Internal Semester Evaluation (ISE) Part 1

Course: Computer Algorithms (7CS203)

Student Name: Suraj Shantinath Upadhye

PRN: 245200001

Semester: ODD 2025–2026

1. Problem Summary

Selection Sort is a simple comparison-based sorting algorithm. It works by repeatedly selecting the minimum element from the unsorted portion of the array and swapping it with the first element of that portion. The process continues until the array is completely sorted.

Algorithm Steps:

1. Start with the first element of the array as the minimum.
2. Compare it with all remaining elements to find the true minimum.
3. Swap the found minimum with the first element of the unsorted portion.
4. Move the boundary of the sorted portion one step forward.
5. Repeat until the entire array is sorted.

Features Implemented in this Project:

- Step-by-step visualization of comparisons and swaps.
- Controls: Play, Pause, Step Forward, Step Backward, Reset.
- Multiple input options:
    - User-defined array
    - Random array
    - Preset arrays (Best/Worst case)
- Real-time display of key variables:
    - Array state
    - Number of comparisons
    - Number of swaps
- Time and space complexity notes.

2. Implementation

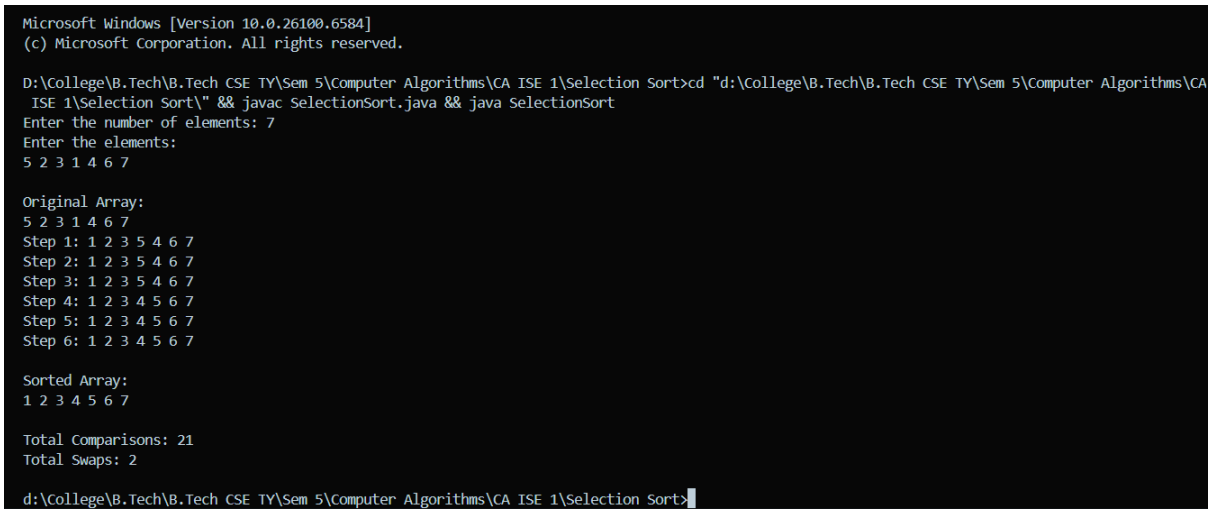
Language Used:

- HTML, CSS, JavaScript for visualization
- Java for console implementation.

Key Implementation Details:

- Each step of Selection Sort is stored in a steps array.
- Active comparisons are highlighted in red, swaps in green.
- Users can control the execution speed (Slow, Medium, Fast).
- Supports both manual and random array inputs.

Visualization Output Console Based:



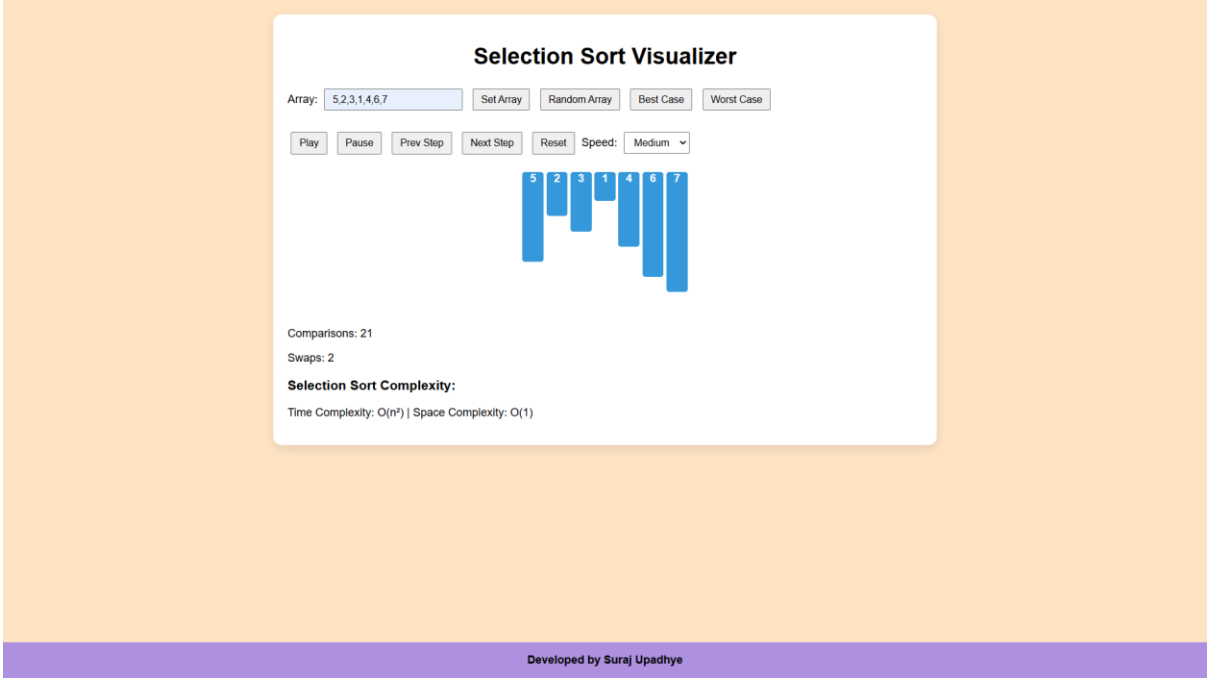
GitHub Repo Link :

https://github.com/Suraj-Upadhye/Selection-Sort-Visualizer

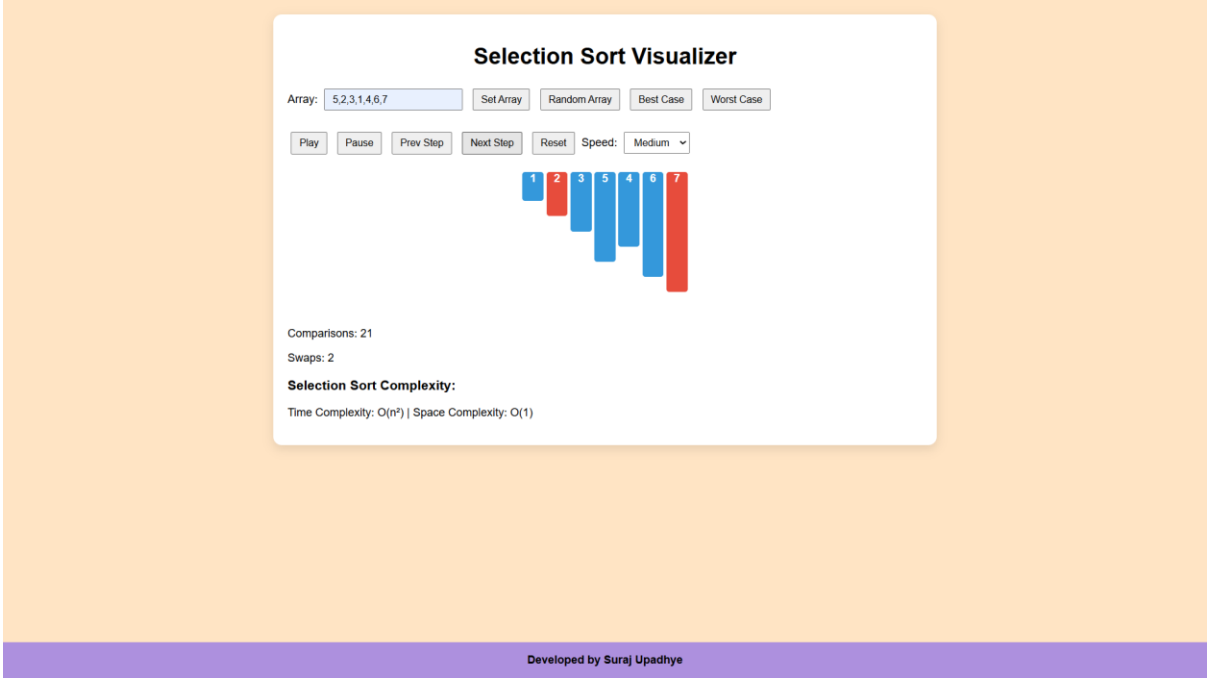
3. Screenshots

Screenshot 1: Selection Sort Visualizer Home Page:

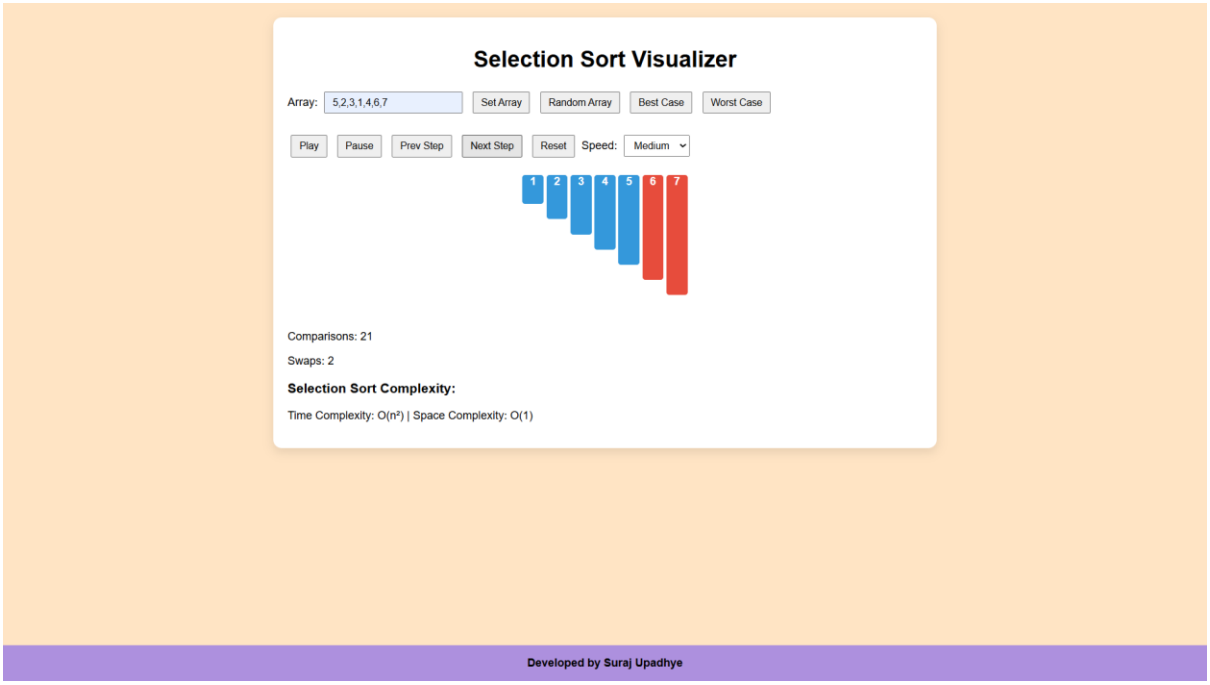
Screenshot 2: Initial Array Input (Custom):



Screenshot 3: Stepwise Visualization:



Screenshot 4: Final Sorted Array:



4. Complexity Analysis

Case	Comparisons	Swaps	Time Complexity	Space Complexity
Best Case	n(n-1)/2	<= n-1	$O(n^2)$	O(1)
Worst Case	n(n-1)/2	<= n-1	$O(n^2)$	O(1)
Average Case	n(n-1)/2	<= n-1	$O(n^2)$	O(1)

Observations:

- Selection Sort always performs the same number of comparisons, regardless of array order.
- The algorithm is **in-place**; no extra space is needed.
- Swaps are minimal compared to Bubble Sort, making it slightly better in practice for small arrays.

5. Conclusion

This project demonstrates Selection Sort with an interactive, stepwise visualization. The visualization helps students understand:

- How the algorithm finds the minimum element at each iteration.
- How comparisons and swaps change the array state.
- The overall efficiency in terms of time and space complexity.

The project meets the ISE requirements with:

- Correct implementation producing expected outputs.
- Step-by-step interactive visualization.
- Multiple input options and control features.
- Clear documentation for reproducibility.