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

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.

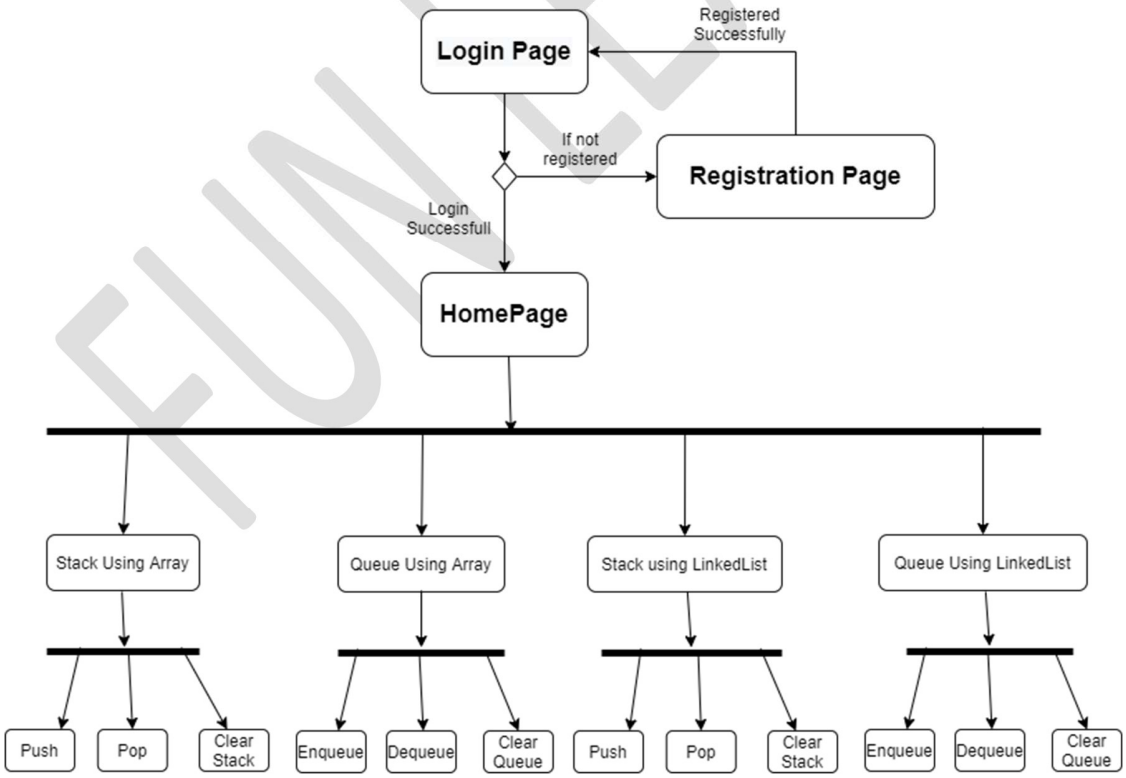
2.1 User Requirements:

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

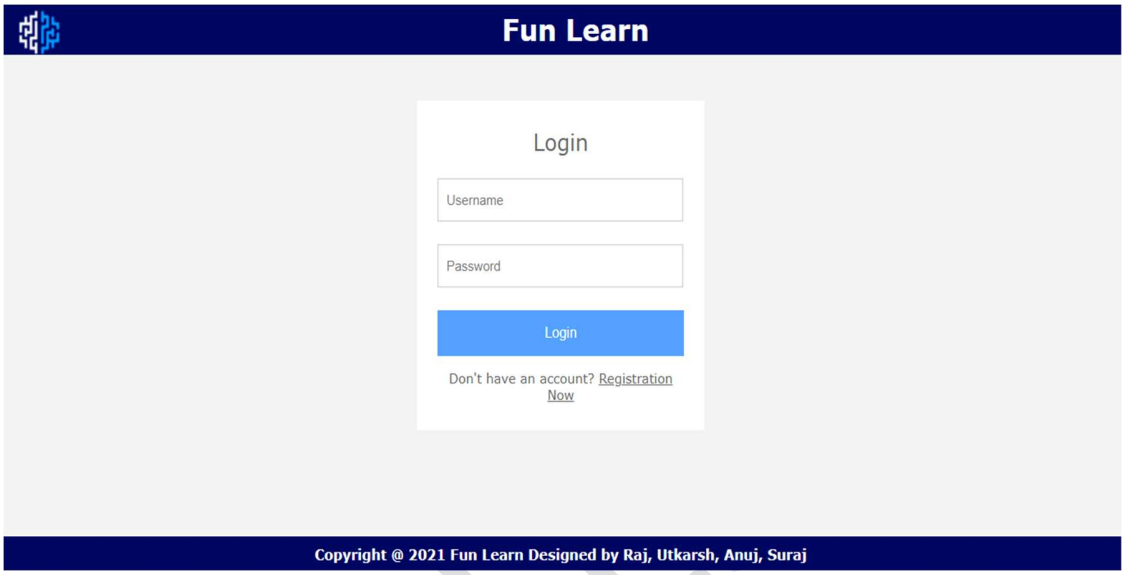
4. select Clear Queue operation

clear Queue animation will perform

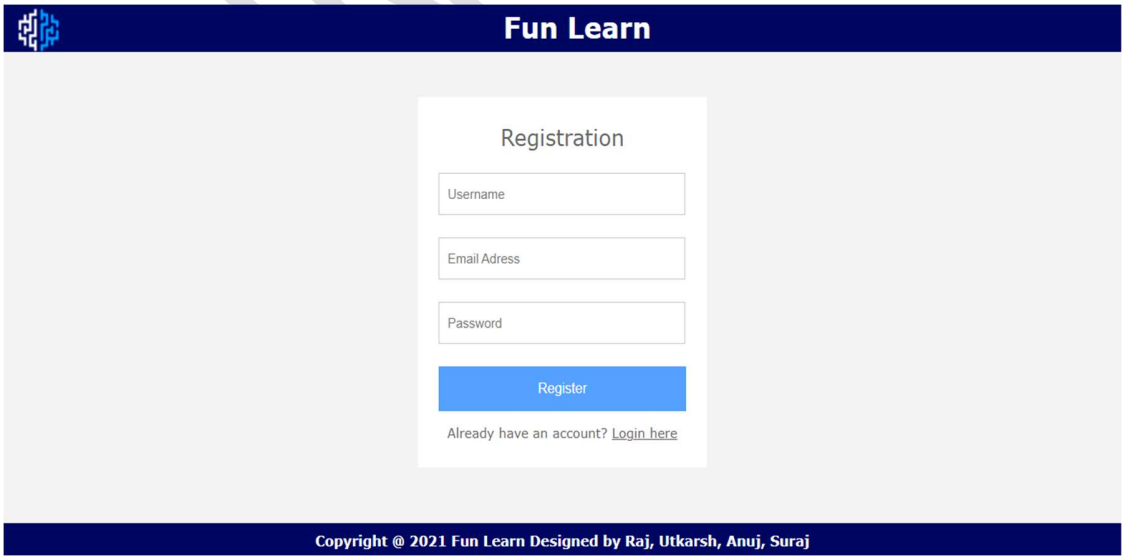
Chapter 3: Analysis and Design.

3.1 User Interface Design:

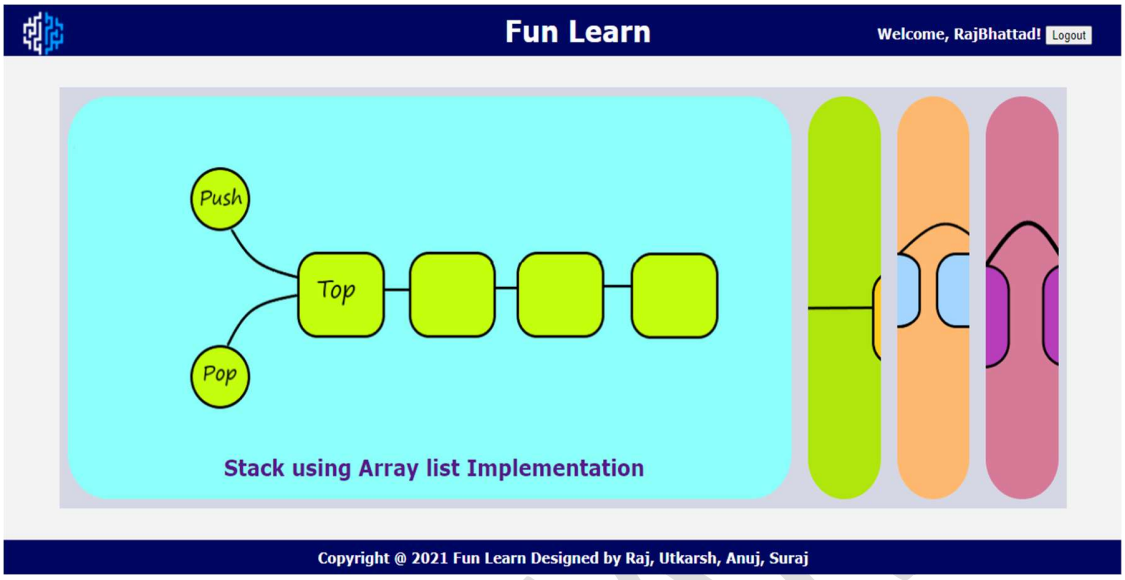
1 Login Page



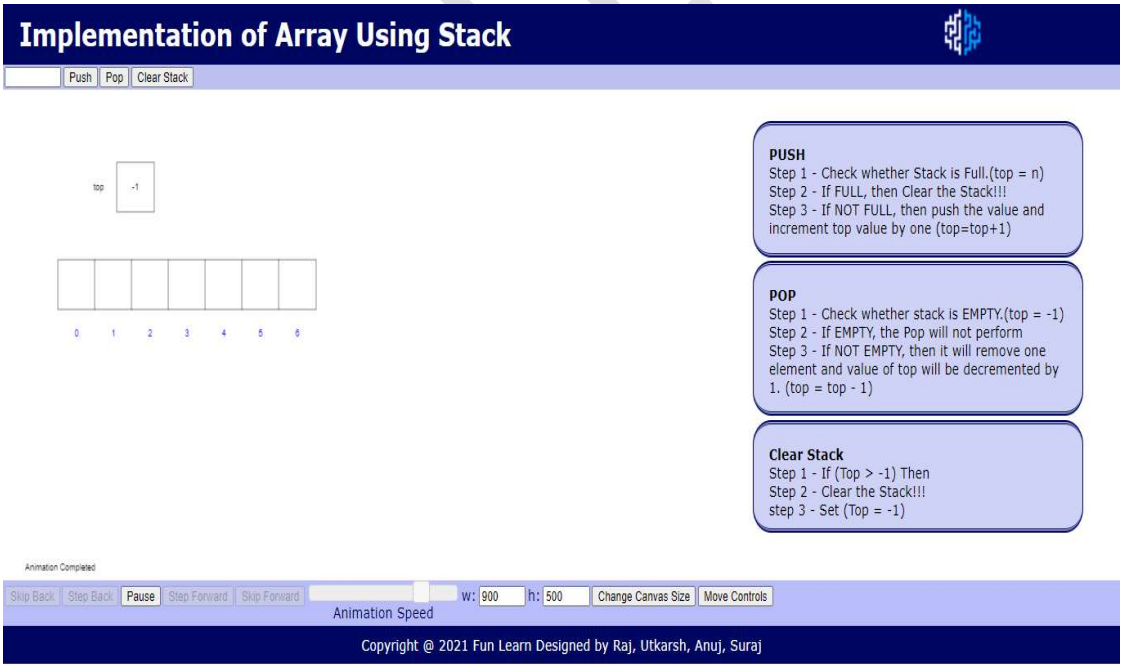
2 Registration Page



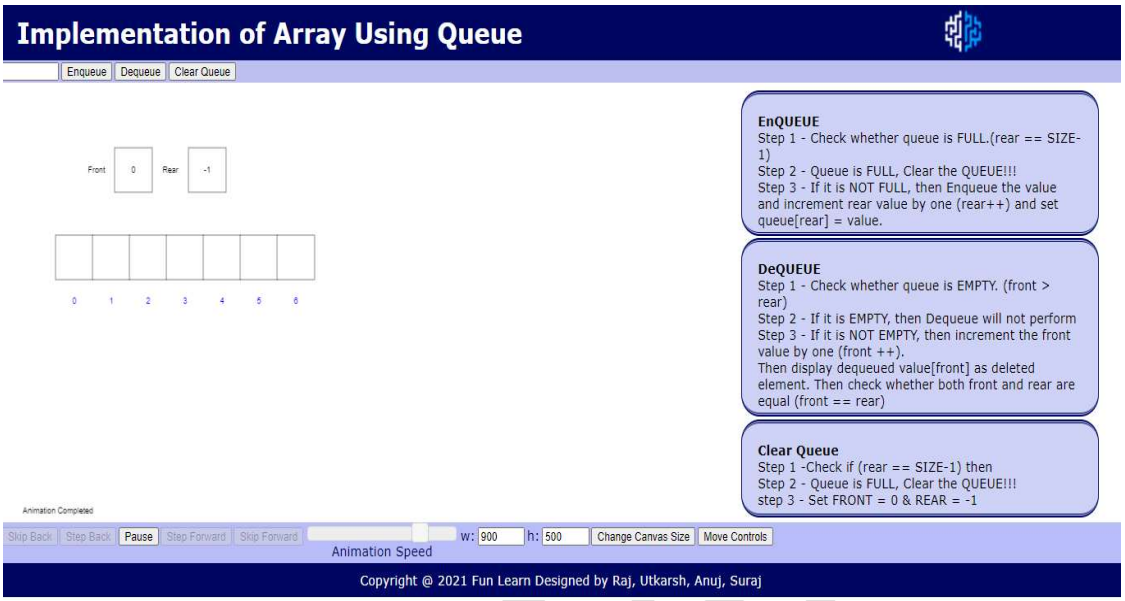
3 Home Page



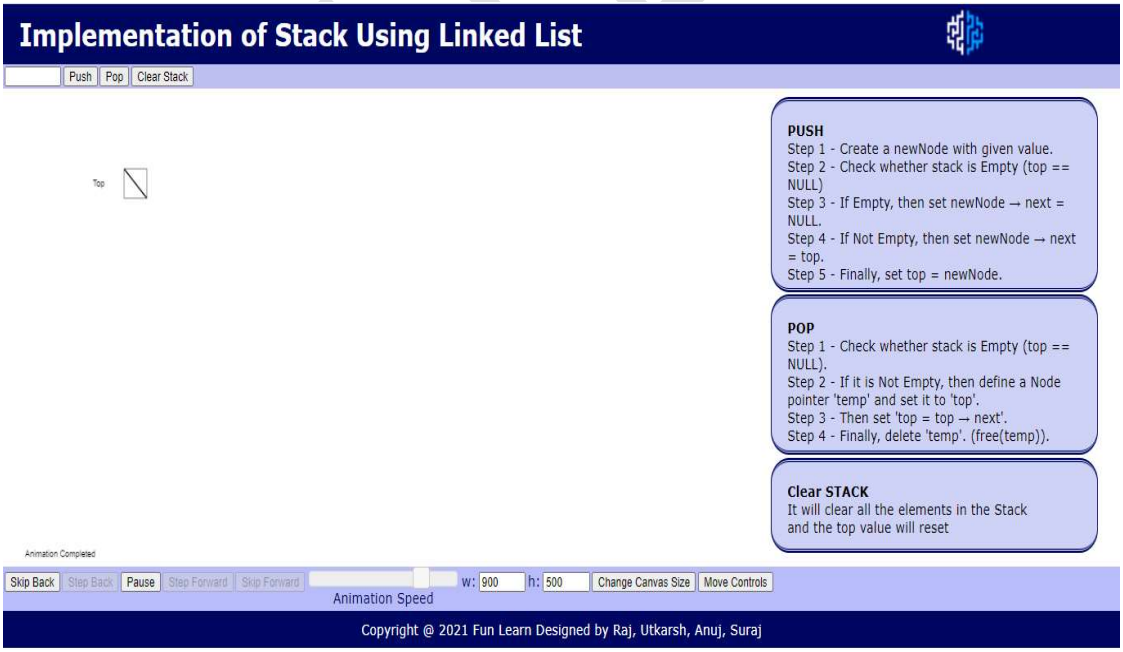
4 Implementation of Array Using Stack



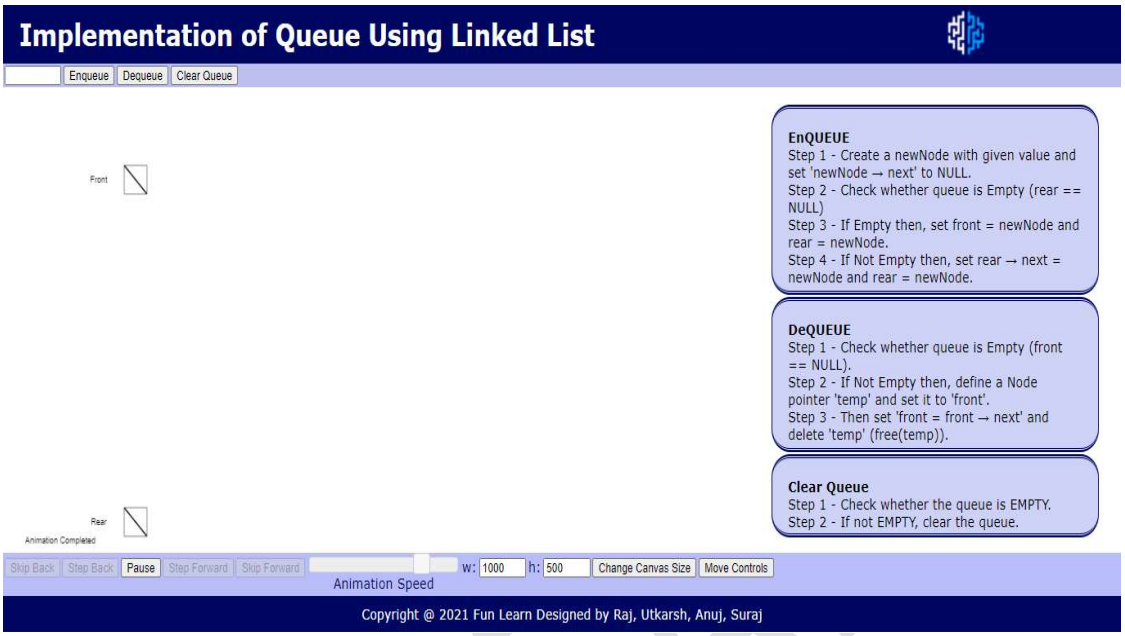
5 Implementation of Array Using Queue



6 Implementation of Stack Using Linked List

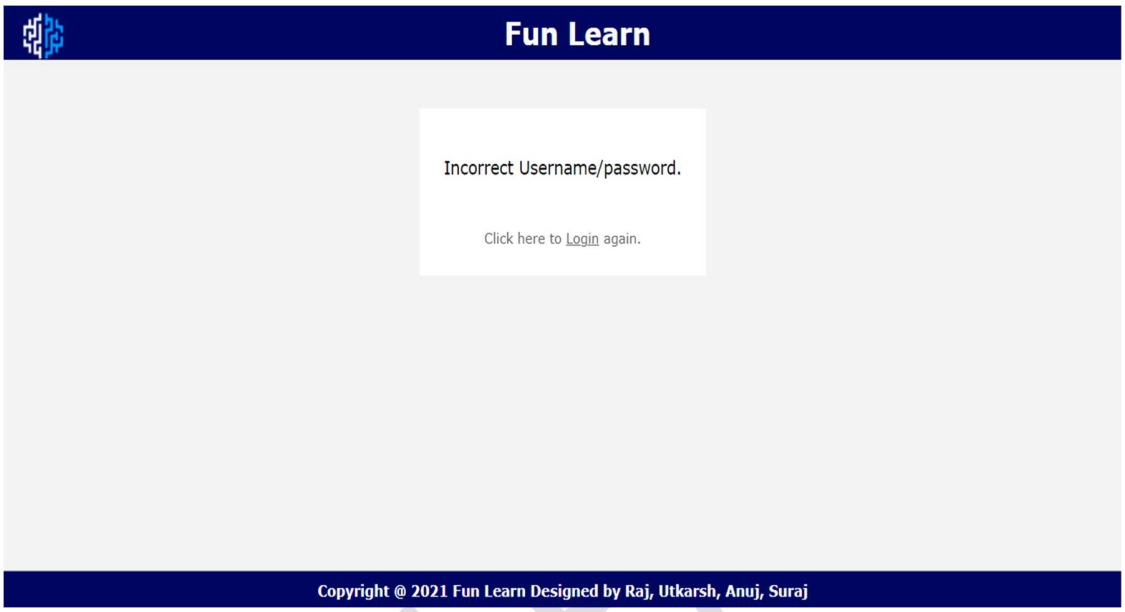


7 Implementation of Queue Using Linked List

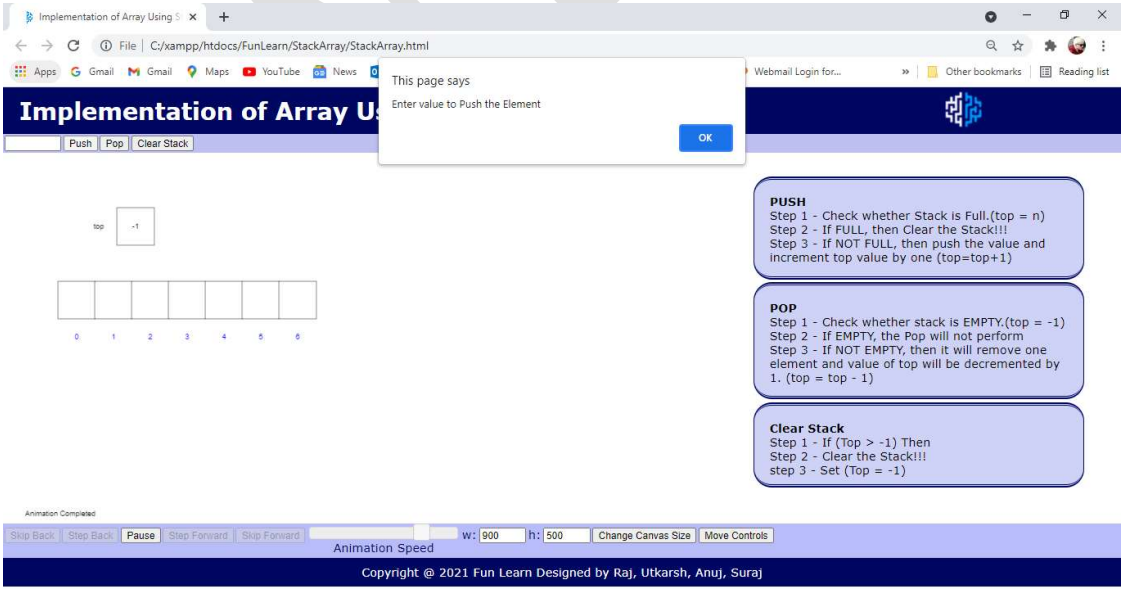


3.2 Validations Applied:

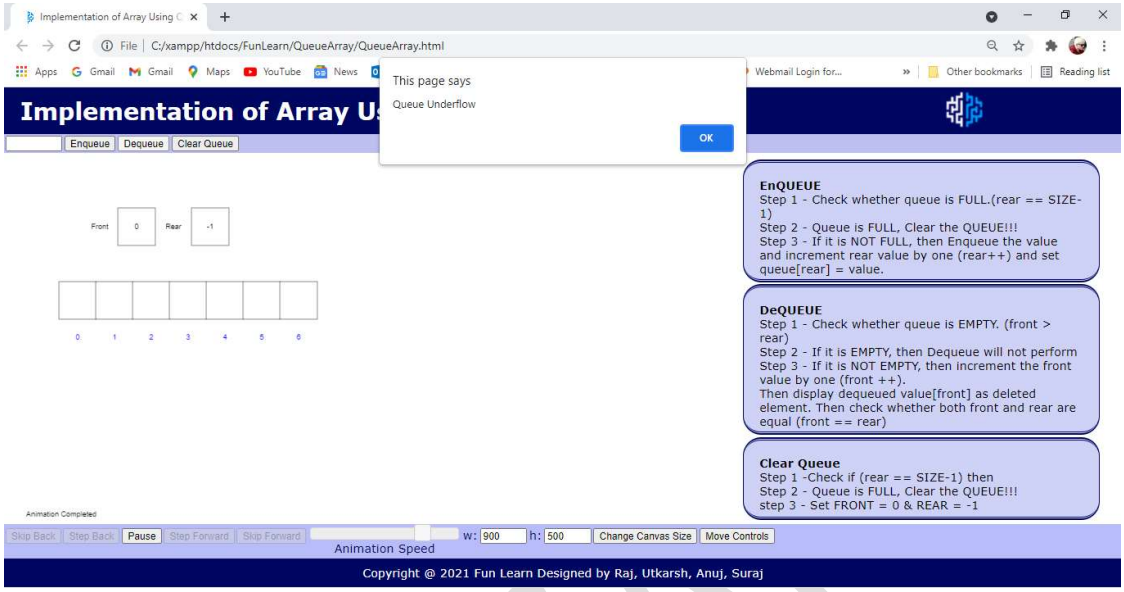
1 Login Page Validation



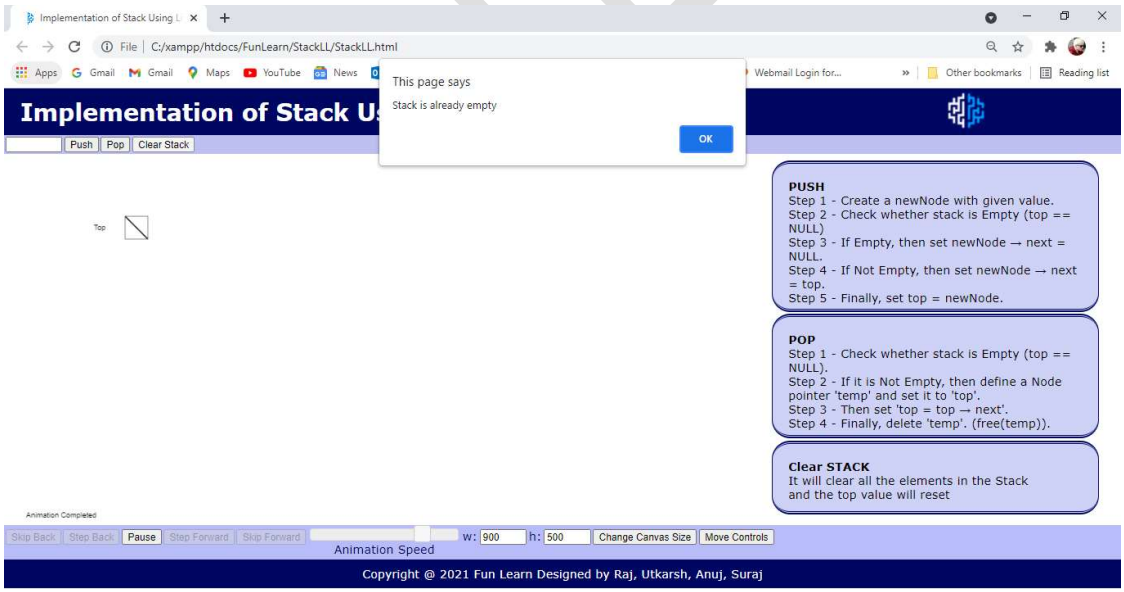
2 Implementation of Array Using Stack



3 Implementation of Array Using Stack



4 Implementation of Array Using Stack

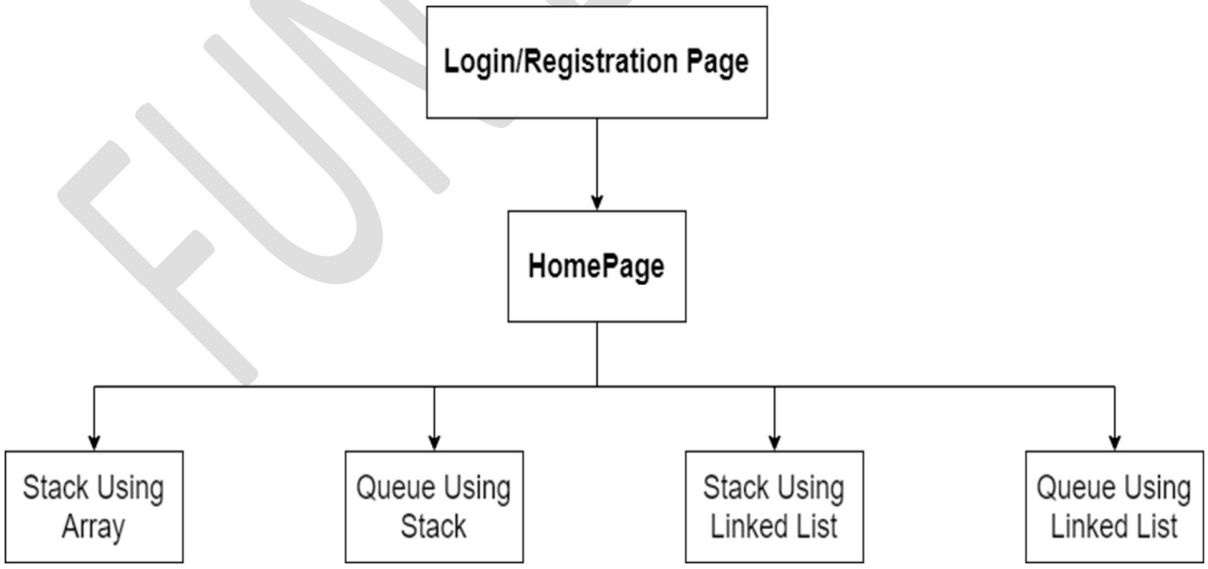


3.3 Table Specifications:

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:



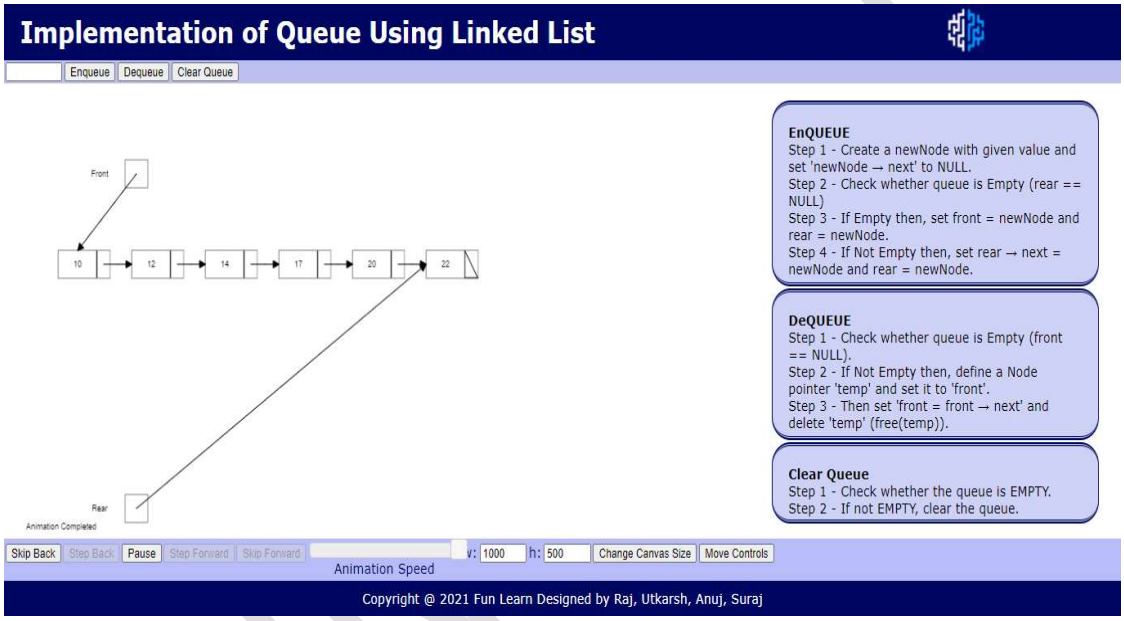
Chapter 4: Output Reports.

4.1 List of Reports Generated.

Not Applicable

4.2 Reports with Data.

1 Report with data of Queue Using Linked List



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,%2Dout%20(FIFO)%20principle.
5 https://www.javapoint.com/
6 https://www.tutorialspoint.com/
7 https://www.geeksforgeeks.com/