Software Requirements Specification

for

Interactive simulation of various Data structures and Algorithms

Version 1.0 approved

Priyank Mungra, Roll No: CE-081

Yash Amethiya, Roll No: CE-002

Guided by: Prof. Jigar M Pandya

Computer Department

Dharmsinh Desai University

Table of Contents

[1 Introduction 4](#_Toc62236427)

[1.1 Purpose 4](#_Toc62236428)

[1.2 Document Conventions 4](#_Toc62236429)

[1.3 Intended Audience and Reading Suggestions 4](#_Toc62236430)

[1.4 Product Scope 4](#_Toc62236431)

[1.5 References 4](#_Toc62236432)

[2 Overall Description 5](#_Toc62236433)

[2.1 Product Perspective 5](#_Toc62236434)

[2.2 Product Functions 5](#_Toc62236435)

[2.3 User Classes and Characteristics 5](#_Toc62236436)

[2.4 Operating Environment 5](#_Toc62236437)

[2.5 Design and Implementation Constraints 5](#_Toc62236438)

[2.6 Assumptions and Dependencies 6](#_Toc62236439)

[3 External Interface Requirements 6](#_Toc62236440)

[3.1 User Interfaces 6](#_Toc62236441)

[3.2 Hardware Interfaces 6](#_Toc62236442)

[4 Functional Requirements 6](#_Toc62236443)

[4.1 Manage Data Structure 6](#_Toc62236444)

[4.1.1 Selection of Data Structure. 6](#_Toc62236445)

[4.1.2 Select Implementation category. 7](#_Toc62236446)

[4.1.3 Add element to data structure 7](#_Toc62236447)

[4.1.4 Remove element from data structure 7](#_Toc62236448)

[4.2 Manage Algorithms 7](#_Toc62236449)

[4.2.1 Selection of Algorithm 7](#_Toc62236450)

[4.2.2 Add element. 7](#_Toc62236451)

[4.2.3 Remove element from Sorting Algorithm Array 8](#_Toc62236452)

[4.2.4 Executing Algorithm 8](#_Toc62236453)

[4.3 Manage Simulation 8](#_Toc62236454)

[4.3.1 Speed of Simulation 8](#_Toc62236455)

[4.3.2 Play and Pause 8](#_Toc62236456)

[4.3.3 Next step and Previous step 8](#_Toc62236457)

[4.3.4 Reset Simulation 9](#_Toc62236458)

[4.4 Stockpile Management 9](#_Toc62236459)

[4.4.1 Save state 9](#_Toc62236460)

[4.4.2 View/Edit State 9](#_Toc62236461)

[4.4.3 Delete State 9](#_Toc62236462)

[4.5 Session Management 9](#_Toc62236463)

[4.5.1 User Login by OAuth 9](#_Toc62236464)

[4.5.2 Logout 9](#_Toc62236465)

[5 Non-functional Requirements 10](#_Toc62236466)

[5.1 Availability 10](#_Toc62236467)

[5.2 Portability 10](#_Toc62236468)

[5.3 Performance Requirements 10](#_Toc62236469)

[6 Future Extensions 10](#_Toc62236470)

***Revision History Table:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Date of update** | **Purpose** | **Changed By** |
| 1 | 26/12/2020 | SRS Initial | Yash Amethiya,  Priyank Mungra |
| 2 | 11/01/2021 | Add functionalities like OAuth, Login/Registration, Stockpile | Priyank Mungra |
| 3 | 21/01/2021 | Selection of algorithm updated, Reset Simulation functionality added | Yash Amethiya |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document will propose all features and procedures to develop the Interactive Simulation of Data Structures and Algorithms. This Document presents detailed idea of the application. This document mentions interfaces and functionality of working application.

## Document Conventions

SRS: Software Requirements Specification

DNF: Disjunctive Normal Form

GUI: Graphical User Interface

CRUD: create, read, update and delete operations

DSA: Data Structure and Algorithms

OS: Operating System

RAM: Random Access Memory

ROM: Read Only Memory

API: Application programming interface.

## Intended Audience and Reading Suggestions

This Document is meant to be ready by the end-users, developer, project manager, marketing staff, testers and documentation writers.

## Product Scope

The aim behind the implementation of this project to make a clear understandability of various data structures and Algorithms. This will really help learners to understand the concepts really well. This web-application will simulate the data structure operations such as searching, sorting, insertion, deletion, etc., and algorithms like Bubble Sort, Insertion Sort, Quick Sort, Merge Sort, etc.

## References

IEEE Std 830-1998(Revision of IEEE Std 830-1993)

This project will take the [given](https://www.cs.usfca.edu/~galles/visualization/Algorithms.html) project as a reference:

<https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>

# Overall Description

## Product Perspective

This is a self-contained independent application. This application is made for educational purpose. This application will help students to understand the concepts of data structure and algorithm very well as can take advantage of visual representation of DSA.

## Product Functions

1. Simulation of Data Structure
2. Simulation of Algorithms
3. Interaction with Visualization

## User Classes and Characteristics

End users can visit and use the web-application and take all the advantages by logging in with their Gmail id or other platforms using OAuth.

## Operating Environment

**Software Requirements:**

* Operating System: Windows 8 or above, Linux, Max OS.
* Internet connectivity is required
* Latest version of browsers like Chrome, Firefox and Safari.

**Hardware Requirements:**

* 4 GB RAM
* 1 GB secondary memory
* Processor: Pentium

## Design and Implementation Constraints

* System will work on wireless network. This is a light application very less amount of RAM is consumed
* Technology used in development are:

Angular Framework

Express.js

Node.js

OAuth

## Assumptions and Dependencies

* End-Users should have a general knowledge of basic computer and internet skills.
* Server should be working every time.
* Application must have user-friendly interface.

# External Interface Requirements

## User Interfaces

* Messages have green color for success and red color for failure.
* Speed slider is provided to increase or decrease animation speed.
* Previous step and next step button for simulation.
* Play and Pause button to toggle the simulation.
* Navigation bar at the top.
* Arrows to denote pointer.
* Bar-chart to represent array with varying values.

## Hardware Interfaces

Screen size should be at least medium (641px to 1007px) for better GUI experience.

# Functional Requirements

The entire project mainly consists of 5 modules, which are as follows:

1. Data Structure
2. Algorithm
3. Simulation
4. Stockpile
5. Managing Application Access.

## Manage Data Structure

This module will manage properties of each data structure and manage different operation of each data structure. Available data structure will be Array, Linked List, Stack, Queue, Graph, Tree.

### Selection of Data Structure.

Description: Users can choose a Data structure from the provided options.

Input: Option to choose Data Structure.

Output: Acknowledgement message.

### Select Implementation category.

Description: Some data structures can be implemented in different manner. For example, Stack can be implemented using array and Linked list both.

Input: implementation category

Output: implementation of data structure using that basic element.

Next: user will be asked to add data to current node.

### Add element to data structure

Description: Users can add element of any data value and insert it to data structure.

Input: Data, Option to add.

Output: Simulation of addition of element.

Process: The element would be added to the data structure according to the properties of that specific data structure and simulation would be done for the same.

Next: User is allowed to perform other operations on data structure.

### Remove element from data structure

Description: Users can remove element from any data structure according to property of that specific data structure. Example: Pop operation on stack.

Input: Option to remove.

Output: Simulation of removal of element.

Process: The element would be removed from data structure according to the properties of that specific data structure.

Next: User is allowed to perform other operations on data structure.

## Manage Algorithms

This module will manage the methods of each Algorithm. Available algorithms will be sorting algorithms like bubble sort, insertion sort, quick sort and merge sort. All the sorting algorithms will be performed on array.

### Selection of Algorithm

Description: Users can choose a sorting algorithm from the provided options.

Input: Option to choose algorithm.

Output: Acknowledgement message and redirect to particular sorting algorithm page.

### Add element.

Description: Users can add numerical values to array.

Input: Space separated numerical values, user selection on add button.

Output: Array with given numerical values will be added and displayed as bar chart.

Process: New array with given values will be created and representation of same array using bar-char would be done.

Next: User is allowed to sort the array and observe the simulation.

### Remove element from Sorting Algorithm Array

Description: Users can remove the specific element from the array.

Input: option to remove.

Output: Updated array would be represented in form of bar chart.

Process: The element would be removed from array and representation of same array using bar-char would be done.

### Executing Algorithm

Description: After users maintains the desired values in the array. He can choose to sort the array.

Input: Option to sort.

Output: Visual Representation of Sorting algorithm would be done.

## Manage Simulation

Users are provided with various options to simulate like play, pause, previous step, next step and speed of simulation.

### Speed of Simulation

Input: Adjusting speed slider.

Output: Acknowledgement message.

Process: The speed of simulation is updated.

### Play and Pause

Description: User can toggle between playing and pausing the simulation.

Input: Option to toggle between play and pause.

Output: Acknowledgement message.

### Next step and Previous step

Description: User can use this functionality to perform step by step execution of algorithm. Previous step may not be available for some algorithms.

Input: Option to select next step and previous step.

Output: Next step or Previous step is executed.

### Reset Simulation

Description: User can reset the simulation back to original state at any point of time.

Input: User selection on Reset button.

Output: Simulation adjusted back to initial state.

## Stockpile Management

Description: Users can save the current state of simulation with the current input data and current animation settings and current state of animation. Users can later access this state from Stockpile and continue learning from the point he/she had left in past.

### Save state

Description: User can save the current state into the stockpile.

Input: Option to save to stockpile

Output: Acknowledgement message.

### View/Edit State

Description: Users can view the list of saved states and continue from where they had left their work.

Input: Option to choose from saved sates in Stockpile

Output: The state stored is loaded to screen and user can continue the simulation.

### Delete State

Description: User can delete the state.

Input: Option to delete the state

Output: Confirmation message.

## Session Management

### User Login by OAuth

Description: User can use the application only after they have logged in. User are given login module designed by OAuth technology.

Input: Gmail id or another platform id

Output: Redirect to home page.

### Logout

Input: Option to Logout

Output: Redirect to User Login by OAuth [(4.5.1)](#_User_Login_by)

# Non-functional Requirements

## Availability

System shall be available all the time.

## Portability

As software just needs a browser and is available to almost any type of desktop operating system user can access it from any device anytime.

## Performance Requirements

The software gives high performance as it is light weight and proper designing techniques are used.

## Responsive

Application is responsive with respect to the screen size.

# Future Extensions

Text Editor to be added for user to note down concepts and save them for future.