**3.SYSTEM DESIGN**

**3.1 Introduction**

* System design is the process of defining the architecture, module interfaces and data for a system to satisfy the specified requirements.
* The purpose of the design phase is to plan the solution of the problem specified by the requirement documents.
* This is the first step that moving from problem domain to the solution domain.
* The design of the system is essentially a blueprint or a plan for a solution for the system.

**3.2 Assumptions and Constraints**

An assumption is a condition you think to be true and constrain is fixed limitations of project development.

* All the functional requirement collected from client are sufficient for the project life-cycle.
* All the Non-functional and Specific requirement specified in SRS well enough for the development of system.
* This software is to be completed with time constraint of 3 months.

**3.3 Functional decomposition**

Functional decomposition is the process of taking a complex process and breaking it down into its smaller, simpler parts. Using Functional decomposition larger or complex functionalities are more easily understood. It is mainly used during project analysis phase, so each phase can be viewed as software. So, this has modular with some sub modules.

**3.3.1 System Software Architecture**

**PERFORMANCE ANALYSIS AND SIMULATION OF DATA STRUCTURE**

A

B

C

E

F

G

H

SORTING

ARRAY OPERATION

SEARCHING

STACK

QUEUE

LINKED LIST

TREE

GRAPH

Selection Sort

Insertion Sort

Exchange Sort

Counting Sort

Heap Sort

Radix Sort

Merge Sort

Quick Sort

Brick Sort

Shell Sort

D

A

Bubble Sort

Selection Sort

Insertion Sort

Exchange Sort

Counting Sort

Heap Sort

Bucket Sort

Merge Sort

Quick Sort

Brick Sort

Shell Sort

B

Insertion

Deletion

C

Linear Search

Binary Search

Exponential Search

Interpolation Search

Jump Search

Ternary Search

D

Array implementation

Push

Pop

Linked list Implementation

Push

Pop

E

Array implementation

Insertion

Deletion

Linked list Implementation

Insertion

Deletion

F

Singly Linked List

Doubly Linked List

1

2

1

Insertion at begin

Insertion at end

Insertion at position

Deletion at begin

Deletion at end

Deletion at position

2

Insertion at begin

Insertion at end

Insertion at position

Deletion at begin

Deletion at end

Deletion at position

3

Insertion at begin

Insertion at end

Insertion at position

Deletion at begin

Deletion at end

Deletion at position

4

Insertion at begin

Insertion at end

Insertion at position

Deletion at begin

Deletion at end

Deletion at position

G

Binary Search Tree

Insertion

Deletion

Searching

Pre-order

Post-order

In-order

H

BFS

DFS

Fig 3.1 System Software Architecture

**3.3.2 System Technical Architecture**

User

Internet

Web browser

Application

Fig 3.2 System Technical Architecture

**3.3.3 System Hardware Architecture**

Processing Hardware

Computer

Mass Storage

Display

Input

Fig 3.3 System Hardware Architecture

**3.3.4 External Interfaces**

Not Applicable.

**3.4 Description of Programs**

**3.4.1 Context Flow Diagram (CFD)**

In CFD entire system is considered as a single process. Context flow diagram shows input and outputs of the system. It shows all the external entities that interact with the system and how the data flows between these external entities and system.

Teacher/student

Requesting for array insertion and deletion

Displays the resultant array

Requesting for visualization Sorting

Teacher/Student

Requesting for visualization Searching

Requesting for stack push and Pop

Displays the resultant stack

Display searched element in array

Display sorted array

Teacher/student

Display resultant linked list

Teacher/Student

Requesting for queue insertion and deletion

Requesting for visualization of linked-list

Display resultant tree

Requesting for visualization of tree

Requesting for visualization of graph

Display resultant graph

Display resultant queue

Fig 3.4 Context Flow Diagram

**3.4.2 Data Flow Diagram (DFDs Level 0, Level 1, Level 2)**

Data flow diagram shows the flow of data through system. Data flow diagrams also called the data flow graphs. It views a system as a function that transforms the inputs into desired outputs. It aims to capture the transformation that taken place within a system to the input data so that eventually the output data is produced.

|  |  |  |
| --- | --- | --- |
| **Symbols** | **Name** | **Description** |
|  | **Process** | **It performs transformation of data from one state to another.** |
|  | **Source /Sink** | **It represents the external entity that may be either source or Sink.** |
|  | **Flow of data** | **It represents the flow of data from source to destination** |
|  | **Data Source/Data storage** | **It is the place where data is stored.** |

Table 3.1 Data Flow Diagram.

**3.4.3 LEVEL 0**

USER

fig 3.5 DFD for modules (Level 0)

**3.5 Description of the Component**

**3.5.1 Sorting module**

**LEVEL-1**

Teacher/Staff

Fig 3.6 Sorting (Level-1)

**3.5.1.1 Bubble Sort**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.7 Bubble Sort

**3.5.1.1.1 Input**

Array of numbers

**3.5.1.1.2 Process**

This sorting technique takes the array of numbers and compares two adjacent elements and swaps them until they are in order.

**3.5.1.1.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.1.4 Interface with other functional component**

Not applicable.

**3.5.1.1.5 Resource allocation**

style.css file.

**3.5.1.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.2 Selection Sort**

**LEVEL-2**

FALSE

Teacher/ student

TRUE

Fig 3.8 Selection Sort

**3.5.1.2.1 Input**

Array of numbers

**3.5.1.2.2 Process**

This sorting technique takes the array of numbers then find smallest element and puts in first position. Then finds second smallest element in the list and so on until they are in order.

**3.5.1.2.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.2.4 Interface with other functional component**

Not applicable

**3.5.1.2.5 Resource allocation**

style.css file.

**3.5.1.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.3 Insertion Sorting**

**LEVEL-2**

FALSE

Teacher/ student

TRUE

Fig 3.9 Insertion Sorting

**3.5.1.3.1 Input**

Array of numbers

**3.5.1.3.2 Process**

This sorting technique takes the array of numbers and compares the selected element with the previous array elements and then displays the elements in the sorted order..

**3.5.1.3.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.3.4 Interface with other functional component**

Not applicable

**3.5.1.3.5 Resource allocation**

style.css file.

**3.5.1.3.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.4 Exchange Sorting**

**LEVEL-2**

FALSE

Teacher/ student

TRUE

Fig 3.10 Exchange Sort

**3.5.1.4.1 Input**

Array of numbers

**3.5.1.4.2 Process**

This sorting technique takes the array of numbers and compares the first element with each following element of the array and then displays the elements in the sorted order.

**3.5.1.4.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.4.4 Interface with other functional component**

Not applicable

**3.5.1.4.5 Resource allocation**

style.css file.

**3.5.1.4.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.5 Counting Sorting**

Teacher/ student

**LEVEL-2**

FALSE

TRUE

Fig 3.11 Counting Sort

**3.5.1.5.1 Input**

Array of numbers

**3.5.1.5.2 Process**

This sorting technique takes the array of numbers and sorts the array also counts the number of repetitions of array elements.

**3.5.1.5.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.5.4 Interface with other functional component**

Not applicable

**3.5.1.5.5 Resource allocation**

style.css file.

**3.5.1.5.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.6 Heap Sorting**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.12 Heap Sort

**3.5.1.6.1 Input**

Array of numbers

**3.5.1.6.2 Process**

This sorting technique takes the array of numbers later it divides the sorted and unsorted region and it iteratively shrinks the unsorted region by extracting the largest element from it and inserting it into sorted region.

**3.5.1.6.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.6.4 Interface with other functional component**

Not applicable

**3.5.1.6.5 Resource allocation**

style.css file.

**3.5.1.6.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.7 Bucket Sorting**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.13 Radix Sort

**3.5.1.7.1 Input**

Array of numbers

**3.5.1.7.2 Process**

It works by distributing an element of an array into a number of buckets. Each bucket is then sorted individually either using a different sorting algorithm or by recursively applying the bucket sorting algorithm.

**3.5.1.7.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.7.4 Interface with other functional component**

Not applicable

**3.5.1.7.5 Resource allocation**

style.css file.

**3.5.1.7.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.8 Merge Sorting**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.14 Merge Sort

**3.5.1.8.1 Input**

Array of numbers

**3.5.1.8.2 Process**

This sorting technique takes the array of numbers and compares the selected element with the previous array elements and then displays the elements in the sorted order.

**3.5.1.8.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.8.4 Interface with other functional component**

Not applicable

**3.5.1.8.5 Resource allocation**

style.css file.

**3.5.1.8.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.9 Quick Sorting**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.15 Quick Sort

**3.5.1.9.1 Input**

Array of numbers

**3.5.1.9.2 Process**

This sorting technique takes the array of numbers and then picks an element as a pivot and partitions the given array around picked pivot.

**3.5.1.9.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.9.4 Interface with other functional component**

Not applicable

**3.5.1.9.5 Resource allocation**

style.css file.

**3.5.1.9.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.10 Brick Sorting**

Teacher/ student

**LEVEL-2**

FALSE

TRUE

Fig 3.16 Brick Sort

**3.5.1.10.1 Input**

Array of numbers

**3.5.1.10.2 Process**

This sorting technique takes the array of numbers it functions by comparing all odd/even indexed pairs of adjacent elements in the list and, if a pair is in the wrong order the elements are switched. It repeats this step for even/odd indexed pair. Then it alternates between the odd/even and even/odd steps until the list is sorted.

**3.5.1.10.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.10.4 Interface with other functional component**

Not applicable

**3.5.1.10.5 Resource allocation**

style.css file.

**3.5.1.10.6 User interface**

Buttons, labels, textbox, message box.

**3.5.1.11 Shell Sorting**

Teacher/ student

**LEVEL-2**

FALSE

TRUE

Fig 3.17 Shell Sort

**3.5.1.11.1 Input**

Array of number.

**3.5.1.11.2 Process**

This sorting technique takes the array of elements it starts by pairs of elements far apart from each other, then progressively reducing the gap between the elements to be compared.

**3.5.1.11.3 Output**

Displays the sorted bar graph also calculates the time and space complexity.

**3.5.1.11.4 Interface with other functional component**

Not applicable

**3.5.1.11.5 Resource allocation**

style.css file.

**3.5.1.11.6 User interface**

Buttons, labels, textbox, message box.

**3.5.2 Array Operation module**

**LEVEL-1**

Teacher/Staff

Fig 3.18 Array Operation (Level-1)

**3.5.2.1.1 Insertion**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.19 Array Insertion

**3.5.2.1.1 Input**

Array of numbers, number to be inserted, position of inserted number.

* + - * 1. **Process**

This operation takes the array of numbers and checks the position in the linear order then inserts the specified element at specified position.

**3.5.2.1.3 Output**

Displays the array with inserted element using bar graph also calculates the time and space complexity.

**3.5.2.1.4 Interface with other functional component**

Not applicable

**3.5.2.1.5 Resource allocation**

style.css file.

**3.5.2.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.2.2.1 Deletion**

Teacher/ student

**LEVEL-2**

FALSE

TRUE

Fig 3.20 Array Deletion.

**3.5.2.2.1 Input**

Array of numbers, number to be deleted.

**3.5.2.2.2 Process**

This operations takes the array of numbers and checks in the linear order then deletes the specified element.

**3.5.2.2.3 Output**

Displays the final bar graph excluding the deleted element and then displays the position of the element deleted also calculates the time and space complexity.

**3.5.2.2.4 Interface with other functional component**

Not applicable

**3.5.2.2.5 Resource allocation**

style.css file.

**3.5.2.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3 Searching module**

Teacher/Staff

**LEVEL-1**

Fig 3.21 Searching Operation

**3.5.3.1 Linear Search**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.22 Linear Search

**3.5.3.1.1 Input**

Array of numbers, element to be searched.

**3.5.3.1.2 Process**

This searching technique takes the array of numbers and searches the element linearly if found displays the position of element found.

**3.5.3.1.3 Output**

Displays the searched element in the bar graph with different colour and also displays the position of the element found, time and space complexity.

**3.5.3.1.4 Interface with other functional component**

Not applicable

**3.5.3.1.5 Resource allocation**

style.css file.

**3.5.3.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3.2 Binary search**

**LEVEL-2**

Teacher/ student

FALSE

TRUE

Fig 3.23 Binary Search

**3.5.3.2.1 Input**

Array of numbers, key element

**3.5.3.2.1 Process**

This technique initially sorts the given array if searched element is greater than middle element then it searches at right side otherwise left.

**3.5.3.2.3 Output**

Displays the searched element in the bar graph with different colour and also displays the position of the element found, time and space complexity.

**3.5.3.2.4 Interface with other functional component**

Not applicable

**3.5.3.2.5 Resource allocation**

style.css file.

**3.5.3.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3.3 Exponential search**

**LEVEL 2**

Teacher/ student

FALSE

TRUE

Fig 3.24 Exponential Search

**3.5.3.3.1 Input**

Array of numbers, key element

**3.5.3.3.1 Process**

It searches for an element in a sorted array by jumping 2^I elements every iteration where I represents the value of loop control variable, and then verifying if search element is present between last jump and current jump.

**3.5.3.3.3 Output**

Displays the searched element along the sorted bar graph also the time and space complexity.

**3.5.3.3.4 Interface with other functional component**

Not applicable

**3.5.3.3.5 Resource allocation**

style.css file.

**3.5.3.3.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3.4 Interpolation search**

**LEVEL 2**

Teacher/ student

FALSE

TRUE

Fig 3.25 Interpolation Search

**3.5.3.4.1 Input**

Array of numbers, key element

**3.5.3.4.2 Process**

Interpolation search is an algorithm for searching for a key in an array that has been ordered by numerical values assigned to the keys (key values).

**3.5.3.4.3 Output**

Displays the searched element along the sorted bar graph also the time and space complexity.

**3.5.3.4.4 Interface with other functional component**

Not applicable

**3.5.3.4.5 Resource allocation**

style.css file.

**3.5.3.4.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3.5 Jump search**

**LEVEL 2**

Teacher/ student

FALSE

TRUE

Fig 3.26 Jump Search

**3.5.3.5.1 Input**

Array of numbers, key element

**3.5.3.5.2 Process**

Jump search technique also works for ordered lists. It creates a block and tries to find the element in that block. If the item is not in the block, it shifts the entire block. The block size is based on the size of the list. If the size of the list is n then block size will be √n.

**3.5.3.5.3 Output**

Displays the searched element along the sorted bar graph also the time and space complexity.

**3.5.3.5.4 Interface with other functional component**

Not applicable

**3.5.3.5.5 Resource allocation**

style.css file.

**3.5.3.5.6 User interface**

Buttons, labels, textbox, message box.

**3.5.3.6 Ternary search**

**LEVEL 2**

Teacher/ student

FALSE

TRUE

Fig 3.27 Ternary Search

**3.5.3.6.1 Input**

Array of numbers, key element

**3.5.3.6.2 Process**

Here the array elements are divided into 3 groups within the key element is searched.

**3.5.3.6.3 Output**

Displays the searched element along the sorted bar graph also the time and space complexity.

**3.5.3.6.4 Interface with other functional component**

Not applicable

**3.5.3.6.5 Resource allocation**

style.css file.

**3.5.3.6.6 User interface**

Buttons, labels, textbox, message box.

**3.5.4 Stack**

**LEVEL 1**

Teacher/Staff

Fig 3.28 Stack Operation

**3.5.4.1 Array Implementation**

**LEVEL 2**

Teacher/Staff

Fig 3.29 Array Implementation

**3.5.4.1.1 Push**

**LEVEL 3**

Teacher/ student

FALSE

TRUE

Fig 3.30 Push Using Array Implementation

**3.5.4.1.1.1 Input**

Number

**3.5.4.1.1.2 Processing**

It stores the number at the beginning of the container and checks whether the stack is full.

**3.5.4.1.1.3 Output**

Enters the number within the container also calculates time and space complexity.

**3.5.4.1.1.4 Interface with other functional component**

Not applicable.

**3.5.4.1.1.5 Resource allocation**

style.css file.

**3.5.4.1.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.4.1.2 Pop**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.31 Pop Using Array Implementation

**3.5.4.1.2.1 Input**

Number.

**3.5.4.1.2.2 Processing**

It removes the last entered number and checks whether the stack empty.

**3.5.4.1.2.3 Output**

Removes the number from the container also calculates time and space complexity.

**3.5.4.1.2.4 Interface with other functional component**

Not applicable.

**3.5.4.1.2.5 Resource allocation**

style.css file.

**3.5.4.1.2. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.4.1.2 Linked list Implementation**

**LEVEL 2**

Teacher/Staff

Fig 3.32 Linked List Implementation

**3.5.4.2.1 Push**

**LEVEL 3**

Teacher/ student

FALSE

TRUE

Figs 3.33 Push Using Linked List

**3.5.4.2.1.1 Input**

Number.

**3.5.4.2.1.2 Processing**

It creates the node first and allocates memory if the list is empty then the node is pushed as the first element of the list otherwise link the previous element to the address field of new node.

**3.5.4.2.1.3 Output**

Pushes the element to the list also calculates time and space complexity.

**3.5.4.2.1.4 Interface with other functional component**

Not applicable.

**3.5.4.2.1.5 Resource allocation**

style.css file.

**3.5.4.2.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.4.2.2 Pop**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Figs 3.34 Pop Using Linked List

**3.5.4.2.2.1 Input**

Number.

**3.5.4.2.2.2 Processing**

Node is removed from the end of the linked list. Therefore, must delete the value stored in the head pointer. Then the following node will become the head node.

**3.5.4.2.2.3 Output**

Removes the element to the list also calculates time and space complexity.

**3.5.4.2.2.4 Interface with other functional component**

Not applicable.

**3.5.4.2.2.5 Resource allocation**

style.css file.

**3.5.4.2.2. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.5 Queue Module**

**LEVEL-1**

Teacher/Student

Fig 3.35 Queue Operation

**3.5.5.1 Array implementation**

**LEVEL 2**

Teacher/Staff

Fig 3.36 Queue Array Implementation

**3.5.5.1.1 Insertion**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.37 Insertion of Queue

**3.5.5.1.1.1 Input**

Number

**3.5.5.1.1 .2 Process**

It stores the number at the beginning of the container and checks whether the queue is full.

**3.5.5.1.1.3 Output**

Enters the number within the container also calculates time and space complexity.

**3.5.5.1.1.4 Interface with other functional component**

Not applicable.

**3.5.5.1.1.5 Resource allocation**

style.css file.

**3.5.5.1.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.5.1.2 Deletion**

Teacher/ student

TRUE

FALSE

**LEVEL-2**

Fig 3.38 Deletion of Queue

**3.5.5.1.2.1 Input**

Number

**3.5.5.1.2.2 Process**

It removes the first entered number and checks whether the queue empty.

**3.5.5.1.2.3 Output**

Removes the number from the container also calculates time and space complexity.

**3.5.5.1.2.4 Interface with other functional component**

Not applicable.

**3.5.5.1.2.5 Resource allocation**

style.css file.

**3.5.5.1.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.5.2 Linked List Implementation**

**LEVEL 2**

Teacher/Staff

Fig 3.39 Queue LinkedList Implementation

**3.5.5.2.1 Insertion**

Teacher/ Student

TRUE

FALSE

**LEVEL 3**

Fig 3.40 Queue Insertion

**3.5.5.2.1.1 Input**

Number

**3.5.5.2.1.2 Process**

It adds an element to the end of the queue. The new element becomes the last element of the queue.

**3.5.5.2.1.3 Output**

Enters the element to the list also calculates time and space complexity.

**3.5.5.2.1.4 Interface with other functional component**

Not applicable.

**3.5.5.2.1.5 Resource allocation**

style.css file.

**3.5.5.2.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.5.2.2 Deletion**

Teacher/ Student

FALSE

TRUE

**LEVEL-2**

Fig 3.41 Queue Deletion

**3.5.5.2.2.1 Input**

Number

**3.5.5.2.2.2 Process**

Removes the element which was first inserted in the queue.

**3.5.5.2.2.3 Output**

Removes the element from the list also calculates time and space complexity.

**3.5.5.2.2.4 Interface with other functional component**

Not applicable.

**3.5.5.2.2.5 Resource allocation**

style.css file.

**3.5.5.2.2. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.5.3 Circular Queue**

**LEVEL 2**

Teacher/Staff

Fig 3.41 Circular Queue

**3.5.5.3.1 Insertion**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.42 Circular Queue Insertion

**3.5.5.3.1.1 Input**

Number

**3.5.5.3.1.2 Process**

It stores the number at the beginning of the circle and checks whether the circular queue is full.

**3.5.5.3.1.3 Output**

Enters the number in the circle also calculates time and space complexity.

**3.5.5.3.1.4 Interface with other functional component**

Not applicable.

**3.5.5.3.1.5 Resource allocation**

style.css file.

**3.5.5.3.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.5.3.2 Deletion**

Teacher/ student

FALSE

TRUE

**LEVEL-2**

Fig 3.43 Circular Queue Deletion

**3.5.5.3.2.1 Input**

Number

**3.5.5.3.2.2 Process**

It removes the first entered number and checks whether the circular queue empty.

**3.5.5.3.2.3 Output**

Removes the number in the circle also calculates time and space complexity.

**3.5.4.1.1.4 Interface with other functional component**

Not applicable.

**3.5.4.1.1. 5 Resource allocation**

style.css file.

**3.5.4.1.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6 LinkedList**

**LEVEL 1**

Teacher/Student

Fig 3.44 Linked List

**3.5.6.1 Singly Linked list**

**LEVEL 2**

Teacher/Student

Fig 3.45 Singly LinkedList Operation

**3.5.6.1.1 Insertion at beginning**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.46 Insertion at Beginning

**3.5.6.1.1.1 Input**

Node elements

**3.5.6.1.1.2 Process**

Insert the node at the beginning of the Singly linked list.

**3.5.6.1.1.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.1.4 Interface with other functional component**

Not applicable.

**3.5.6.1.1. 5 Resource allocation**

style.css file.

**3.5.6.1.1. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.2 Insertion at end**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.47 Insertion at End

**3.5.6.1.2.1 Input**

Node elements

**3.5.6.1.2.2 Process**

Insert the node at the end of the Singly linked list.

**3.5.6.1.2.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.2.4 Interface with other functional component**

Not applicable.

**3.5.6.1.2.5 Resource allocation**

style.css file.

**3.5.6.1.2. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.3 Insertion at position**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.48 Insertion at Position

**3.5.6.1.3.1 Input**

Node elements

**3.5.6.1.3.2 Process**

Insert the node at the specific position of the Singly linked list.

**3.5.6.1.3.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.3.4 Interface with other functional component**

Not applicable.

**3.5.6.1.3.5 Resource allocation**

style.css file.

**3.5.6.1.3. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.4 Deletion at beginning**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.49 Deletion at Beginning

**3.5.6.1.4.1 Input**

Node elements

**3.5.6.1.4.2 Process**

Delete the node at the beginning of the Singly linked list.

**3.5.6.1.4.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.4.4 Interface with other functional component**

Not applicable.

**3.5.6.1.4.5 Resource allocation**

style.css file.

**3.5.6.1.4. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.5 Deletion at end**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.50 Deletion at The End

**3.5.6.1.5.1 Input**

Node elements

**3.5.6.1.5.2 Process**

Delete the node at the end of the Singly linked list.

**3.5.6.1.5.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.5.4 Interface with other functional component** Not applicable.

**3.5.6.1.5.5 Resource allocation**

style.css file.

**3.5.6.1.5. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.6 Deletion at position**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.51 Deletion at Position

**3.5.6.1.6.1 Input**

Node elements

**3.5.6.1.6.2 Process**

Delete the node at specific location of the Singly linked list.

**3.5.6.1.6.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.6.4 Interface with other functional component**

Not applicable.

**3.5.6.1.6.5 Resource allocation**

style.css file.

**3.5.6.1.6.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.1.7 Deletion on element**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.52 Deletion on Element

**3.5.6.1.7.1 Input**

Element

**3.5.6.1.7.2 Process**

Delete the specific element of the Singly linked list.

**3.5.6.1.7.3 Output**

Displays the Singly linked list with time and space complexity.

**3.5.6.1.7.4 Interface with other functional component** Not applicable.

**3.5.6.1.7.5 Resource allocation**

style.css file.

**3.5.6.1.7. 6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2 Doubly Linked list**

**LEVEL 2**

Teacher/Student

Fig 3.53 Doubly Linked List

**3.5.6.2.1 Insertion at beginning**

**LEVEL 3**

Teacher/ student

TRUE

FALSE

Fig 3.54 Insertion at Beginning

**3.5.6.2.1.1 Input**

Node elements

**3.5.6.2.1.2 Process**

Insert the node at the beginning of the Doubly linked list.

**3.5.6.2.1.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.1.4 Interface with other functional component**

Not applicable.

**3.5.6.2.1.5 Resource allocation**

style.css file.

**3.5.6.2.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2.2 Insertion at end**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.55 Insertion at End

**3.5.6.2.2.1 Input**

Node elements

**3.5.6.2.2.2 Process**

Insert the node at the end of the Doubly linked list.

**3.5.6.2.2.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.2.4 Interface with other functional component**

Not applicable.

**3.5.6.2.2.5 Resource allocation**

style.css file.

**3.5.6.2.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2.3 Insertion at position**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.56 Insertion at Position

**3.5.6.2.3.1 Input**

Node elements

**3.5.6.2.3.2 Process**

Insert the node at the specific position of the Doubly linked list.

**3.5.6.2.3.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.3.4 Interface with other functional component**

Not applicable.

**3.5.6.2.3.5 Resource allocation**

style.css file.

**3.5.6.2.3.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2.4 Deletion at beginning**

**LEVEL 3**

Teacher/ student

TRUE

FALSE

Fig 3.57 Deletion at Beginning

**3.5.6.2.4.1 Input**

Node elements

**3.5.6.2.4.2 Process**

Delete the node at the beginning of the Doubly linked list.

**3.5.6.2.4.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.4.4 Interface with other functional component**

Not applicable.

**3.5.6.2.4. 5 Resource allocation**

style.css file.

**3.5.6.2.4.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2.5 Deletion at end**

**LEVEL 3**

Teacher/ student

TRUE

FALSE

Fig 3.58 Deletion at End

**3.5.6.2.5.1 Input**

Node elements

**3.5.6.2.5.2 Process**

Delete the node at the end of the Doubly linked list.

**3.5.6.2.5.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.5.4 Interface with other functional component**

Not applicable.

**3.5.6.2.5.5 Resource allocation**

style.css file.

**3.5.6.2.5.6 User interface**

Buttons, labels, textbox, message box.

* + - * 1. **Deletion at position**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.59 Deletion at Position

**3.5.6.2.6.1 Input**

Node elements

**3.5.6.2.6.2 Process**

Delete the node at specific location of the Doubly linked list.

**3.5.6.2.6.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.6.4 Interface with other functional component**

Not applicable.

**3.5.6.2.6. 5 Resource allocation**

style.css file.

**3.5.6.2.6.6 User interface**

Buttons, labels, textbox, message box.

**3.5.6.2.7 Deletion at element**

**LEVEL 3**

Teacher/ student

TRUE

FALSE

Fig 3.60 Deletion at Element

**3.5.6.2.7.1 Input**

Node elements

**3.5.6.2.7.2 Process**

Delete the specific element of the Doubly linked list.

**3.5.6.2.7.3 Output**

Displays the Doubly linked list with time and space complexity.

**3.5.6.2.7.4 Interface with other functional component**

Not applicable.

**3.5.6.2.7. 5 Resource allocation**

style.css file.

**3.5.6.2.7.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7 Tree**

**LEVEL 1**

Teachers /student

Fig 3.76 Tree

**3.5.7.1 Binary Search Tree**

Teacher/Student

**LEVEL 2**

Fig 3.77 Binary Search Tree

**3.5.7.1.1 Insertion**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.78 Insertion

**3.5.7.1.1.1 Input**

Number

**3.5.7.1.1.2 Process**

This technique is used to add new element to the binary search tree in appropriate location.

**3.5.7.1.1.3 Output**

Displays the binary tree including the new element also calculates the time and space complexity.

**3.5.7.1.1.4 Interface with other functional component**

Not applicable.

**3.5.7.1.1.5 Resource allocation**

style.css file.

**3.5.7.1.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7.1.2 Deletion**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.79 Insertion

**3.5.7.1.2.1 Input**

Number

**3.5.7.1.2.2 Process**

This technique will delete the specified from a binary search tree.

**3.5.7.1.2.3 Output**

Displays the binary tree excluding the removed element also calculates the time and space complexity.

**3.5.7.1.2.4 Interface with other functional component**

Not applicable.

**3.5.7.1.2.5 Resource allocation**

style.css file.

**3.5.7.1.2.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7.1.3 Searching**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.80 Searching

**3.5.7.1.3.1 Input**

key element

**3.5.7.1.3.2 Process**

This technique is used for locating a key from within a set.

**3.5.7.1.3.3 Output**

Displays the binary tree with searched element position also calculates the time and space complexity.

**3.5.7.1.3.4 Interface with other functional component**

Not applicable.

**3.5.7.1.3.5 Resource allocation**

style.css file.

**3.5.7.1.3.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7.1.4 Pre-order**

Teacher/ student

FALSE

TRUE

**LEVEL 3**

Fig 3.81 Pre-Order

**Input**

List of numbers.

**3.5.7.1.4.2 Process**

This technique that traverses from root to the left subtree then to the right sub tree.

**3.5.7.1.4.3 Output**

Displays the list of traversed order.

**3.5.7.1.4.4 Interface with other functional component**

Not applicable.

**3.5.7.1.4.5 Resource allocation**

style.css file.

**3.5.7.1.4.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7.1.5 Post-order**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.82 Post-Order

**3.5.7.1.5.1 Input**

List of numbers.

**3.5.7.1.5.2 Process**

This technique that traverses from left subtree to the right subtree then to the root.

**3.5.7.1.5.3 Output**

Displays the list of traversed order.

**3.5.7.1.5.4 Interface with other functional component**

Not applicable.

**3.5.7.1.5.5 Resource allocation**

style.css file.

**3.5.7.1.5.6 User interface**

Buttons, labels, textbox, message box.

**3.5.7.1.6 In-order**

Teacher/ student

TRUE

FALSE

**LEVEL 3**

Fig 3.83 In-Order

**3.5.7.1.6.1 Input**

List of numbers.

**3.5.7.1.6.2 Process**

This technique that traverses from left subtree to the root then to the right subtree.

**3.5.7.1.6.3 Output**

Displays the list of traversed order.

**3.5.7.1.6.4 Interface with other functional component**

Not applicable.

**3.5.7.1.6.5 Resource allocation**

style.css file.

**3.5.7.1.6.6 User interface**

Buttons, labels, textbox, message box.

**3.5.8 Graph**

**LEVEL 1**

Teacher/Staff

Fig 3.84 Graph

**3.5.8.1 BFS**

Teacher/ student

TRUE

FALSE

**LEVEL 2**

Fig 3.85 BFS

**3.5.8.1.1 Input**

Enter Source and Destination Node.

**3.5.8.1.2 Process**

BFS or Breadth-First-Search starts from the top node in the graph and travels down until it reached the root node.

**3.5.8.1.3 Output**

It will display the complete from Source to Destination Node in given tree.

**3.5.8.1.4 Interface with other functional component**

Not applicable.

**3.5.8.1.5 Resource allocation**

style.css file.

**3.5.8.1.6 User interface**

Buttons, labels, textbox, message box.

**3.5.8.2 DFS**

Teacher/ student

TRUE

FALSE

**LEVEL 2**

Fig 3.86 DFS

**3.5.8.2.1 Input**

Enter Source and Destination Node.

**3.5.8.2.2 Process**

DFS or Depth-First-Search starts from the top node and follows the path to reaches the end node of the path.

**3.5.8.2.3 Output**

It will display the complete from Source to Destination Node in given tree

**3.5.8.2.4 Interface with other functional component**

Not applicable.

**3.5.8.2.5 Resource allocation**

style.css file.

**3.5.8.2.6 User interface**

Buttons, labels, textbox, message box.