

**Abbottabad University of Science & Technology**

**SOFTWARE REQUIREMENTS SPECIFICATION   
(SRS DOCUMENT)**

**For**

**< Sorting Algorithm and Visualizer>**  
 Version 1.0

***By***

**Saqib Hafeez 14805**

**Supervisor**

**(Sir Jamal Abdul Ahad)**

**Contents**

[**1. Introduction** 3](#_Toc185842880)

[**1.1 Purpose** 3](#_Toc185842881)

[**1.2 Document Conventions** 3](#_Toc185842882)

[**1.3 Project Scope** 3](#_Toc185842883)

[**1.4 References** 4](#_Toc185842884)

[**2. Overall Description** 4](#_Toc185842885)

[**2.1 Product Perspective** 4](#_Toc185842886)

[**2.2 User Classes and Characteristics** 5](#_Toc185842887)

[**2.3 Operating Environment** 5](#_Toc185842888)

[**2.4 Design and Implementation Constraints** 5](#_Toc185842889)

[**2.5 Assumptions and Dependencies** 6](#_Toc185842890)

[**3. System Features** 6](#_Toc185842891)

[**Feature 1: Visualization of Sorting Algorithms** 6](#_Toc185842892)

[**Feature 2: Algorithm Selection** 6](#_Toc185842893)

[**5. External Interface Requirements** 7](#_Toc185842894)

[**5.1 User Interfaces** 7](#_Toc185842895)

[**5.3 Hardware Interfaces** 7](#_Toc185842896)

[**6. Quality Attributes** 7](#_Toc185842897)

[**6.1 Performance** 7](#_Toc185842898)

[**6.2 Reliability** 7](#_Toc185842899)

[**6.3 Usability** 7](#_Toc185842900)

[**6.4 Security** 8](#_Toc185842901)

[**6.5 Maintainability** 8](#_Toc185842902)

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide the Software Requirements Specification (SRS) for the **Sorting Algorithm Visualizer**. This document outlines the functional and non-functional requirements, design constraints, and other necessary components for the successful development of the project.

**1.2 Document Conventions**

The following conventions will be used throughout this document:

* **Bold**: Represents names of important features or terms.
* **Italics**: Represents references or additional explanations.
* **Code**: Represents specific code examples or syntax.

**1.3 Project Scope**

**1.3.1 Scope Definition**

This project focuses on building a **Sorting Algorithm Visualizer**. The visualizer will help users to visually understand and compare different sorting algorithms. Users will be able to see how algorithms like Bubble Sort, Merge Sort, Quick Sort, etc., work on an array of random numbers.

**1.3.2 Core Features**

* Visualization of different sorting algorithms.
* Ability to interact with the visualizer by inputting arrays of various sizes.
* Ability to select different sorting algorithms.
* Step-by-step visualization with pause, resume, and restart features.
* Display of algorithm performance metrics (e.g., time complexity).

**1.3.3 Subsequent Releases**

Future versions may include additional features such as:

* Support for more sorting algorithms.
* Optimization algorithms.
* Comparison of time and space complexities.
* Customizable visual styles (themes and color schemes).
* Mobile app support for better accessibility.

**1.3.4 Alignment with User and Business Goals**

The project aims to provide educational value for learners in understanding sorting algorithms and their performance. This visualizer will serve as a tool for instructors, students, and developers who are learning or teaching algorithms.

**1.4 References**

* **Sorting Algorithms – Wikipedia**: <https://en.wikipedia.org/wiki/Sorting_algorithm>
* **Design Principles – Software Engineering**: https://www.oreilly.com/library/view/software-engineering-9th/9780133065201/

**2. Overall Description**

**2.1 Product Perspective**

**2.1.1 Product Context**

The Sorting Algorithm Visualizer will be a web-based application built using HTML5, CSS, and JavaScript. It will provide a graphical representation of sorting algorithms' inner workings.

**2.1.2 Product Origin**

The idea for this visualizer originated from the need to teach sorting algorithms effectively. Traditional textbooks and lectures may not provide adequate visual explanations of how algorithms work on actual data, leading to confusion among students.

**2.1.3 Product Relationship to Existing Systems**

This project does not rely on existing systems but instead serves as an educational tool and a comparison platform for sorting algorithms.

**2.1.4 Product Ecosystem**

The Sorting Algorithm Visualizer will run in web browsers and will be compatible with most modern browsers (e.g., Chrome, Firefox, Edge).

**2.2 User Classes and Characteristics**

**2.2.1 Beginners**

Users with little to no experience in algorithms. They will use the visualizer to learn the basic working of sorting algorithms.

**2.2.2 Intermediate Users**

Users with some programming knowledge. They will use the visualizer to better understand different sorting techniques and their performance.

**2.2.3 Advanced Users**

Users who have deep knowledge of algorithms. They may use the visualizer to analyze the behavior of sorting algorithms in different scenarios or test edge cases.

**2.2.4 Alignment with User Needs**

The visualizer is designed to cater to all levels of users by providing clear, step-by-step visualizations, interactive controls, and performance metrics.

**2.3 Operating Environment**

**2.3.1 Hardware Platform**

* Computers (Desktops and Laptops) with internet access.

**2.3.2 Operating Systems and Versions**

* **Windows 10 or later**
* **MacOS X 10.14 or later**
* **Linux (Any modern distro)**

**2.4 Design and Implementation Constraints**

**2.4.2 Database Technology**

* Not applicable. This application will be front-end based with no need for server-side databases.

**2.4.4 Third-Party Integrations**

* Use of JavaScript libraries (e.g., D3.js for visualization, React for UI).

**2.4.5 User Interface Design**

* A simple and clean design focusing on user experience. The UI will provide easy navigation for selecting algorithms and adjusting settings.

**2.5 Assumptions and Dependencies**

**2.5.1 Assumptions**

* The application will run in a modern browser with JavaScript support.
* The user understands sorting algorithms.

**2.5.2 Dependencies**

* D3.js library for dynamic visualizations.
* ReactJS for building interactive UI components.

**3. System Features**

**Feature 1: Visualization of Sorting Algorithms**

* **Description**: The system should display the sorting process visually. Each algorithm should be represented with colored bars to show the array’s elements.
* **Functional Requirements**:
  + The system should allow users to choose from multiple algorithms.
  + The system should animate each sorting step.
  + Users can pause, resume, or restart the visualization.

**Feature 2: Algorithm Selection**

* **Description**: The system allows users to select different sorting algorithms for visualization.
* **Functional Requirements**:
  + Users can choose from Bubble Sort, Merge Sort, Quick Sort, etc.
  + Each algorithm is explained briefly within the app.

**5. External Interface Requirements**

**5.1 User Interfaces**

**5.1.1 Design Standards and Guidelines**

* Simple, user-friendly UI with easy navigation.
* Clear labeling of buttons and algorithm selection menu.

**5.1.2 Screen Layout and Resolution**

* The UI will be responsive, adjusting to various screen sizes.
* Supported resolutions: 1366x768 and higher.

**5.3 Hardware Interfaces**

**5.3.1 Supported Device Types**

* Desktops, Laptops, and Tablets.

**6. Quality Attributes**

**6.1 Performance**

* The system should be responsive with minimal latency.
* Sorting visualizations should run smoothly, even for larger datasets.

**6.2 Reliability**

* The system should be stable and not crash during algorithm execution.
* The system should handle user interactions (pausing, restarting) without errors.

**6.3 Usability**

* The visualizer should be easy to use, with intuitive controls for selecting and visualizing algorithms.

**6.4 Security**

* Since the application is a client-side application, security risks are minimal, but care should be taken to ensure that no malicious scripts are executed.

**6.5 Maintainability**

* The code should be modular to allow easy updates and additions of new sorting algorithms.