**Title:** Test Plan for Data Structure and Algorithm – Simple Data Type

**By whom:** Van Minh Le

**Date:** 23.03.2025

**Version:** 1.0

**1. Introduction**

This test plan defines the strategy, process, and workflow for testing the JavaScript program that implements data structure operations and search algorithms. The goal is to verify the correctness, functionality, and efficiency of the implemented algorithms.

**1.1. Scope**

**1.1.1. In Scope**

The following functionalities will be tested:

* Building an array with predefined numbers.
* Sorting the array in ascending order.
* Inserting new numbers while maintaining the sorted order.
* Removing numbers while maintaining the sorted order.
* Implementing and testing **sequentialSearch** function.
* Implementing and testing **binarySearch** function.
* Code comments and adherence to coding guidelines.
* Debugging using browser developer tools.
* Version control with GitHub.

**1.1.2. Out of Scope**

* Performance benchmarking.
* Testing with extremely large data sets.
* UI or user interface components.

**1.2. Quality Objective**

The objective of this test is to ensure:

* All required operations on arrays function correctly.
* Searching algorithms return the correct index or -1 if not found.
* Code follows JavaScript coding standards.
* Debugging tools are used effectively.
* All test cases pass before submission.

**1.3. Roles and Responsibilities**

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| Developer | Implements functions and ensures correctness. |
| Tester | Executes test cases and records results. |
| Reviewer | Reviews test results and verifies code quality. |

**2. Test Methodology**

**2.1. Test Levels**

* **Unit Testing**: Each function is tested individually.
* **Integration Testing**: Ensure array operations work together correctly.
* **System Testing**: Verify that all functionalities meet the requirement.

**2.2. Suspension Criteria and Resumption Requirements**

* Testing is suspended if critical bugs block further execution.
* Testing resumes once the issues are fixed and verified.

**2.3. Test Completeness**

The testing is considered complete when:

* 100% of the required functionalities are tested.
* All test cases pass without critical defects.
* Code meets JavaScript coding guidelines.

**3. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Status |
| TC-001 | Sort the array in ascending order | [11, 5, 8, 3, 25, 16, 31, 45, 14, 20] | [3, 5, 8, 11, 14, 16, 20, 25, 31, 45] | Pass |
| TC-002 | Insert 19, 23, 30 into sorted array | [3, 5, 8, 11, 14, 16, 20, 25, 31, 45] | [3, 5, 8, 11, 14, 16, 19, 20, 23, 25, 30, 31, 45] | Pass |
| TC-003 | Remove 8, 31 from array | [3, 5, 8, 11, 14, 16, 19, 20, 23, 25, 30, 31, 45] | [3, 5, 11, 14, 16, 19, 20, 23, 25, 30, 45] | Pass |
| TC-004 | Sequential search (value exists) | ([array], 20) | Index of 6 | Pass |
| TC-005 | Sequential search (value does not exist) | ([array], 4) | -1 | Pass |
| TC-006 | Binary search (value exists) | ([array], 11) | Index of 2 | Pass |
| TC-007 | Binary search (value does not exist) | ([array], 100) | -1 | Pass |

**4. Resource & Environment Needs**

**4.1. Testing Tools**

* Code Editor: VS Code
* Debugging: Chrome Developer Tools
* Version Control: GitHub

**4.2. Test Environment**

* OS: Windows
* Browser: Chrome (latest version)
* JavaScript: ECMAScript 6+

**5. Terms/Acronyms**

|  |  |
| --- | --- |
| TERM/ACRONYM | DEFINITION |
| API | Application Program Interface |
| AUT | Application Under Test |
| JS | JavaScript |