Title: Data Structures and Algorithms – Complex Data Types

By: Van Minh Le

Date: 23.03.2025

Version: 1.0

**1. Introduction**

This test plan defines the testing strategy, scope, quality objectives, and methodology for testing the implementation of data structures and algorithms using JavaScript, specifically focusing on operations with an array of movie objects.

**1.1 Scope**

**1.1.1 In Scope**

* Implementing a Movie class with Movie ID, Title, Year, and Rating properties.
* Creating an array of at least 10 movie objects with unique Movie IDs.
* Sorting the array of movies by Movie ID in ascending order.
* Implementing and testing a search algorithm (sequential or binary search) to locate a movie by its Movie ID.

**1.1.2 Out of Scope**

* Performance testing.
* Integration with external APIs or databases.
* UI testing.

**1.2 Quality Objective**

* Ensure all implemented functionalities meet specified requirements.
* Validate that sorting and searching functions work as expected.
* Ensure test coverage for all testable requirements.
* Detect and resolve bugs before submission.

**1.3 Roles and Responsibilities**

* **Developer**: Implement the JavaScript functions and algorithms.
* **Tester**: Execute test cases, document results, and report bugs.
* **Reviewer**: Verify the implementation, test results, and overall correctness.

**2. Test Methodology**

**2.1 Test Levels**

* **Unit Testing**: Validate individual functions for sorting and searching.
* **Integration Testing**: Ensure the array operations (creation, sorting, searching) work together correctly.
* **Regression Testing**: Verify that modifications do not introduce new bugs.

**2.2 Suspension Criteria and Resumption Requirements**

* Testing will be suspended if there are critical defects that prevent further test execution.
* Testing will resume once all blocking issues are resolved.

**2.3 Test Completeness**

* All test cases are executed with expected results.
* All critical bugs are fixed.
* Code meets the specified requirements.

**3. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Steps** | **Expected Result** | **Status** |
| TC01 | Create Movie class | Define a class with required properties | Class should be created successfully | Pass |
| TC02 | Create an array of movies | Initialize an array with at least 10 movie objects | **-A021:** The Shawshank Redemption, Year: 1994, Rating: 9.3 **-C020:** The Godfather, Year: 1972, Rating: 9.2 **--D003:** The Dark Knight, Year: 2008, Rating: 9  **-A104:** The Godfather Part II, Year: 1974, Rating: 9 **-B005:** 12 Angry Men, Year: 1957, Rating: 9  **-A106:** Schindler's List, Year: 1993, Rating: 9 **-S007:** The Lord of the Rings, Year: 2003, Rating: 9 **-D108:** Pulp Fiction, Year: 1994, Rating: 8.9 **-L009:** The Good, the Bad and the Ugly, Year: 1966, Rating: 8.8 **-B110:** Fight Club , Year: 1999, Rating: 8.81 | Pass |
| TC03 | Sort movies by Movie ID | Implement sorting algorithm and execute | **A021:** The Shawshank Redemption, Year: 1994, Rating: 9.3 **A104:** The Godfather Part II, Year: 1974, Rating: 9 **A106:** Schindler's List, Year: 1993, Rating: 9 **B005:** 12 Angry Men, Year: 1957, Rating: 9 **B110:** Fight Club , Year: 1999, Rating: 8.81 **C020:** The Godfather, Year: 1972, Rating: 9.2 **D003:** The Dark Knight, Year: 2008, Rating: 9 **D108:** Pulp Fiction, Year: 1994, Rating: 8.9 **L009:** The Good, the Bad and the Ugly, Year: 1966, Rating: 8.8 **S007:** The Lord of the Rings, Year: 2003, Rating: 9 | Pass |
| TC04 | Sequential Search | Implement and test sequential search function | * Case 01:   Search the movie with ID: **S007** in movie list: **Found it at the index: 9**   * Case 02:   Search the movie with ID: **D007** in movie list: **Found it at the index: -1** | Pass |
| TC05 | Binary Search | Implement and test binary search function | * Case 01:   Search the movie with ID: **A021** in movie list: **Found it at the index: 0**   * Case 02:   Search the movie with ID: **A000** in movie list: **Found it at the index: -1** | Pass |

**4. Resource & Environment Needs**

**4.1 Testing Tools**

* JavaScript runtime (Node.js / Browser Console)
* Code Editor (VS Code, Sublime Text, etc.)
* GitHub for version control

**4.2 Test Environment**

* Windows
* Node.js installed for testing in a non-browser environment
* Developer tools for debugging

**5. Terms/Acronyms**

|  |  |
| --- | --- |
| **TERM/ACRONYM** | **DEFINITION** |
| AUT | Application Under Test |
| ID | Identifier |
| TC | Test Case |

This document will be updated based on test execution results and findings.