**DECLARATION**

We hereby declare that this project work entitled “PERFORMANACE ANALYSIS AND SIMULATION OF DATA STRUCTURE ALGORITHMS” has been prepared by us during the year 2022-23 under the guidance of Mr. GANESH K, Lecturer, Department of Computer Science, Bhandarkar’s’ Arts and Science College, Kundapura in the partial fulfilment of BCA degree prescribed by Mangalore University.

We also declare that this project is the outcome of our own effort, that it has not been submitted to any other university for the award of any degree.

Date:

SANJANA S : 201231522136

SAPTHAMI CHATRA : 201231522226

PREETI JANARDHAN NAIK : 201231522213

**ACKNOWLEDGEMENT**

It gives us an immense pleasure to present the project report on **“PERFORMANCE ANALYSIS AND SIMULATION OF DATA STRUCTURE ALGORITHMS”**

Our sincere thanks to **Mr. Ganesh. K,** Lecturer, Department of Computer Science our project guide who helped us in developing this project.

We would like to express our gratitude to **Mrs. Vijayalakshmi N Shetty**, **Head of the Department, Computer Science** for her kind concern and encouragement during the completion of our project.

We are sincerely thankful to **Dr. N. P. Narayan Shetty**, Principal of Bhandarkar’s Arts and Science College Kundapura, for granting an opportunity to work our project.

Our sincere thanks for all faculty members of Computer Science Department. We are thankful to our parents for their encouragement towards the project. Last but not the least, we whole heartly appreciates the co-operation of our friends.

**Thank You,**

Sapthami Chatra

Preeti Janardhan Naik

Sanjana S

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SNO** | **TITLE** | **PAGE NO** |
| **1.** | **INTRODUCTION** | 1 |
| 1.1 Introduction of the system | 1 |
| 1.1.1 Project title | 1 |
| 1.1.2 Category | 1 |
| 1.1.3 Overview | 1 |
| 1.2 Background | 1 |
| 1.2.1 Introduction of the company | 1 |
| 1.2.2 Brief note on existing system | 1 |
| 1.3 Objective of the system | 2 |
| 1.4 Scope of the system | 2 |
| 1.5 Structure of the system | 2 |
| 1.5.1 Analysis | 2 |
| 1.5.2 Module Description | 2 |
| 1.5.2.1 Sorting Module | 2 |
| 1.5.2.1.1 Bubble sort | 2-3 |
| 1.5.2.1.2 Selection sort | 3 |
| 1.5.2.1.3 Insertion sort | 3 |
| 1.5.2.1.4 Exchange sort | 3 |
| 1.5.2.1.5 Counting sort | 3 |
| 1.5.2.1.6 Heap sort | 3 |
| 1.5.2.1.7 Bucket sort | 3 |
| 1.5.2.1.8 Merge sort | 3 |
| 1.5.2.1.9 Quick sort | 3-4 |
| 1.5.2.1.10 Brick sort | 4 |
| 1.5.2.1.11 Shell sort | 4 |
| 1.5.2.2 Array Operation Module | 4 |
| 1.5.2.2.1 Insertion | 4 |
| 1.5.2.2.2 Deletion | 4 |
| 1.5.2.3 Searching Module | 4-5 |
| 1.5.2.3.1 Linear search | 5 |
| 1.5.2.3.2 Binary search | 5 |
| 1.5.2.3.3 Exponential search | 5 |
| 1.5.2.3.4 Interpolation search | 5 |
| 1.5.2.3.5 Jump search | 5 |
| 1.5.2.3.6 Ternary search | 5 |
| 1.5.2.4 Stack Module | 5-6 |
| 1.5.2.4.1 Array implementation | 6 |
| 1.5.2.4.1.1 Push | 6 |
| 1.5.2.4.1.2 Pop | 6 |
| 1.5.2.4.2 Linked list implementation | 6 |
| 1.5.2.4.2.1 Push | 6 |
| 1.5.2.4.2.2 Pop | 6 |
| 1.5.2.5 Queue Module | 6 |
| 1.5.2.5.1 Array implementation | 6 |
| 1.5.2.5.1.1 Insertion | 6 |
| 1.5.2.5.1.2 Deletion | 6-7 |
| 1.5.2.5.2 Linked list implementation | 7 |
| 1.5.2.5.2.1 Insertion | 7 |
| 1.5.2.5.2.2 Deletion | 7 |
| 1.5.2.5.3 Circular queue | 7 |
| 1.5.2.5.3.1 Insertion | 7 |
| 1.5.2.5.3.2 Deletion | 7 |
| 1.5.2.6 Linked list Module | 7 |
| 1.5.2.6.1 Singly linked list | 7 |
| 1.5.2.6.1.1 Insertion at beginning | 7 |
| 1.5.2.6.1.2 Insertion at end | 7 |
| 1.5.2.6.1.3 Insertion at position | 8 |
| 1.5.2.6.1.4 Deletion at beginning | 8 |
| 1.5.2.6.1.5 Deletion at end | 8 |
| 1.5.2.6.1.6 Deletion at position | 8 |
| 1.5.2.6.1.7 Deletion on position | 8 |
| 1.5.2.6.2 Doubly linked list | 8 |
| 1.5.2.6.2.1 Insertion at beginning | 8 |
| 1.5.2.6.2.2 Insertion at end | 8 |
| 1.5.2.6.2.3 Insertion at position | 8 |
| 1.5.2.6.2.4 Deletion at beginning | 8-9 |
| 1.5.2.6.2.5 Deletion at end | 9 |
| 1.5.2.6.2.6 Deletion at position | 9 |
| 1.5.2.6.2.7 Deletion on position | 9 |
| 1.5.2.7 Tree Module | 9 |
| 1.5.2.7.1 Insertion | 9 |
| 1.5.2.7.2 Deletion | 9 |
| 1.5.2.7.3 Searching | 9 |
| 1.5.2.7.4 Pre-order | 9 |
| 1.5.2.7.5 Post-order | 9 |
| 1.5.2.7.6 In-order | 9-10 |
| 1.5.2.8 Graph Module | 10 |
| 1.5.2.8.1 BFS | 10 |
| 1.5.2.8.2 DFS | 10 |
| 1.6 System architecture | 11 |
| 1.7 End users | 11 |
| 1.8 Software/Hardware used for development | 11 |
| 1.8.1 Software | 11-12 |
| 1.8.2 Hardware | 12 |
| 1.9 Software/Hardware required for implementation | 12 |
| 1.9.1 Software | 12 |
| 1.9.2 Hardware | 12 |
| **2.** | **SOFTWARE REQUIRMENT SPECIFICATION** | 13 |
| 2.1 Introduction | 13 |
| 2.2 Overall Description | 13 |
| 2.2.1 Product perspective | 13 |
| 2.2.1.1 System interface | 13 |
| 2.2.1.2 User interface | 13 |
| 2.2.1.3 Hardware interface | 13 |
| 2.2.1.4 Software interface | 13-14 |
| 2.2.1.5 Communication interface | 14 |
| 2.2.1.6 Interface with Servers | 14 |
| 2.2.2 Product function | 14 |
| 2.2.3 User characteristics | 14 |
| 2.2.4 General constraints | 14-15 |
| 2.2.5 Assumption and Dependencies | 15 |
| 2.3 Special requirements | 15 |
| 2.4 Functional requirements | 15 |
| 2.4.1 Sorting module | 15 |
| 2.4.1.1 Bubble sort | 15 |
| 2.4.1.2 Selection sort | 15-16 |
| 2.4.1.3 Insertion sort | 16 |
| 2.4.1.4 Exchange sort | 16 |
| 2.4.1.5 Counting sort | 16 |
| 2.4.1.6 Heap sort | 16-17 |
| 2.4.1.7 Bucket sort | 17 |
| 2.4.1.8 Merge sort | 17 |
| 2.4.1.9 Quick sort | 17 |
| 2.4.1.10 Brick sort | 17-18 |
| 2.4.1.11 Shell sort | 18 |
| 2.4.2 Array Operation Module | 18 |
| 2.4.2.1 Insertion | 18 |
| 2.4.2.2 Deletion | 18 |
| 2.4.3 Searching Module | 19 |
| 2.4.3.1 Linear search | 19 |
| 2.4.3.2 Binary search | 19 |
| 2.4.3.3 Exponential search | 19 |
| 2.4.3.4 Interpolation search | 19-20 |
| 2.4.3.5 Jump search | 20 |
| 2.4.3.6 Ternary search | 20 |
| 2.4.4 Stack Module | 20 |
| 2.4.4.1 Array implementation | 20 |
| 2.4.4.1.1 Push | 20 |
| 2.4.4.1.2 Pop | 20-21 |
| 2.4.4.2 Linked list implementation | 21 |
| 2.4.4.2.1 Push | 21 |
| 2.4.4.2.2 Pop | 21 |
| 2.4.5 Queue Module | 21 |
| 2.4.5.1 Array implementation | 21 |
| 2.4.5.1.1 Insertion | 21-22 |
| 2.4.5.1.2 Deletion | 22 |
| 2.4.5.2 Linked list implementation | 22 |
| 2.4.5.2.1 Insertion | 22 |
| 2.4.5.2.2 Deletion | 22 |
| 2.4.5.3 Circular queue | 22 |
| 2.4.5.3.1 Insertion | 22-23 |
| 2.4.5.3.2 Deletion | 23 |
| 2.4.6 Linked list Module | 23 |
| 2.4.6.1 Singly linked list | 23 |
| 2.4.6.1.1 Insertion at beginning | 23 |
| 2.4.6.1.2 Insertion at end | 23 |
| 2.4.6.1.3 Insertion at position | 23-24 |
| 2.4.6.1.4 Deletion at beginning | 24 |
| 2.4.6.1.5 Deletion at end | 24 |
| 2.4.6.1.6 Deletion at position | 24 |
| 2.4.6.1.7 Deletion on element | 24 |
| 2.4.6.2 Doubly linked list | 24 |
| 2.4.6.2.1 Insertion at beginning | 24 |
| 2.4.6.2.2 Insertion at end | 24-25 |
| 2.4.6.2.3 Insertion at position | 25 |
| 2.4.6.2.4 Deletion at beginning | 25 |
| 2.4.6.2.5 Deletion at end | 25 |
| 2.4.6.2.6 Deletion at position | 25 |
| 2.4.6.2.7 Deletion on element | 25 |
| 2.4.7 Tree Module | 25 |
| 2.47.1 Binary search tree | 25 |
| 2.4.7.1.1 Insertion | 26 |
| 2.4.7.1.2 Deletion | 26 |
| 2.4.7.1.3 Searching | 26 |
| 2.4.7.1.4 Pre-order | 26-27 |
| 2.4.7.1.5 Post-order | 27 |
| 2.4.7.1.6 In-order | 27 |
| 2.4.8 Graph Module | 27 |
| 2.4.8.1 BFS | 27 |
| 2.4.8.2 DFS | 27 |
| 2.5 Design constraints | 27 |
| 2.5.1 Hardware constraints | 27 |
| 2.5.2 Software constraints | 27 |
| 2.5.3 Fault tolerance | 27-28 |
| 2.5.4 Security | 28 |
| 2.5.5 Standard compliance | 28 |
| 2.6 System attributes | 28 |
| 2.7 Other requirements | 28 |
| **23** | **SYSTEM DESIGN** | 29 |
| 3.1 Introduction | 29 |
| 3.2 Assumptions and Constraints | 29 |
| 3.3 Functional decomposition | 29 |
| 3.3.1 System software Architecture | 30-34 |
| 3.3.2 System technical architecture | 35 |
| 3.3.3 System hardware architecture | 35 |
| 3.3.4 External interface | 35 |
| 3.4 Description of program | 35 |
| 3.4.1 Context flow diagram | 35-36 |
| 3.4.2 Data flow diagram | 37-38 |
| 3.5 Description of the component | 39 |
| 3.5.1 Sorting module | 39 |
| 3.5.1.1 Bubble sort | 39-40 |
| 3.5.1.2 Selection sort | 40-41 |
| 3.5.1.3 Insertion sort | 42-43 |
| 3.5.1.4 Exchange sort | 43-44 |
| 3.5.1.5 Counting sort | 44-45 |
| 3.5.1.6 Heap sort | 45-46 |
| 3.5.1.7 Bucket sort | 47-48 |
| 3.5.1.8 Merge sort | 48-49 |
| 3.5.1.9 Quick sort | 49-50 |
| 3.5.1.10 Brick sort | 50-51 |
| 3.5.1.11 Shell sort | 51-52 |
| 3.5.2 Array Operation Module | 52 |
| 3.5.2.1 Insertion | 53-54 |
| 3.5.2.2 Deletion | 54-55 |
| 3.5.3 Searching Module | 55 |
| 3.5.3.1 Linear search | 55-56 |
| 3.5.3.2 Binary search | 56-57 |
| 3.5.3.3 Exponential search | 57-58 |
| 3.5.3.4 Interpolation search | 58-59 |
| 3.5.3.5 Jump search | 59-60 |
| 3.5.3.6 Ternary search | 60-61 |
| 3.5.4 Stack Module | 61 |
| 3.5.4.1 Array implementation | 61 |
| 3.5.4.1.1 Push | 62-63 |
| 3.5.4.1.2 Pop | 63-64 |
| 3.5.4.2 Linked list implementation | 64 |
| 3.5.4.2.1 Push | 64-65 |
| 3.5.4.2.2 Pop | 65-66 |
| 3.5.5 Queue Module | 66 |
| 3.5.5.1 Array implementation | 67 |
| 3.5.5.1.1 Insertion | 67-68 |
| 3.5.5.1.2 Deletion | 68-69 |
| 3.5.5.2 Linked list implementation | 69 |
| 3.5.5.2.1 Insertion | 69-70 |
| 3.5.5.2.2 Deletion | 70-71 |
| 3.5.5.3 Circular queue | 71 |
| 3.5.5.3.1 Insertion | 72-73 |
| 3.5.5.3.2 Deletion | 73-74 |
| 3.5.6 Linked list Module | 74 |
| 3.5.6.1 Singly linked list | 74 |
| 3.5.6.1.1 Insertion at beginning | 75-76 |
| 3.5.6.1.2 Insertion at end | 76-77 |
| 3.5.6.1.3 Insertion at position | 77-78 |
| 3.5.6.1.4 Deletion at beginning | 78-79 |
| 3.5.6.1.5 Deletion at end | 79-80 |
| 3.5.6.1.6 Deletion at position | 80-81 |
| 3.5.6.1.7 Deletion on element | 81-82 |
| 3.5.6.2 Doubly linked list | 82 |
| 3.5.6.2.1 Insertion at beginning | 83-84 |
| 3.5.6.2.2 Insertion at end | 84-85 |
| 3.5.6.2.3 Insertion at position | 85-86 |
| 3.5.6.2.4 Deletion at beginning | 86-87 |
| 3.5.6.2.5 Deletion at end | 87-88 |
| 3.5.6.2.6 Deletion at position | 88-89 |
| 3.5.6.2.7 Deletion on element | 89-90 |
| 3.5.7 Tree Module | 90 |
| 3.5.7.1 Binary search tree | 90 |
| 3.5.7.1.1 Insertion | 91 |
| 3.5.7.1.2 Deletion | 92 |
| 3.5.7.1.3 Searching | 93 |
| 3.5.7.1.4 Pre-order | 94 |
| 3.5.7.1.5 Post-order | 95 |
| 3.5.7.1.6 In-order | 96 |
| 3.5.8 Graph Module | 96 |
| 3.5.8.1 BFS | 97-98 |
| 3.5.8.2 DFS | 98-99 |
| 4 | **DETAILED DESIGN** | 100 |
|  | 4.1 Introduction | 100 |
|  | 4.2 Structure of software package | 100-103 |
|  | 4.3 Module decomposition of software | 104-105 |
|  | 4.3.1 Sorting module | 106 |
|  | 4.3.1.1 Bubble sort | 106-107 |
|  | 4.3.1.2 Selection sort | 107-109 |
|  | 4.3.1.3 Insertion sort | 109-111 |
|  | 4.3.1.4 Exchange sort | 111-113 |
|  | 4.3.1.5 Counting sort | 113-115 |
|  | 4.3.1.6 Heap sort | 115-118 |
|  | 4.3.1.7 Bucket sort | 118-120 |
|  | 4.3.1.8 Merge sort | 120-122 |
|  | 4.3.1.9 Quick sort | 122-125 |
|  | 4.3.1.10 Brick sort | 125-127 |
|  | 4.3.1.11 Shell sort | 127-129 |
|  | 4.3.2 Array Operation Module | 129 |
|  | 4.3.2.1 Insertion | 129-130 |
|  | 4.3.2.2 Deletion | 130-131 |
|  | 4.3.3 Searching Module | 131 |
|  | 4.3.3.1 Linear search | 131-132 |
|  | 4.3.3.2 Binary search | 132-133 |
|  | 4.3.3.3 Exponential search | 133 |
|  | 4.3.3.4 Interpolation search | 134 |
|  | 4.3.3.5 Jump search | 134-135 |
|  | 4.3.3.6 Ternary search | 135-136 |
|  | 4.3.4 Stack Module | 136 |
|  | 4.3.4.1 Array implementation | 136 |
|  | 4.3.4.1.1 Push | 136-137 |
|  | 4.3.4.1.2 Pop | 137-138 |
|  | 4.3.4.2 Linked list implementation | 138 |
|  | 4.3.4.2.1 Push | 138-139 |
|  | 4.3.4.2.2 Pop | 140-141 |
|  | 4.3.5 Queue Module | 141 |
|  | 4.3.5.1 Array implementation | 141 |
|  | 4.3.5.1.1 Insertion | 141-142 |
|  | 4.3.5.1.2 Deletion | 142-143 |
|  | 4.3.5.2 Linked list implementation | 143 |
|  | 4.3.5.2.1 Insertion | 143-145 |
|  | 4.3.5.2.2 Deletion | 145-146 |
|  | 4.3.5.3 Circular queue | 146 |
|  | 4.3.5.3.1 Insertion | 146-147 |
|  | 4.3.5.3.2 Deletion | 147-149 |
|  | 4.3.6 Linked list Module | 149 |
|  | 4.3.6.1 Singly linked list | 149 |
|  | 4.3.6.1.1 Insertion at beginning | 149-150 |
|  | 4.3.6.1.2 Insertion at end | 150-151 |
|  | 4.3.6.1.3 Insertion at position | 151-153 |
|  | 4.3.6.1.4 Deletion at beginning | 153-154 |
|  | 4.3.6.1.5 Deletion at end | 154-156 |
|  | 4.3.6.1.6 Deletion at position | 156-157 |
|  | 4.3.6.1.7 Deletion on element | 157-159 |
|  | 4.3.6.2 Doubly linked list | 159 |
|  | 4.3.6.2.1 Insertion at beginning | 159-160 |
|  | 4.3.6.2.2 Insertion at end | 160-162 |
|  | 4.3.6.2.3 Insertion at position | 162-163 |
|  | 4.3.6.2.4 Deletion at beginning | 163-165 |
|  | 4.3.6.2.5 Deletion at end | 165-166 |
|  | 4.3.6.2.6 Deletion at position | 166-168 |
|  | 4.3.6.2.7 Deletion on element | 168-170 |
|  | 4.3.7 Tree Module | 170 |
|  | 4.3.7.1 Binary search tree | 170 |
|  | 4.3.7.1.1 Insertion | 170 |
|  | 4.3.7.1.2 Deletion | 170-171 |
|  | 4.3.7.1.3 Searching | 171-172 |
|  | 4.3.7.1.4 Pre-order | 172 |
|  | 4.3.7.1.5 Post-order | 172-173 |
|  | 4.3.7.1.6 In-order | 173-174 |
|  | 4.3.8 Graph Module | 174 |
|  | 4.3.8.1 BFS | 174-176 |
|  | 4.3.8.2 DFS | 177-179 |
| **5** | **User Interface** |  |
| **6** | **Testing** |  |
|  | **Conclusion** |  |
|  | **Limitations** |  |
|  | **Future scope** |  |
|  | **Abbreviations and Acronyms** |  |
|  | **Bibliography** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG**  **NO** | **LIST OF FIGURES** | **PAGE NO** |
| 1.1 | System Architecture |  |
| 3.1 | System software architecture | 34 |
| 3.2 | System technical architecture | 35 |
| 3.3 | System hardware architecture | 35 |
| 3.4 | Context flow diagram | 36 |
| 3.5 | DFD for modules (Level 0) | 38 |
| 3.6 | Sorting (Level 1) | 39 |
| 3.7 | Bubble sort | 39 |
| 3.8 | Selection sort | 40 |
| 3.9 | Insertion sort | 42 |
| 3.10 | Exchange sort | 43 |
| 3.11 | Counting sort | 44 |
| 3.12 | Heap sort | 45 |
| 3.13 | Bucket sort | 47 |
| 3.14 | Merge sort | 48 |
| 3.15 | Quick sort | 49 |
| 3.16 | Brick sort | 50 |
| 3.17 | Shell sort | 51 |
| 3.18 | Array Operation (Level 1) | 52 |
| 3.19 | Array Insertion | 53 |
| 3.20 | Array Deletion | 54 |
| 3.21 | Searching Operation (Level 1) | 55 |
| 3.22 | Linear search | 55 |
| 3.23 | Binary search | 56 |
| 3.24 | Exponential search | 57 |
| 3.25 | Interpolation search | 58 |
| 3.26 | Jump search | 59 |
| 3.27 | Ternary search | 60 |
| 3.28 | Stack Operation (Level 1) | 61 |
| 3.29 | Array implementation | 61 |
| 3.30 | Push using array implementation | 62 |
| 3.31 | Pop using array implementation | 63 |
| 3.32 | Linked list implementation | 64 |
| 3.33 | Push using linked list | 64 |
| 3.34 | Pop using linked list | 65 |
| 3.35 | Queue Operation (Level 1) | 66 |
| 3.36 | Queue array operation | 67 |
| 3.37 | Insertion of queue | 67 |
| 3.38 | Deletion of queue | 68 |
| 3.39 | Queue linked list implementation | 69 |
| 3.40 | Queue insertion | 69 |
| 3.41 | Queue deletion | 70 |
| 3.42 | Circular queue | 71 |
| 3.43 | Circular queue insertion | 72 |
| 3.44 | Circular queue deletion | 73 |
| 3.45 | Linked list | 74 |
| 3.46 | Singly linked list operation | 74 |
| 3.47 | Insertion at beginning | 75 |
| 3.48 | Insertion at end | 76 |
| 3.49 | Insertion at position | 77 |
| 3.50 | Deletion at beginning | 78 |
| 3.51 | Deletion at end | 79 |
| 3.52 | Deletion at position | 80 |
| 3.53 | Deletion on element | 81 |
| 3.54 | Doubly linked list operation | 82 |
| 3.55 | Insertion at beginning | 83 |
| 3.56 | Insertion at end | 84 |
| 3.57 | Insertion at position | 85 |
| 3.58 | Deletion at beginning | 86 |
| 3.59 | Deletion at end | 87 |
| 3.60 | Deletion at position | 88 |
| 3.61 | Deletion on element | 89 |
| 3.62 | Tree (Level 1) | 90 |
| 3.63 | Binary search tree | 90 |
| 3.64 | Insertion | 91 |
| 3.65 | Deletion | 92 |
| 3.66 | Searching | 93 |
| 3.67 | Pre-order | 94 |
| 3.68 | Post-order | 95 |
| 3.69 | In-order | 96 |
| 3.70 | Graph (Level 1) | 96 |
| 3.71 | BFS | 97 |
| 3.72 | DFS | 98 |
| 4.1 | Structure of software package | 103 |
| 4.2 | Bubble sort(flowchart) | 107 |
| 4.3 | Selection sort(flowchart) | 109 |
| 4.4 | Insertion sort(flowchart) | 111 |
| 4.5 | Exchange sort(flowchart) | 112 |
| 4.6 | Counting sort(flowchart) | 115 |
| 4.7 | Heap sort(flowchart) | 118 |
| 4.8 | Bucket sort(flowchart) | 120 |
| 4.9 | Merge sort(flowchart) | 122 |
| 4.10 | Quick sort(flowchart) | 124 |
| 4.11 | Brick sort(flowchart) | 126 |
| 4.12 | Shell sort(flowchart) | 128 |
| 4.13 | Linear search(structure chart) | 132 |
| 4.14 | Binary search(structure chart) | 132 |
| 4.15 | Exponential search(structure chart) | 133 |
| 4.16 | Interpolation search(structure chart) | 134 |
| 4.17 | Jump search(structure chart) | 135 |
| 4.18 | Ternary search(structure chart) | 135 |
| 4.19 | Insertion (structure chart) | 170 |
| 4.20 | Deletion (structure chart) | 171 |
| 4.21 | Searching (structure chart) | 171 |
| 4.22 | Pre- order (structure chart) | 172 |
| 4.23 | Post-order (structure chart) | 174 |
| 4.24 | In-order (structure chart) | 174 |
| 4.25 | BFS (flowchart) | 176 |
| 4.26 | DFS (flowchart) | 179 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABEL NO** | **TABLE NAME** | **PAGE NO** |
| 3.1 | DFD Symbols | 37 |
| 4.1 | Structure chart | 104 |
| 4.2 | Flowchart | 105 |

**CONCLUSION**

In conclusion, this project was successfully implemented using JavaScript. It is capable of carried out various data structure operations such as Sorting, Searching, Array Operation, Stack, Queue, Linked List, Tree and Graph. And also calculates time and space complexities of these operations.

During the implementation we have faced many challenges in making the application to take valid data and its processing. We have made an effort to create the user interface that are easily understood by the user. And handled all the challenges successfully. And also handles the various exception during the development project work.

We have learned about different types of sorting, searching and other operation, algorithm implementation of various data structure operation, time and space complexities etc…

Moreover, this project helped for us to understand Software Development Life Cycle (SDLC), Time bound work, team spirit and preparing project document, project testing, GUI designing and presentation.

In addition to that, we learned JavaScript coding and JavaScript tools.

Finally concluded that we tried to fulfil the objectives of project work and goal of our project Performance Analysis and Simulation of Data Structure Algorithms.

Sapthami chatra:201231522226

Preeti Janardhan Naik:201231522213

Sanjana s: 201231522136

**LIMITATION**

* In counting sort only, the elements between 1-10 can be given by the user.
* In bucket sort only the elements between 1-20 can be taken and only 4 buckets are provided for sorting.
* In stack at max of only 5 elements can be pushed.
* In queue at max of only 5 elements can be inserted.
* In BFS and DFS at max of 5 vertexes can be given as an input by the user.

**SCOPE FOR ENHANCEMENT (FUTURE SCOPE)**

* In the bucket sort it is possible to implement a greater number of buckets as per user needs.
* By implementing circular linked list, it is possible to understand the working of it in the data structure.
* It is also possible to take any number of array elements in counting sort.
* It is possible to take any number of vertices from the user in the graph.

**ABBREVIATIONS AND ACRONYMS**

* **RAM**-Random Access Memory
* **GUI-**Graphical User Interface
* **OS**-Operating System
* **SRS**-Software Requirement Specification
* **CFD**-Context Flow Diagram
* **DFD**-Data Flow Diagram
* **BST**-Binary Search Tree
* **BFS**-Breadth First Search
* **DFS**-Depth First Search
* **LIFO**-Last in First Out
* **FIFO-**First in First Out

**BIBLIOGRAPHY/REFERENCES**

[1[Jeremy McPeak and Paul Wilton, Beginning JavaScript, fifth Edition, Wrox publisher,2013

[2] David Flanagan, JavaScript Pocket Reference,3 Edition, O’Reilly Media, Inc.,2012

[3] Ivelin Demirov, Learn JavaScript Visually, Nai Inc 2017

[4] : Pankaj Jalote, An Integrated Approach to Software Engineering, 3 Edition, Narosa Publishing House.

[5] Dr. K. V. K. K. Prasad, Software Testing tools, Dreamtech Press.