

A PROJECT REPORT ON

FUN LEARN

MES Institute of Management & Career Courses, Pune



Submitted By:

Raj Bhattad 2011023
Anuj Gaikwad 2011046
Utkarsh Bhangle 2011022
Suraj Biyani 2011025

INDEX

Sr. No.	Name of Topic	Page No
1	Chapter 1: Introduction	1
	1.1 Scope of the Work	1
	1.2 Operating Environment- Hardware/Software	1
	1.3 Detail Description of the Technology Used	2
	1.4 List of Modules and Assigned to member	2
2	Chapter 2: Proposed System	3
	2.1 User Requirements	3
	2.2 Flow diagram	3
	2.3 Module Algorithm	4
3	Chapter 3: Analysis and Design	6
	3.1 User Interface Design	6
	3.2 Validations applied	10
	3.3 Table Specifications	11
	3.4 Menu Tree	11
4	Output Reports	12
	4.1 List of Reports generated	12
	4.2 Report with data	12
	4.5 Sample Code	12
5	Bibliography	16

Chapter 1: Introduction.

Fun-Learn is a tool to help users to better understand data structures and algorithms, by allowing them to learn the basics on their own and at their own pace.

With all the 'steps' of the algorithm being animated on the website, teachers can use it to quickly demonstrate algorithm examples in class, instead of spending time painstakingly drawing them on the board.

1.1 Scope of system:

The purpose of this project is to build the flow of data structures and algorithms. An experiment to explore design methodologies of DSA and just to have fun with the process. The scope of this project is to build to show the flow of algorithms through animation. System should have a clear animation where students will get proper knowledge after seeing that animation of algorithms. The project is simply built in HTML5, CSS and JavaScript.

1.2 Operating Environment – Hardware and Software:

On Client Side:

- Operating System(any)
- Web Browser (Mozilla Firefox, Google Chrome, Opera, Safari)

Hardware Requirements:

- Minimum requirements: -
- 1 GB RAM
- 80 GB HDD

1.3 Detail Description of the Technology Used:

1. HTML: - Used to design the Front-End of the WebApp
2. CSS: - CSS is used to give styling to the webApp so that it looks attractive
3. PHP: - PHP is used to connect database
4. JavaScript: - JavaScript is used to write Animation Library and Algorithm Library
5. jQuery: - jQuery is used for Algorithm and Animation Libraries.

1.4 Module Assigned to

Module	Assigned to Member
Stack using Array Implementation	Utkarsh Bhangle
Queue using Array Implementation	Anuj Gaikwad
Stack using Linked list Implementation	Raj Bhattad
Queue using Linked List Implementation	Suraj Biyani

Chapter 2: Proposed System.

Fun-Learn is a website where it provides facilities like animation, lines of algorithm as per animation of specific topic to users who want to master Data Structure and Algorithms in an easy manner.

It also contains basics of DSA like Stack, Queue, Linked List, etc.

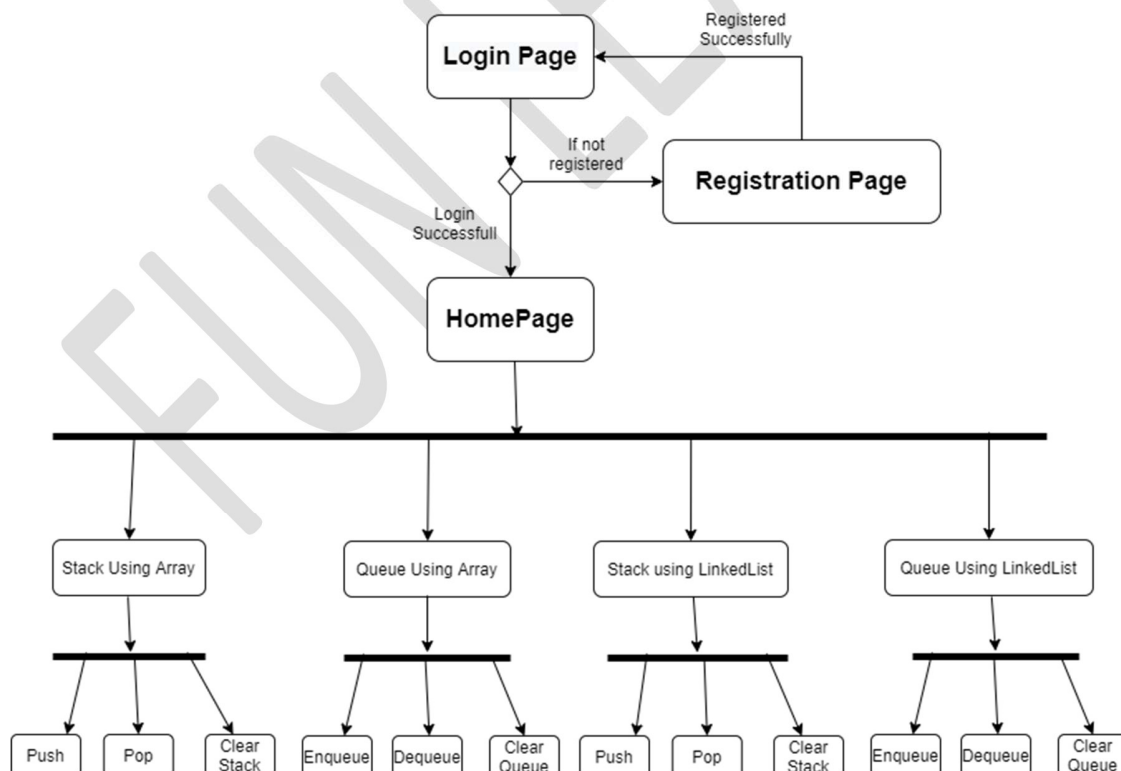
2.1 User Requirements:

System should have a clear animation where user will get proper knowledge about the algorithms after seeing that animation

Website should be created keeping the following things in mind.

1. Proper Animation.
2. Minimum time required for loading the website.
3. Greater Efficiency.
4. Interactive GUI.

2.2 Flow Diagram:



2.3 Module Algorithm:

A. Module 1

*Stack Using Array implementation

1. Enter numbers in stack

2. perform push operation

push animation will perform

3. if (stack! = NULL) then

select pop operation and

pop animation will perform

4. select Clear stack operation

clear stack animation will perform

B. Module 2

Queue using array Implementation

1. Enter numbers in queue

2. perform Enqueue operation

Enqueue animation will perform

3. if (Queue! = NULL) then

select Dequeue operation and

Dequeue animation will perform

4. select Clear Queue operation

clear Queue animation will perform

C. Module 3

*Stack Using Linked List implementation

1. Enter numbers in stack

2. perform push operation

push animation will perform

3. if (stack! = NULL) then

select pop operation and

pop animation will perform

4. select Clear stack operation

clear stack animation will perform

D. Module 4

*Queue using Linked List Implementation

1. Enter numbers in queue

2. perform Enqueue operation

Enqueue animation will perform

3. if (Queue! = NULL) then

select Dequeue operation and

Dequeue animation will perform

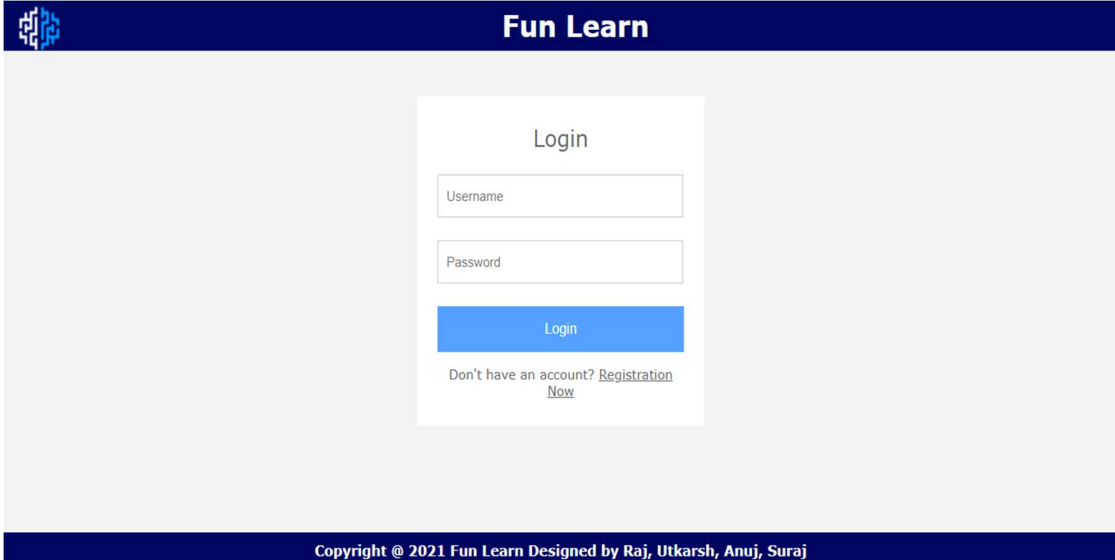
4. select Clear Queue operation

clear Queue animation will perform

Chapter 3: Analysis and Design.

3.1 User Interface Design:

1 Login Page



The image shows a web browser window displaying the 'Fun Learn' login page. The page has a dark blue header with the 'Fun Learn' logo on the left and the text 'Fun Learn' in white on the right. The main content area is light gray and contains a white login form. The form has a title 'Login', two input fields labeled 'Username' and 'Password', and a blue 'Login' button. Below the button, there is a link that says 'Don't have an account? [Registration Now](#)'. The footer is a dark blue bar with the text 'Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj'.

Fun Learn

Login

Username

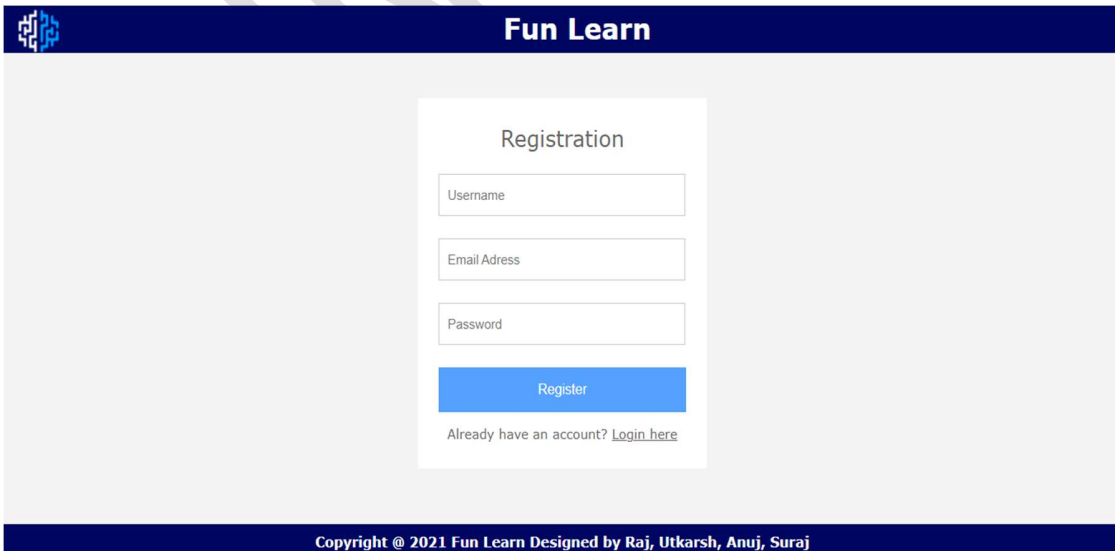
Password

Login

Don't have an account? [Registration Now](#)

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

2 Registration Page



The image shows a web browser window displaying the 'Fun Learn' registration page. The page has a dark blue header with the 'Fun Learn' logo on the left and the text 'Fun Learn' in white on the right. The main content area is light gray and contains a white registration form. The form has a title 'Registration', three input fields labeled 'Username', 'Email Address', and 'Password', and a blue 'Register' button. Below the button, there is a link that says 'Already have an account? [Login here](#)'. The footer is a dark blue bar with the text 'Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj'.

Fun Learn

Registration

Username

Email Address


Password

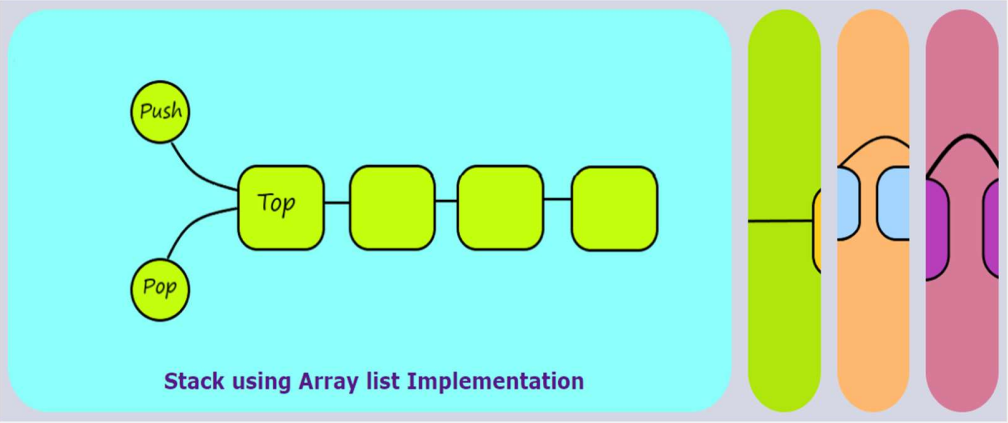
Register

Already have an account? [Login here](#)

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

3 Home Page


 **Fun Learn** Welcome, RajBhattad! [Logout](#)



Stack using Array list Implementation

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

4 Implementation of Array Using Stack

Implementation of Array Using Stack 

Push Pop Clear Stack

top

-1

0

1

2

3

4

5

6

PUSH
Step 1 - Check whether Stack is Full, (top = n)
Step 2 - If FULL, then Clear the Stack!!!
Step 3 - If NOT FULL, then push the value and increment top value by one (top=top+1)

POP
Step 1 - Check whether stack is EMPTY, (top = -1)
Step 2 - If EMPTY, the Pop will not perform
Step 3 - If NOT EMPTY, then it will remove one element and value of top will be decremented by 1. (top = top - 1)

Clear Stack
Step 1 - If (Top > -1) Then
Step 2 - Clear the Stack!!!
step 3 - Set (Top = -1)

Animation Completed

Skip Back Step Back Pause Step Forward Skip Forward

Animation Speed

w: 900 h: 500 Change Canvas Size Move Controls

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

5 Implementation of Array Using Queue

Implementation of Array Using Queue

Enqueue Dequeue Clear Queue

Front 0 Rear -1

0

1

2

3

4

5

6

EnQUEUE
Step 1 - Check whether queue is FULL. (rear == SIZE-1)
Step 2 - Queue is FULL, Clear the QUEUE!!!
Step 3 - If it is NOT FULL, then Enqueue the value and increment rear value by one (rear++) and set queue[rear] = value.

DeQUEUE
Step 1 - Check whether queue is EMPTY. (front > rear)
Step 2 - If it is EMPTY, then Dequeue will not perform
Step 3 - If it is NOT EMPTY, then increment the front value by one (front ++).
Then display dequeued value[front] as deleted element. Then check whether both front and rear are equal (front == rear)

Clear Queue
Step 1 - Check if (rear == SIZE-1) then
Step 2 - Queue is FULL, Clear the QUEUE!!!
Step 3 - Set FRONT = 0 & REAR = -1

Animation Completed

Skip Back Step Back Pause Step Forward Skip Forward Animation Speed w: 900 h: 500 Change Canvas Size Move Controls

Copyright © 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

6 Implementation of Stack Using Linked List

Implementation of Stack Using Linked List

Push Pop Clear Stack

Top

PUSH
Step 1 - Create a newNode with given value.
Step 2 - Check whether stack is Empty (top == NULL)
Step 3 - If Empty, then set newNode → next = NULL.
Step 4 - If Not Empty, then set newNode → next = top.
Step 5 - Finally, set top = newNode.

POP
Step 1 - Check whether stack is Empty (top == NULL).
Step 2 - If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'.
Step 3 - Then set 'top = top → next'.
Step 4 - Finally, delete 'temp'. (free(temp)).

Clear STACK
It will clear all the elements in the Stack and the top value will reset

Animation Completed

Skip Back Step Back Pause Step Forward Skip Forward Animation Speed w: 900 h: 500 Change Canvas Size Move Controls

Copyright © 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

7 Implementation of Queue Using Linked List

Implementation of Queue Using Linked List

Enqueue

Dequeue

Clear Queue

Front

Rear

Animation Completed

Skip Back

Step Back

Pause

Step Forward

Skip Forward

Animation Speed

w: 1000

h: 500

Change Canvas Size

Move Controls

Copyright © 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

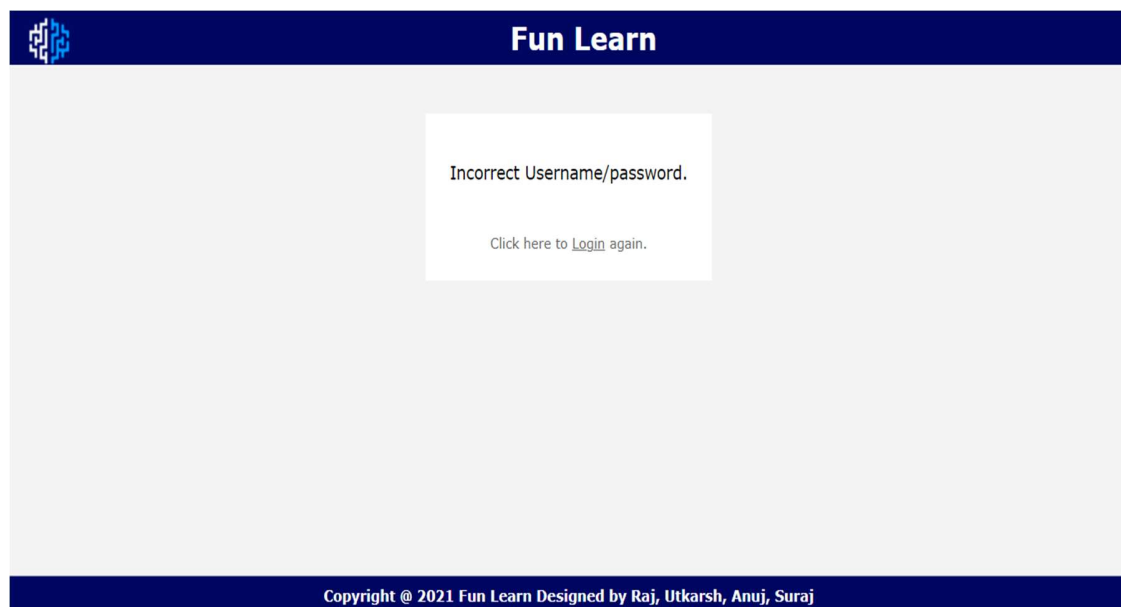
EnQUEUE
Step 1 - Create a newNode with given value and set 'newNode → next' to NULL.
Step 2 - Check whether queue is Empty (rear == NULL)
Step 3 - If Empty then, set front = newNode and rear = newNode.
Step 4 - If Not Empty then, set rear → next = newNode and rear = newNode.

DeQUEUE
Step 1 - Check whether queue is Empty (front == NULL).
Step 2 - If Not Empty then, define a Node pointer 'temp' and set it to 'front'.
Step 3 - Then set 'front = front → next' and delete 'temp' (free(temp)).

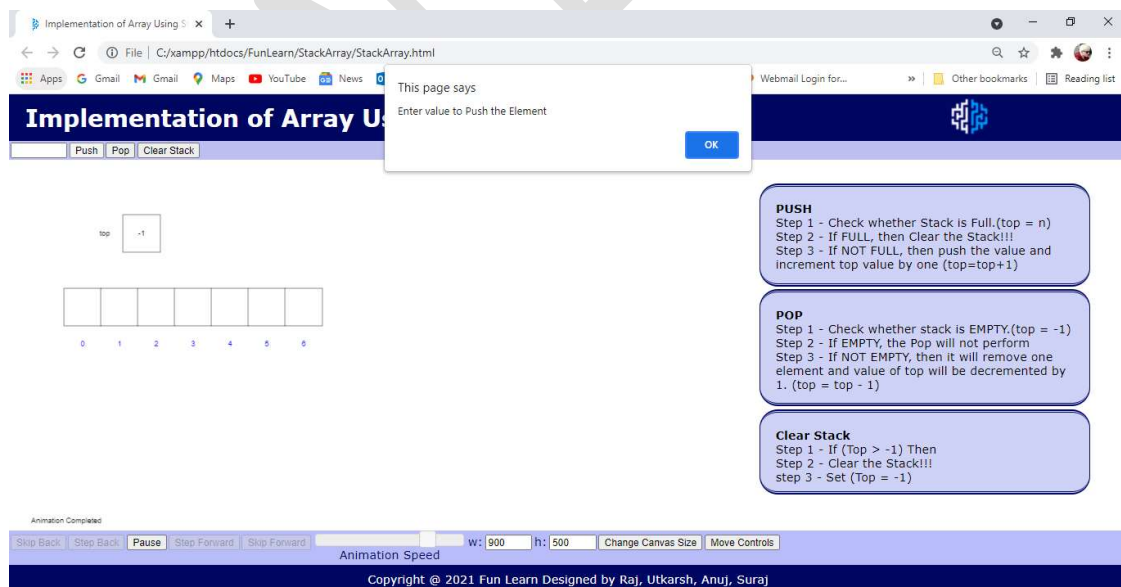
Clear Queue
Step 1 - Check whether the queue is EMPTY.
Step 2 - If not EMPTY, clear the queue.

3.2 Validations Applied:

1 Login Page Validation



2 Implementation of Array Using Stack



3 Implementation of Array Using Stack

Implementation of Array Using Stack

Buttons: Enqueue Dequeue Clear Queue

Message: This page says Queue Underflow

Diagram: Front 0, Rear -1, Array [0, 1, 2, 3, 4, 5, 6]

EnQUEUE
Step 1 - Check whether queue is FULL. (rear == SIZE-1)
Step 2 - Queue is FULL, Clear the QUEUE!!!
Step 3 - If it is NOT FULL, then Enqueue the value and increment rear value by one (rear++) and set queue[rear] = value.

DeQUEUE
Step 1 - Check whether queue is EMPTY. (front > rear)
Step 2 - If it is EMPTY, then Dequeue will not perform
Step 3 - If it is NOT EMPTY, then increment the front value by one (front ++).
Then display dequeued value[front] as deleted element. Then check whether both front and rear are equal (front == rear)

Clear Queue
Step 1 - Check if (rear == SIZE-1) then
Step 2 - Queue is FULL, Clear the QUEUE!!!
Step 3 - Set FRONT = 0 & REAR = -1

Animation Completed

Controls: Skip Back, Step Back, Pause, Step Forward, Skip Forward, Animation Speed, w: 900, h: 500, Change Canvas Size, Move Controls

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

4 Implementation of Array Using Stack

Implementation of Stack Using Linked List

Buttons: Push Pop Clear Stack

Message: This page says Stack is already empty

Diagram: Top

PUSH
Step 1 - Create a newNode with given value.
Step 2 - Check whether stack is Empty (top == NULL)
Step 3 - If Empty, then set newNode → next = NULL.
Step 4 - If Not Empty, then set newNode → next = top.
Step 5 - Finally, set top = newNode.

POP
Step 1 - Check whether stack is Empty (top == NULL).
Step 2 - If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'.
Step 3 - Then set 'top = top → next'.
Step 4 - Finally, delete 'temp'. (free(temp)).

Clear STACK
It will clear all the elements in the Stack and the top value will reset

Animation Completed

Controls: Skip Back, Step Back, Pause, Step Forward, Skip Forward, Animation Speed, w: 900, h: 500, Change Canvas Size, Move Controls

Copyright @ 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

3.3 Table Specifications:

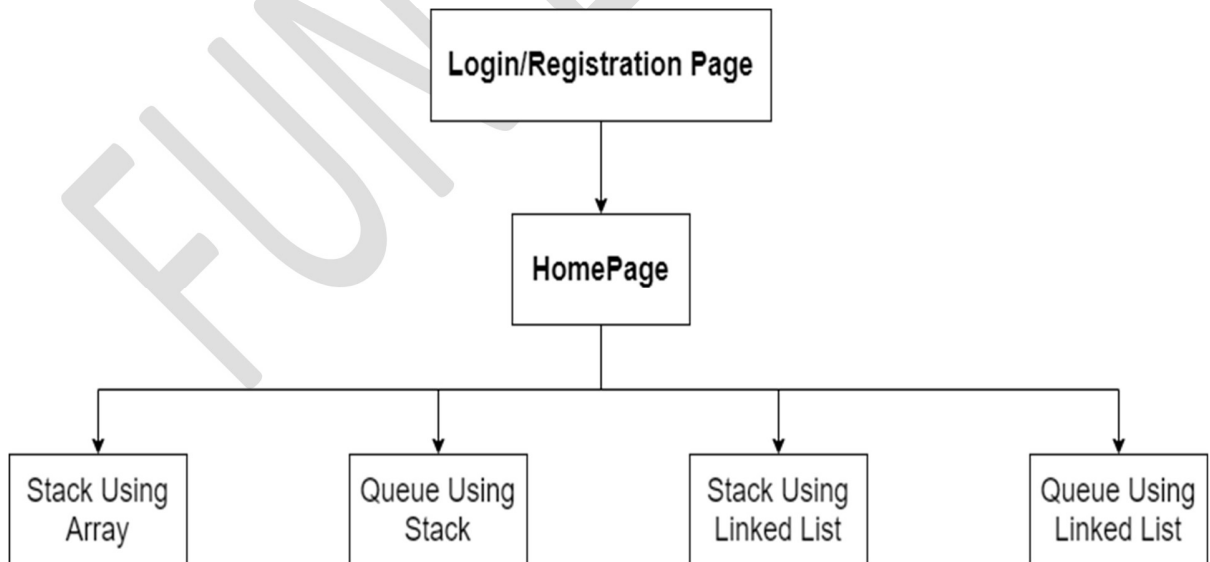
Database Name: FunLearn

Table Name: users

Data Dictionary:

Sr. No.	Field Name	Data Type	Size	Description
1	Id	Int	11	Unique Id is given for every user
2	username	Varchar	100	Username of user at login time
3	email	Varchar	100	Email is taken at the time of registration
4	password	Varchar	100	Password of user at the time of Login
5	create_datetime	Date/Time		Date and Time is taken when user registers first time

3.4 Menu Tree:



Chapter 4: Output Reports.

4.1 List of Reports Generated.

Not Applicable

4.2 Reports with Data.

1 Report with data of Queue Using Linked List

Implementation of Queue Using Linked List

Enqueue Dequeue Clear Queue

EnQUEUE
Step 1 - Create a newNode with given value and set 'newNode → next' to NULL.
Step 2 - Check whether queue is Empty (rear == NULL)
Step 3 - If Empty then, set front = newNode and rear = newNode.
Step 4 - If Not Empty then, set rear → next = newNode and rear = newNode.

DeQUEUE
Step 1 - Check whether queue is Empty (front == NULL).
Step 2 - If Not Empty then, define a Node pointer 'temp' and set it to 'front'.
Step 3 - Then set 'front = front → next' and delete 'temp' (free(temp)).

Clear Queue
Step 1 - Check whether the queue is EMPTY.
Step 2 - If not EMPTY, clear the queue.

Skip Back Step Back Pause Step Forward Skip Forward Animation Speed v: 1000 h: 500 Change Canvas Size Move Controls

Animation Completed

Copyright © 2021 Fun Learn Designed by Raj, Utkarsh, Anuj, Suraj

4.3 Sample Code

```
var LINKED_LIST_START_X = 100;
var LINKED_LIST_START_Y = 200;
var LINKED_LIST_ELEM_WIDTH = 70;
var LINKED_LIST_ELEM_HEIGHT = 30;

var LINKED_LIST_INSERT_X = 250;
var LINKED_LIST_INSERT_Y = 50;

var LINKED_LIST_ELEMS_PER_LINE = 8;
var LINKED_LIST_ELEM_SPACING = 100;
var LINKED_LIST_LINE_SPACING = 100;

var TOP_POS_X = 180;
var TOP_POS_Y = 100;
var TOP_LABEL_X = 130;
var TOP_LABEL_Y = 100;
```

```

var TOP_ELEM_WIDTH = 30;
var TOP_ELEM_HEIGHT = 30;

var TAIL_POS_X = 180;
var TAIL_LABEL_X = 130;

var PUSH_LABEL_X = 50;
var PUSH_LABEL_Y = 30;
var PUSH_ELEMENT_X = 120;
var PUSH_ELEMENT_Y = 30;

var SIZE = 32;

function QueueLL(am, w, h)
{
    this.init(am, w, h);
}

QueueLL.prototype = new Algorithm();
QueueLL.prototype.constructor = QueueLL;
QueueLL.superclass = Algorithm.prototype;

QueueLL.prototype.init = function(am, w, h)
{
    QueueLL.superclass.init.call(this, am, w, h);
    this.addControls();
    this.nextIndex = 0;
    this.commands = [];
    this.tail_pos_y = h - LINKED_LIST_ELEM_HEIGHT;
    this.tail_label_y = this.tail_pos_y;
    this.setup();
    this.initialIndex = this.nextIndex;
}

QueueLL.prototype.addControls = function()
{
    this.controls = [];
    this.enqueueField = addControlToAlgorithmBar("Text", "");
    this.enqueueField.onkeydown = this.returnSubmit(this.enqueueField,
this.enqueueCallback.bind(this), 6);
    this.enqueueButton = addControlToAlgorithmBar("Button", "Enqueue");
    this.enqueueButton.onclick = this.enqueueCallback.bind(this);
    this.controls.push(this.enqueueField);
    this.controls.push(this.enqueueButton);

    this.dequeueButton = addControlToAlgorithmBar("Button", "Dequeue");
    this.dequeueButton.onclick = this.dequeueCallback.bind(this);
    this.controls.push(this.dequeueButton);

    this.clearButton = addControlToAlgorithmBar("Button", "Clear Queue");

```



```

        this.clearButton.onclick = this.clearCallback.bind(this);
        this.controls.push(this.clearButton);
    }

    QueueLL.prototype.enableUI = function(event)
    {
        for (var i = 0; i < this.controls.length; i++)
        {
            this.controls[i].disabled = false;
        }
    }

    QueueLL.prototype.disableUI = function(event)
    {
        for (var i = 0; i < this.controls.length; i++)
        {
            this.controls[i].disabled = true;
        }
    }

    QueueLL.prototype.setup = function()
    {
        this.linkedListElemID = new Array(SIZE);
        for (var i = 0; i < SIZE; i++)
        {
            this.linkedListElemID[i] = this.nextIndex++;
        }
        this.headID = this.nextIndex++;
        this.headLabelID = this.nextIndex++;

        this.tailID = this.nextIndex++;
        this.tailLabelID = this.nextIndex++;

        this.arrayData = new Array(SIZE);
        this.top = 0;
        this.leftoverLabelID = this.nextIndex++;

        this.cmd("CreateLabel", this.headLabelID, "Head", TOP_LABEL_X, TOP_LABEL_Y);
        this.cmd("CreateRectangle", this.headID, "", TOP_ELEM_WIDTH, TOP_ELEM_HEIGHT,
TOP_POS_X, TOP_POS_Y);
        this.cmd("SetNull", this.headID, 1);

        this.cmd("CreateLabel", this.tailLabelID, "Tail", TAIL_LABEL_X, this.tail_label_y);
        this.cmd("CreateRectangle", this.tailID, "", TOP_ELEM_WIDTH, TOP_ELEM_HEIGHT,
TAIL_POS_X, this.tail_pos_y);
        this.cmd("SetNull", this.tailID, 1);

        this.cmd("CreateLabel", this.leftoverLabelID, "", 5, PUSH_LABEL_Y, 0);
        this.animationManager.StartNewAnimation(this.commands);
    }

```

```

        this.animationManager.skipForward();
        this.animationManager.clearHistory();
    }

    QueueLL.prototype.resetLinkedListPositions = function()
    {
        for (var i = this.top - 1; i >= 0; i--)
        {
            var nextX = (this.top - 1 - i) % LINKED_LIST_ELEMS_PER_LINE *
LINKED_LIST_ELEM_SPACING + LINKED_LIST_START_X;
            var nextY = Math.floor((this.top - 1 - i) / LINKED_LIST_ELEMS_PER_LINE) *
LINKED_LIST_LINE_SPACING + LINKED_LIST_START_Y;
            this.cmd("Move", this.linkedListElemID[i], nextX, nextY);
        }
    }
}

```

Chapter 5: Bibliography.

- 1 <https://www.udemy.com/course/algorithms-and-data-structures-in-javascript/>
- 2 <https://www.section.io/engineering-education/best-javascript-animation-libraries/>
- 3 <https://www.oreilly.com/library/view/data-structures-and/9781449373931/ch04.html>
- 4 [https://www.javascripttutorial.net/javascript-queue/#:~:text=Introduction%20to%20the%20Queue%20data,%20Dout%20\(FIFO\)%20principle.](https://www.javascripttutorial.net/javascript-queue/#:~:text=Introduction%20to%20the%20Queue%20data,%20Dout%20(FIFO)%20principle.)
- 5 <https://www.javapoint.com/>
- 6 <https://www.tutorialspoint.com/>
- 7 <https://www.geeksforgeeks.com/>