[**Introduction 3**](#_gjdgxs)

[**Modules in the project: 3**](#_30j0zll)

[**Architecture Diagram 4**](#_1fob9te)

[Instructions: 4](#_3znysh7)

[**Module Description 5**](#_2et92p0)

[Programming Details naming conventions to be used: 5](#_tyjcwt)

[Field/table details:(eg university)[you consider you module ] 5](#_3dy6vkm)

[Algorithm Details: 6](#_1t3h5sf)

[(i)Sorting 6](#_4d34og8)

[(ii)Searching 6](#_2s8eyo1)

[(ii) Storing the details in a text file 6](#_17dp8vu)

[**Sample coding Template(Eg:University module) 6**](#_3rdcrjn)

[university.cs 6](#_26in1rg)

[Instruction to use Sample Code: 9](#_lnxbz9)

[**ChatGPT Usage 10**](#_35nkun2)

[Program Generated by ChatGPT 10](#_1ksv4uv)

[Instruction to use ChatGPT or other LLM Models 15](#_44sinio)

[**Rubrics 15**](#_2jxsxqh)

| **Field Name** | **Data type** |
| --- | --- |
| id | integer |
| univ\_code | String |
| univ\_name | String |
| univ\_address | String |
| univ\_email | String |
| univ\_website | String |

| **Criteria** | **Sub-Criteria** | **Marks** | **Description** |
| --- | --- | --- | --- |
| **1. CRUD and Storage** | **CRUD Operations** | 3 | Correct implementation of Create, Update, Retrieve, and Delete operations. |
| **Storage in Text File and Retrieval** | 2 | Proper storage of university data in a text file and correct retrieval of data. |
| **2. Sorting (Total: 10)** | **Sorting Implementation** | 5 | Correct implementation of sorting algorithms *(e.g. on univ\_code, univ\_name, and univ\_email)*. |
| **Comparison with Other Algorithms** | 3 | Proper comparison of sorting algorithm with another (e.g., Quick Sort vs Merge Sort). |
| **Time Complexity and Algorithm Display** | 2 | Correct time complexity analysis and display of sorting algorithm pseudocode. |
| **3. Searching (Total: 10)** | **Searching Implementation** | 5 | Correct implementation of searching algorithms *(eg on univ\_code, univ\_name, and univ\_email)* |
| **Comparison with Other Algorithms** | 3 | Proper comparison of searching algorithm with another (e.g., Binary Search vs Linear Search). |
| **Time Complexity and Algorithm Display** | 2 | Correct time complexity analysis and display of searching algorithm pseudocode. |
| **4. Presentation and Reporting (Total: 15)** | **Project Report Preparation and Submission** | 5 | Clear, well-organized project report with all required details. |
| **Presentation** | 10 | Includes slide preparation (5 marks) and clarity of presentation (5 marks), adhering to instructions. |
| **5. Good Housekeeping Practices (Total: 10)** | **Source Code Control Tool Usage** | 5 | Efficient use of tools like GitHub, GitLab, or Azure DevOps for version control. |
| **Ethical Usage of ChatGPT and LLM Models** | 5 | Responsible and ethical use of AI tools like ChatGPT for assistance without plagiarism. |
| **Total** |  | **50** |  |

# CS204 Project Documentation

## Course Objective Setting Module: Bubble Sort vs Selection Sort

### Objective

The purpose of this project is to implement and compare the Bubble Sort and Selection Sort algorithms within the course objective setting module. The application aims to manage and sort course objectives based on specific attributes such as objective codes or descriptions. By analyzing these algorithms, the project seeks to determine their efficiency in terms of time complexity and practical usability.

### Modules

1. Create Module: Adds new course objectives to the system.  
2. Update Module: Modifies existing course objective details.  
3. Retrieve Module: Fetches and displays course objectives.  
4. Delete Module: Removes unwanted course objectives.  
5. Sorting Module: Implements Bubble Sort and Selection Sort to order course objectives by code or description.  
6. Searching Module: Retrieves specific objectives based on keywords.

### Implementation

The sorting module employs two algorithms:  
- Bubble Sort: Repeatedly compares adjacent elements and swaps them if they are in the wrong order.  
- Selection Sort: Finds the minimum element in each iteration and places it at the beginning.

Both algorithms are evaluated using datasets of varying sizes to compare their time complexity and suitability for real-world use cases. The results are presented in tabular form.

### Algorithm Comparison

1. Time Complexity:  
- Bubble Sort: O(n²) for both average and worst-case scenarios.  
- Selection Sort: O(n²) for both average and worst-case scenarios, but generally performs fewer swaps.  
2. Stability:  
- Bubble Sort: Stable (preserves the order of duplicate elements).  
- Selection Sort: Unstable (may not preserve the order of duplicate elements).  
3. Practical Use:  
- Bubble Sort is more suitable for educational purposes and small datasets.  
- Selection Sort is preferred for scenarios with minimal memory write operations.

### Evaluation

The comparison of Bubble Sort and Selection Sort reveals their respective strengths and weaknesses. Bubble Sort is ideal for scenarios requiring a stable sorting algorithm, whereas Selection Sort is better suited for applications with constraints on memory writes. The project demonstrates the practical implementation of both algorithms and provides insights into their applicability in managing course objectives.

### Conclusion

This project successfully implements the course objective setting module using Bubble Sort and Selection Sort. Through comprehensive analysis and comparison, the project highlights the importance of algorithm selection based on the requirements of specific applications. Future work may include integrating more advanced sorting techniques such as Merge Sort or Quick Sort for improved performance.