Streamlining OBE Department Module

Efficient Data Handling Through OBE

Presented by :

Mukul Kumar (AP23110010700)
Krishna Chaitanya (AP23110011025)
Justin Joshua (AP23110011018)
V.Agastya(AP23110011048)
I.Viswanadh (AP23110011029)

Table of Contents

Photo by Pexels

1

OBE Implementation Overview

Streamlined Academic Workflows

- **OBE Adoption**: SRM University (AP) implements **Outcome-Based Education (OBE)** for streamlined academic processes.
- **Department Module**: Designed for efficient data management, supporting **CRUD operations**.
- **Optimized Algorithms**: Integrates **Merge Sort** for sorting and **Binary Search** for quick data retrieval.
- **Enhanced Workflows**: Ensures effective academic workflows with comprehensive data handling and processing capabilities.
- **Efficiency Focus**: Combines robust algorithms with CRUD operations to optimize departmental data management.

2

Architecture Overview

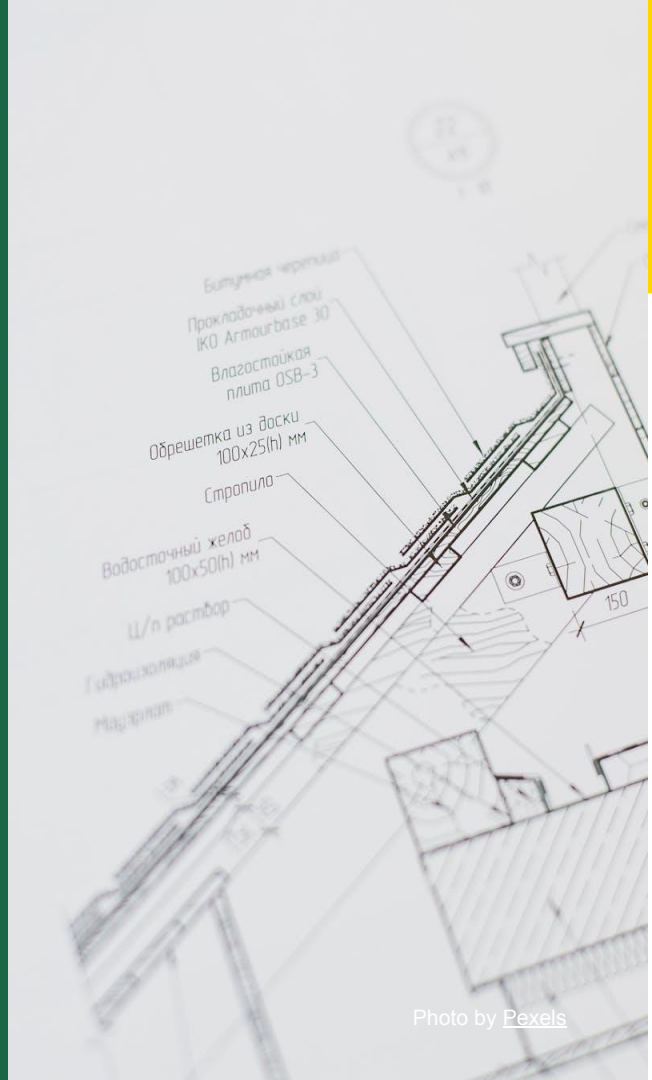

Photo by Pexels

Efficient Data Handling

Architecture Overview: Focuses on efficient data management within the Department Module of the OBE system.

Key Components:

- **CRUD Operations**: Create, Retrieve, Update, Delete functionalities for managing department data.
- **Sorting Algorithms**: Includes **Merge Sort** for optimized data organization.
- **Search Capabilities**: Implements **Binary Search** for quick data retrieval.

Efficiency: Streamlines departmental data handling through integrated sorting, searching, and CRUD processes.

Visualization: The architecture diagram highlights these components and their interplay for optimized operations.

Photo by Pexels

3

Module Functions

CRUD Operations & Data Optimization

- **CRUD Functionality**: Supports **Create**, **Retrieve**, **Update**, and **Delete** operations for managing department data.
- **Data Storage**: Data is stored in a text file for easy access and modification.
- **Sorting Algorithms**: Implements **Merge Sort** and **Selection Sort** for efficient data organization.
- **Efficiency**: Combines CRUD operations with optimized sorting for streamlined data management in the OBE Department Module.

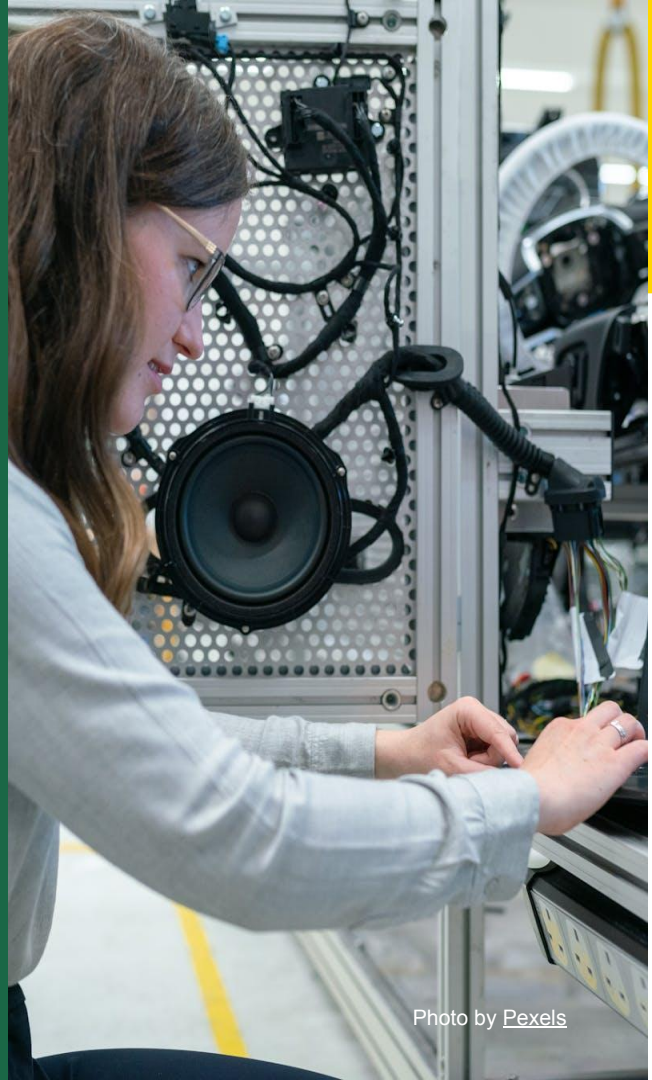
Photo by Pexels

4

Programming Structure

Structured Data Handling

- **Structured Programming**: Emphasizes file naming conventions and clear function names for efficient data handling.
- **Sorting and Searching**: Implements **Merge Sort** and **Binary Search** for optimized performance.
- **Efficiency**: Combines structured naming practices with robust algorithms for streamlined processing in the Department Module.
- **Integrated Approach**: Ensures effective data management through clear programming structure and algorithmic efficiency.

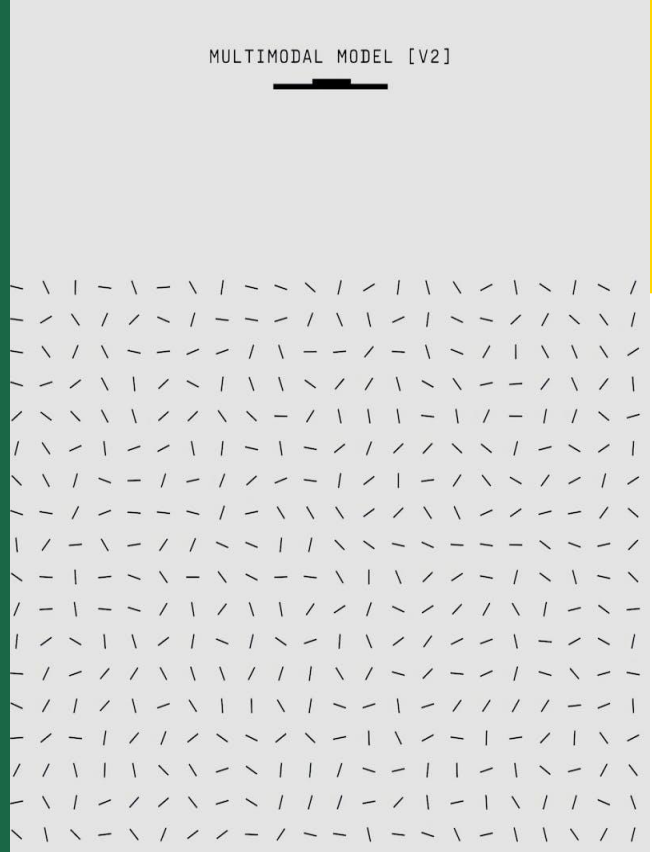

Photo by Pexels

5

Storage File & Sorting Algorithm

Efficient Storage & Sorting Processes

- **Data Storage**: Departmental data is stored in **'department_setting.txt'**.
- **Sorting Algorithm**: Merge Sort is used for efficient and optimized data sorting within the OBE system.
- **Efficiency**: Combines structured storage with a **divide-and-conquer** sorting approach to ensure streamlined data handling.
- **System Optimization**: The use of Merge Sort ensures quick and effective sorting of departmental records.

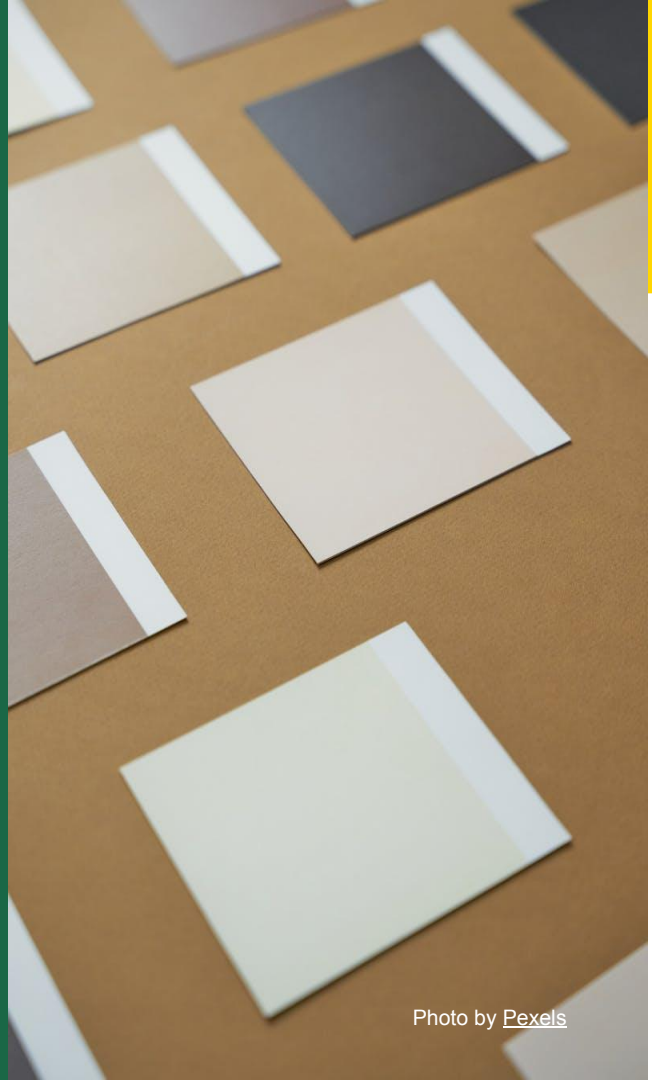

Photo by Pexels

6

Sorting Comparison

Optimized Sorting Techniques

- **Merge Sort**: Utilizes a divide-and-conquer approach, sorting elements by dividing and merging, with a time complexity of **O(n log n)**.
- **Selection Sort**: Uses an iterative selection method to find and place the smallest element, resulting in **$O(n^2)$** time complexity.
- **Efficiency Comparison**: Merge Sort outperforms Selection Sort for large datasets due to its logarithmic efficiency.
- **Sorting Methods**: Merge Sort's divide-and-conquer strategy is more efficient than the iterative process of Selection Sort, especially in handling larger inputs.

Photo by Pexels

7

Searching Algorithm

Efficient Data Retrieval

- **Binary Search**: Key algorithm in the OBE Department Module for quick retrieval from sorted lists.
- **Efficiency**: Locates elements by comparing with the middle value and narrowing the search, achieving **O(log n)** time complexity.
- **Optimized Data Processing**: Enhances retrieval efficiency and streamlines data processing within the module.
- **Suitability**: Ideal for sorted lists, ensuring quick and reliable element retrieval.

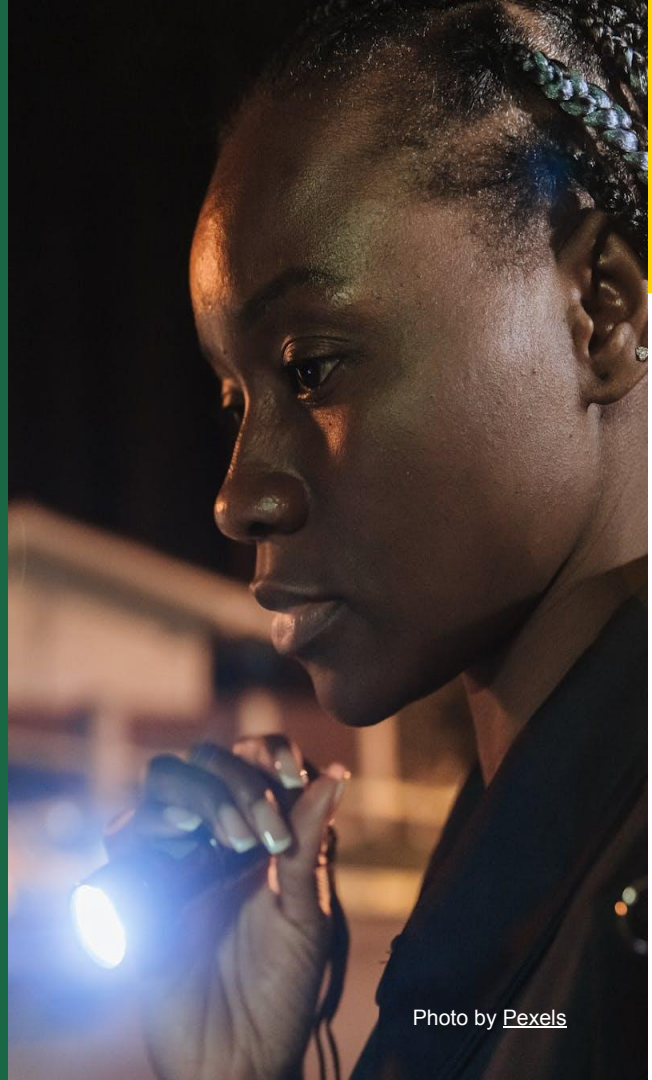
Photo by Pexels

8

Searching Comparison

Comparison of Search Algorithms

- **Binary Search**: Efficiently narrows the search space by dividing it, achieving a time complexity of **O(log n)**.
- **Linear Search**: Sequentially examines each element, leading to a time complexity of **O(n)**.
- **Efficiency Comparison**: Binary Search is significantly faster than Linear Search for large datasets due to its logarithmic approach.
- **Application**: Binary Search is more suitable for sorted data, enabling rapid element retrieval, unlike the slower sequential method of Linear Search.

9

Conclusion

The Department Module successfully manages department records, with efficient sorting and searching algorithms tailored for optimized data handling. Merge Sort and Binary Search provide quick, scalable performance, while comparisons with Selection Sort and Linear Search demonstrate the importance of algorithmic choice in enhancing system efficiency.