive or less efficient algorithms.

# **GitHub Repository**

## Link to access repo

The full source code for this project is available on GitHub: https://github.com/rubencct/2024570\_CA2