|  |  |
| --- | --- |
| LA1 06.08.2024 | **Creation of a webpage** |

**AIM:**

To create a website using HTML, CSS & JavaScript.

**ALGORITHM:**

1. Create HTML Structure (`index.html`):

- Define the document type and language in the `<head>`.

- Include the title and link to the CSS file.

- Add a header with the university name and student details.

- Create sections for:

- Welcome message and image.

- Embedded YouTube video using an `<iframe>`.

- Feedback form with inputs for name, email, and message.

- Add a footer with copyright information.

2. Style the Webpage (`styles.css`):

- Set body styles (font, background, padding).

- Style the header and each section (margins, padding, background, border-radius).

- Style form elements (display, margins, padding, borders).

- Style the submit button (background color, hover effect).

3. Run the Webpage:

- Save all files and open `index.html` in a web browser.

**CODE:**

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Learn About AI</title>

<link rel="stylesheet" href="styles.css">

<script defer src="script.js"></script>

</head>

<body>

<header>

<h1>Artificial Intelligence</h1>

<nav>

<a href="#about">What is AI?</a>

<a href="#applications">AI Applications</a>

<a href="#technologies">AI Technologies</a>

</nav>

</header>

<main>

<section id="about">

<h2>What is AI?</h2>

<p>Artificial Intelligence (AI) is the simulation of human intelligence in machines programmed to think and learn.</p>

</section>

<section id="applications">

<h2>AI Applications</h2>

<ul>

<li>Healthcare</li>

<li>Finance</li>

<li>Autonomous Vehicles</li>

<li>Education</li>

</ul>

</section>

<section id="technologies">

<h2>AI Technologies</h2>

<p>Core technologies driving AI include Machine Learning, Neural Networks, and Natural Language Processing.</p>

</section>

</main>

<footer>

<p>Contact us: aiwebsite@example.com</p>

</footer>

</body>

</html>

**CSS**

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

color: #333;

line-height: 1.6;

}

/\* Header Styling \*/

header {

background: #333;

color: #fff;

padding: 1rem;

text-align: center;

}

nav a {

color: #fff;

margin: 0 1rem;

text-decoration: none;

}

/\* Main Content Styling \*/

main {

padding: 2rem;

}

section {

margin-bottom: 2rem;

}

h2 {

color: #444;

}

/\* Footer Styling \*/

footer {

background: #333;

color: #fff;

text-align: center;

padding: 1rem;

position: fixed;

bottom: 0;

width: 100%;

}

**JAVASCRIPT**

document.querySelectorAll('nav a').forEach(anchor => {

anchor.addEventListener('click', function(e) {

e.preventDefault();

document.querySelector(this.getAttribute('href')).scrollIntoView({

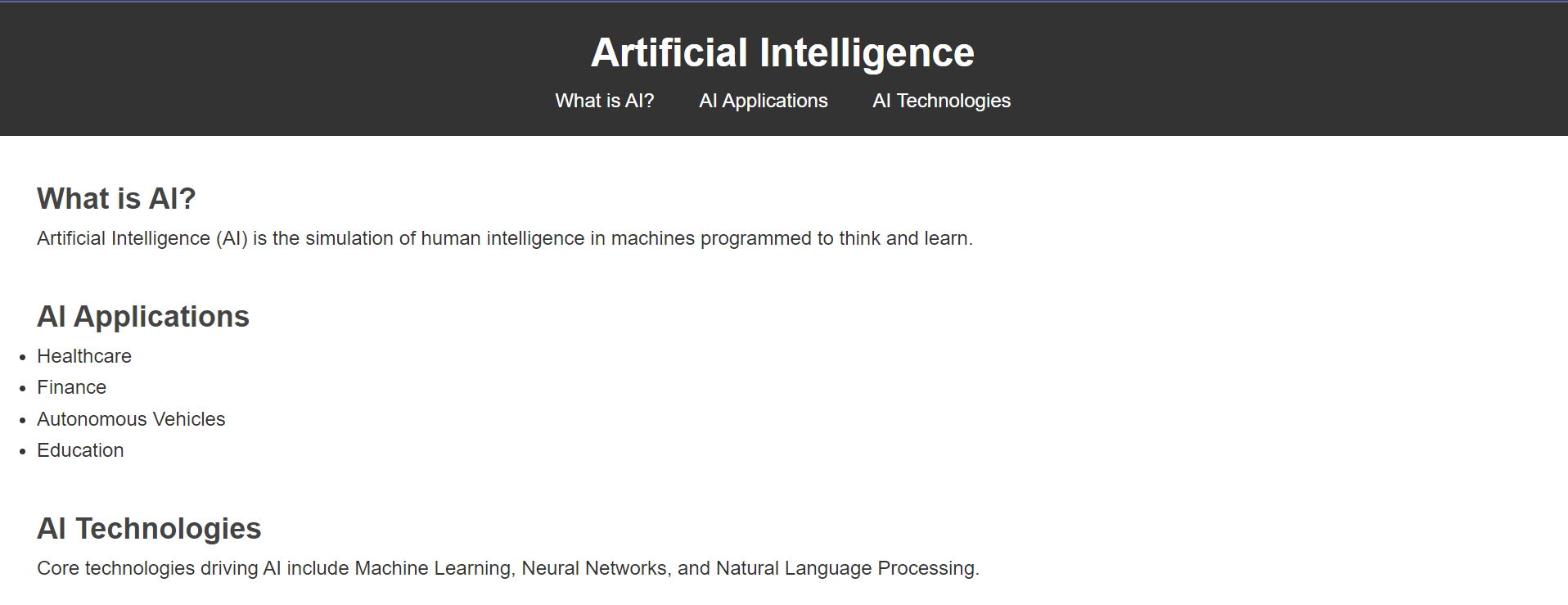
behavior: 'smooth'

});

});

});

**OUTPUT:**



**RESULT:**

Thus, a responsive website is created successfully using HTML,CSS & JavaScript.

|  |  |
| --- | --- |
| LA2 13.08.2024 | **React JS program to switch between layouts** |

**AIM:**

To create a react website to switch between layouts.

**ALGORITHM:**

1. Create two layout components: CardLayout and TableLayout.

2. Use a state variable to manage the current layout (card or table).

3. Implement buttons to toggle between layouts.

4. Conditionally render the appropriate layout based on the state.

**CODE:**

**App.js**

import React, { useState } from 'react';

import './App.css';

const CardLayout = () => (

<div className="layout card">

<div className="card-item">Item 1</div>

<div className="card-item">Item 2</div>

<div className="card-item">Item 3</div>

</div>

);

const TableLayout = () => (

<table className="layout table">

<thead>

<tr>

<th>Item</th>

</tr>

</thead>

<tbody>

<tr><td>Item 1</td></tr>

<tr><td>Item 2</td></tr>

<tr><td>Item 3</td></tr>

</tbody>

</table>

);

const App = () => {

const [layout, setLayout] = useState('card');

return (

<div className="App">

<header className="header">

<button className="button" onClick={() => setLayout('card')}>Card Layout</button>

<button className="button" onClick={() => setLayout('table')}>Table Layout</button>

</header>

{layout === 'card' ? <CardLayout /> : <TableLayout />}

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

}

.header {

margin-bottom: 20px;

}

.button {

background-color: #ff5722;

color: white;

border: none;

padding: 10px 20px;

margin: 5px;

cursor: pointer;

font-size: 18px;

border-radius: 5px;

}

.button:hover {

background-color: #e64a19;

}

.layout.card {

display: flex;

justify-content: center;

gap: 20px;

}

.card-item {

background-color: #00bcd4;

color: white;

padding: 30px;

border-radius: 10px;

font-size: 24px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

}

.layout.table {

margin: 0 auto;

border-collapse: collapse;

}

.table th, .table td {

border: 1px solid #ddd;

padding: 15px;

text-align: center;

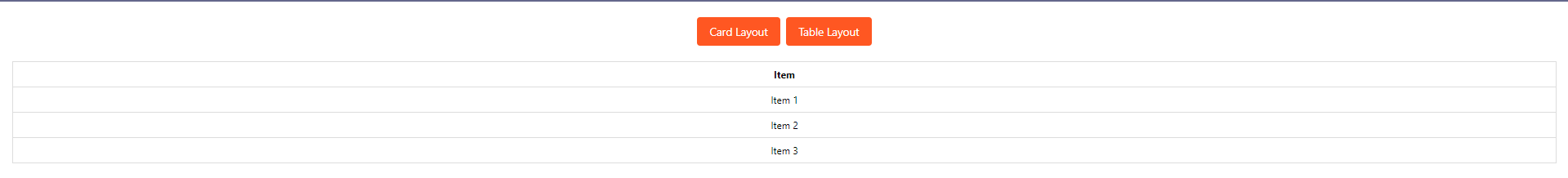
}

.table th {

background-color: #00bcd4;

color: white;

}



**RESULT:**

Thus, a react website to switch between layouts is created successfully.

|  |  |
| --- | --- |
| LA3 13.08.2024 | **React JS program to implement search filter items** |

**AIM:**

To create a React website to implement Search to Filter Items.

**ALGORITHM:**

1. Maintain a list of items and a search query state.

2. Filter the items based on the search query.

3. Render the filtered items in a list.

**CODE:**

**App.js**

import React, { useState } from 'react';

import './App.css';

const items = ['Laptop', 'Smartphone', 'Headphones', 'Smartwatch', 'Keyboard'];

const App = () => {

const [query, setQuery] = useState('');

const filteredItems = items.filter(item =>

item.toLowerCase().includes(query.toLowerCase())

);

return (

<div className="App">

<input

className="search-box"

type="text"

placeholder="Search items..."

value={query}

onChange={(e) => setQuery(e.target.value)}

/>

<ul className="item-list">

{filteredItems.length > 0 ? (

filteredItems.map((item, index) => (

<li key={index} className="item">{item}</li>

))

) : (

<li className="item">No items found</li>

)}

</ul>

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

}

.search-box {

width: 300px;

padding: 10px;

font-size: 18px;

border: 2px solid #ff5722;

border-radius: 5px;

margin-bottom: 20px;

}

.item-list {

list-style: none;

padding: 0;

}

.item {

background-color: #3f51b5;

color: white;

padding: 15px;

border-radius: 5px;

margin: 10px auto;

font-size: 20px;

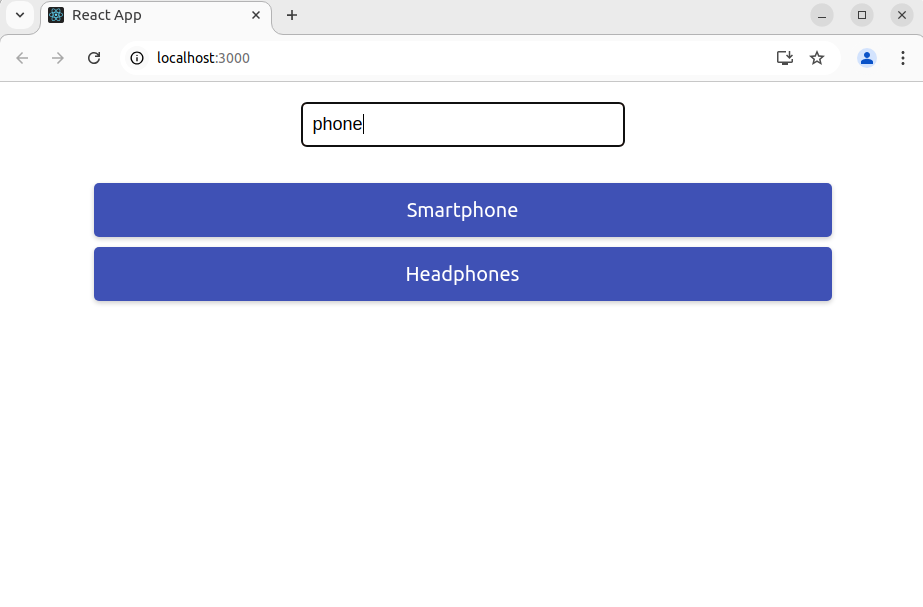
width: 80%;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

}

A screenshot of a computer

Description automatically generated

**RESULT:**

Thus, a react website to implement Search to Filter Items is created successfully.

|  |  |
| --- | --- |
| LA4 20.08.2024 | **React JS program to create a form** |

**AIM:**

To create a react webpage to create a form.

**ALGORITHM:**

1. Create a form with input fields for name and email.

2. Use state to manage the form values.

3. Update state in real-time as the user types.

4. Display the current form data below the form.

**CODE:**

**App.js**

import React, { useState } from 'react';

import './App.css';

const App = () => {

const [form, setForm] = useState({

name: '',

email: '',

});

const handleChange = (e) => {

const { name, value } = e.target;

setForm({ ...form, [name]: value });

};

return (

<div className="App">

<form className="form">

<label>

Name:

<input

type="text"

name="name"

value={form.name}

onChange={handleChange}

className="input"

/>

</label>

<br />

<label>

Email:

<input

type="email"

name="email"

value={form.email}

onChange={handleChange}

className="input"

/>

</label>

</form>

<div className="form-data">

<h2>Real-Time Form Data:</h2>

<p><strong>Name:</strong> {form.name}</p>

<p><strong>Email:</strong> {form.email}</p>

</div>

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

}

.form {

margin-bottom: 20px;

}

.input {

padding: 10px;

font-size: 18px;

border: 2px solid #ff5722;

border-radius: 5px;

margin: 5px 0;

}

.form-data {

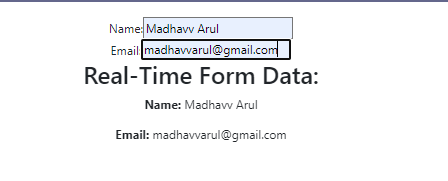
background-color: #e8f5e9;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}



**RESULT:**

Thus, a react website to create a form is created successfully.

|  |  |
| --- | --- |
| LA5 20.08.2024 | **Implementation of a navigation menu** |

**AIM:**

To create a website for implementation of Navigation Menu.

**ALGORITHM:**

1. Define a state variable to track the current view.

2. Create functional components for each page (e.g., Home, Services, Contact).

3. Implement buttons in the navigation bar to switch between views.

4. Each button should update the state to the corresponding view when clicked.

5. Use conditional rendering to display the component corresponding to the current view.

**CODE:**

**App.js**

import React, { useState } from 'react';

import './App.css';

// Define components for different pages

const Home = () => <h2>Home Page</h2>;

const Services = () => <h2>Services Page</h2>;

const Contact = () => <h2>Contact Page</h2>;

const App = () => {

const [currentView, setCurrentView] = useState('home'); // State to track current view

// Handler to change the view

const handleViewChange = (view) => {

setCurrentView(view);

};

return (

<div className="App">

<nav className="navbar">

<button className="nav-link" onClick={() => handleViewChange('home')}>Home</button>

<button className="nav-link" onClick={() => handleViewChange('services')}>Services</button>

<button className="nav-link" onClick={() => handleViewChange('contact')}>Contact</button>

</nav>

<div className="content">

{currentView === 'home' && <Home />}

{currentView === 'services' && <Services />}

{currentView === 'contact' && <Contact />}

</div>

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

}

.navbar {

background-color: #ff5722;

padding: 15px;

margin-bottom: 20px;

}

.nav-link {

background-color: #ffffff;

color: #ff5722;

border: none;

padding: 10px 20px;

margin: 0 10px;

cursor: pointer;

font-size: 18px;

border-radius: 5px;

}

.nav-link:hover {

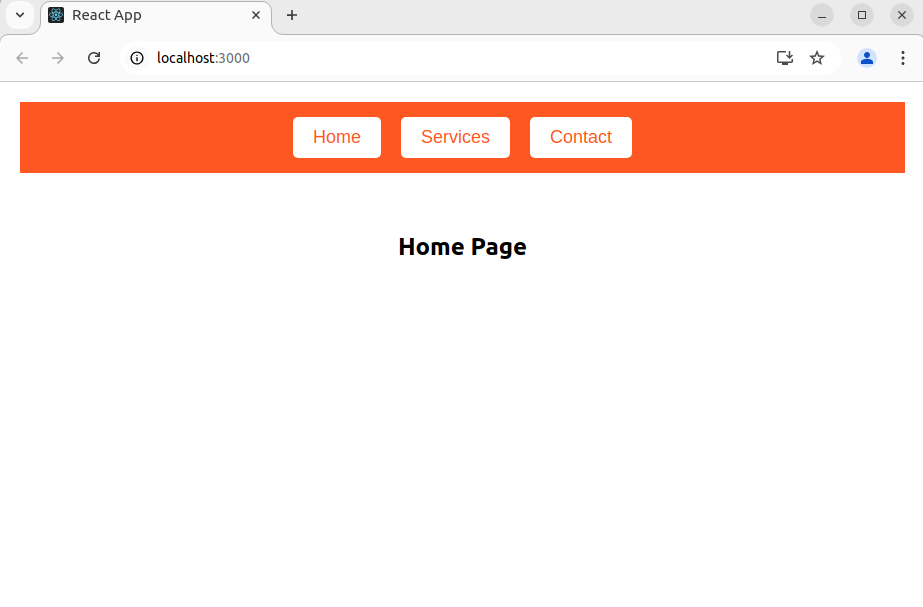
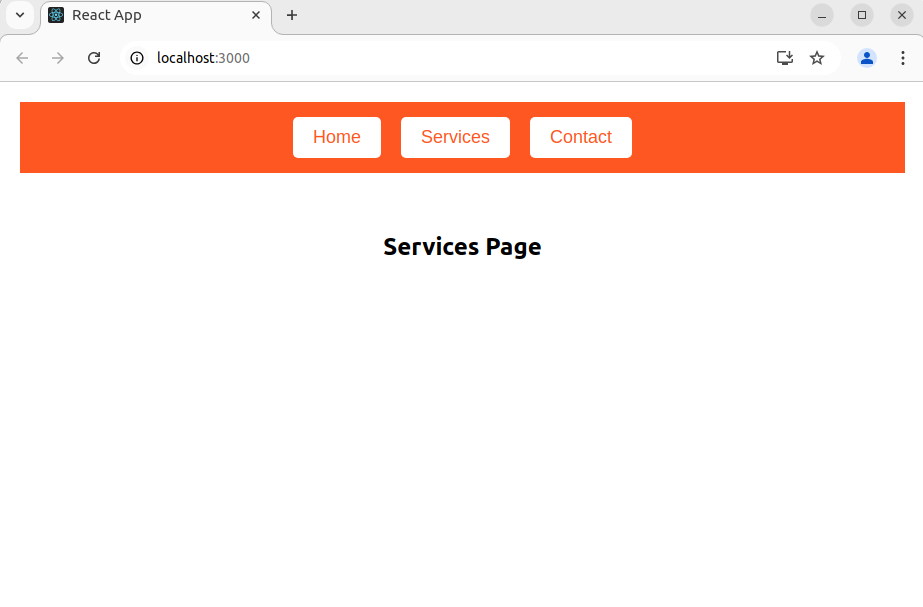
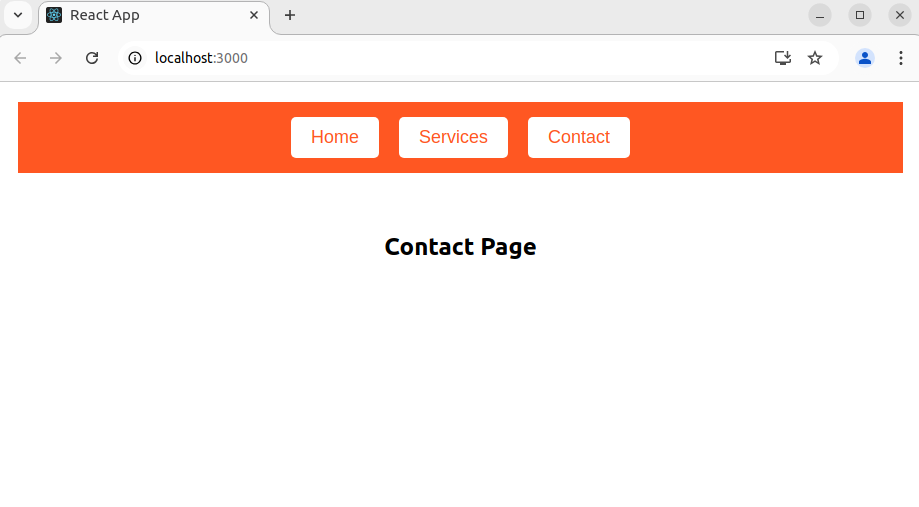
background-color: #fbe9e7;

}

.content {

padding: 20px;

}



**RESULT:**

Thus, a react website to implement navigation menu is created successfully.

|  |  |
| --- | --- |
| LA6 27.08.2024 | **Creation of a media player** |

**AIM:**

To create a media player using ReactJS

**ALGORITHM:**

1. Create a functional component that includes a `<video>` tag to display the video and a `<div>` for additional information.

2. Embed the video source URL in the `<video>` tag and set it to be responsive.

3. Apply CSS to style the video player, including borders, shadows, and responsive design.

4. Add a title and description below the video for context and enhance visual appeal.

5. Use media queries to ensure the video player adjusts correctly on different screen sizes.

**CODE:**

**App.js**

import React from 'react';

import './App.css';

const App = () => {

return (

<div className="App">

<h1 className="title">Video Player</h1>

<div className="video-container">

<video controls className="video-player">

<source src="https://www.w3schools.com/html/mov\_bbb.mp4" type="video/mp4" />

Your browser does not support the video tag.

</video>

<div className="video-description">

<p>This is a sample video description. Customize this area to add more information about the video content.</p>

</div>

</div>

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

background-color: #fafafa;

min-height: 100vh;

}

.title {

font-size: 2em;

margin-bottom: 20px;

color: #ff5722;

}

.video-container {

position: relative;

display: inline-block;

border: 2px solid #ff5722;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

overflow: hidden;

background-color: #000;

}

.video-player {

width: 100%;

height: auto;

display: block;

}

.video-description {

margin-top: 10px;

padding: 10px;

background-color: #ffffff;

border: 1px solid #ddd;

border-radius: 5px;

color: #333;

font-size: 1em;

}

.video-description p {

margin: 0;

}

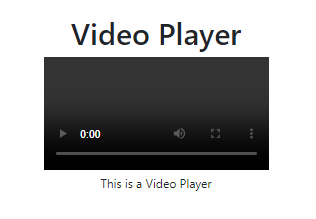
@media (max-width: 768px) {

.video-player {

width: 100%;

}

}



**RESULT:**

Thus, a react website to implement media player is created successfully.

|  |  |
| --- | --- |
| LA7 27.08.2024 | **Creation of a text editor** |

**AIM:**

To create a text editor Using ReactJS

**ALGORITHM:**

1. Define a state variable to manage the content of the editor.

2. Integrate the ReactQuill editor component and set its value to the state variable. Use a handler function to update the state as the content changes.

3. Display a live preview of the editor content by rendering the HTML directly below the editor.

4. Apply CSS to ensure the editor and preview areas are visually appealing and responsive.

**CODE:**

**App.js**

import React, { useState } from 'react';

import ReactQuill from 'react-quill';

import 'react-quill/dist/quill.snow.css'; // import styles

import './App.css';

const App = () => {

const [editorValue, setEditorValue] = useState('');

const handleChange = (value) => {

setEditorValue(value);

};

return (

<div className="App">

<header className="header">

<h1>React Quill Editor</h1>

</header>

<div className="editor-container">

<ReactQuill

value={editorValue}

onChange={handleChange}

className="editor"

/>

</div>

<div className="preview-container">

<h2>Preview</h2>

<div

className="preview"

dangerouslySetInnerHTML={{ \_\_html: editorValue }}

/>

</div>

</div>

);

};

export default App;

**App.css**

.App {

text-align: center;

padding: 20px;

font-family: Arial, sans-serif;

}

.header {

margin-bottom: 20px;

}

.editor-container {

margin: 20px auto;

max-width: 800px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

.editor {

height: 300px;

}

.preview-container {

margin: 20px auto;

max-width: 800px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

background-color: #ffffff;

}

.preview {

padding: 20px;

border: 1px solid #ddd;

border-radius: 5px;

min-height: 150px;

}

A screenshot of a computer

Description automatically generated

**RESULT:**

Thus, a react website to implement text editor is created successfully.

|  |  |
| --- | --- |
| LA8 03.09.2024 | **Creation of a basic HTML page** |

**AIM:**

To create a basic HTML webpage.

**ALGORITHM:**

1. Define the HTML structure with `<!DOCTYPE html>`, `<html>`, `<head>`, and `<body>`.

2. In `<head>`, add the `<title>` and a `<style>` block for CSS.

3. Style the `body`, `header`, `nav`, `main`, and `footer` using CSS for layout and design.

4. In `<body>`, create a `<header>` with the page title.

5. Add a `<nav>` with anchor links for navigation.

6. Create a `<main>` section with subsections for "Home," "About," and "Contact."

7. Add a `<footer>` with copyright information.

8. Close all tags.

**CODE:**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Basic Webpage</title>

<style>

/\* CSS Styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f9;

color: #333;

}

header {

background-color: #4CAF50;

color: white;

padding: 10px 0;

text-align: center;

}

nav {

text-align: center;

margin: 20px 0;

}

nav a {

margin: 0 15px;

text-decoration: none;

color: #4CAF50;

font-weight: bold;

}

nav a:hover {

color: #333;

}

main {

padding: 20px;

max-width: 800px;

margin: 0 auto;

}

section {

margin-bottom: 20px;

}

h2 {

color: #4CAF50;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 10px 0;

position: fixed;

bottom: 0;

width: 100%;

}

</style>

</head>

<body>

<header>

<h1>Welcome to My Basic Webpage</h1>

</header>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

<a href="#contact">Contact</a>

</nav>

<main>

<section id="home">

<h2>Home</h2>

<p>This is the home section of the basic webpage. You can navigate to different sections using the links above.</p>

</section>

<section id="about">

<h2>About</h2>

<p>This section provides information about the purpose of this webpage. It's a simple demo of HTML and CSS integration.</p>

</section>

<section id="contact">

<h2>Contact</h2>

<p>If you have any questions, feel free to reach out!</p>

</section>

</main>

<footer>

<p>&copy; 2024 Basic Webpage. All rights reserved.</p>

</footer>

</body>

</html>

**OUTPUT:**

A screenshot of a webpage

Description automatically generated

**RESULT:**

Thus, a basic html webpage is created.

|  |  |
| --- | --- |
| LA9 10.09.2024 | **Form validation** |

**AIM:**

To create a web application and implement form validation.

**ALGORITHM:**

1. Start with `<!DOCTYPE html>` and define the structure using `<html>`, `<head>`, and `<body>` tags.

2. Set the page title as "Registration Form" inside the `<title>` tag.

3. Add a `<style>` block to design the form and page elements.

4. In the `<body>`, create a `<div>` with class `form-container` to hold the form.

5. Add a form title with `<h2>`.

6. Inside the form, create input fields for Name, Email, and Password with validation (`required`, `minlength`).

7. Add a submit button with hover effects.

8. Close all tags to complete the structure.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Registration Form</title>

<style>

body {

font-family: Arial, Helvetica, sans-serif;

margin: 0;

padding: 20px;

background-color: bisque;

}

.form-container {

width: 400px;

margin: 0 auto;

background-color: aqua;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

color: black;

}

label {

display: block;

margin-top: 10px;

font-weight: bold;

}

input {

width: 100%;

padding: 5px;

margin: 8px 0;

border: 1px solid brown;

border-radius: 4px;

}

button {

width: 100%;

padding: 10px;

background-color: red;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: bisque;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Registration Form</h2>

<form action="#" method="POST">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="password">Password:</label>

<input type="password" id="password" name="password" minlength="8" required>

<button type="submit">Submit</button>

</form>

</div>

</body>

</html>

**OUTPUT:**

A screen shot of a registration form

Description automatically generated

**RESULT:**

Thus, a webpage to implement form validation is created.

|  |  |
| --- | --- |
| LA10 17.09.2024 | **Fetching data from server with AJAX** |

**AIM:**

To illustrate fetching data from server with AJAX.

**ALGORITHM:**

1. Define the `loadDoc()` function to handle the AJAX request.

2. Create a new instance of the `XMLHttpRequest` object to communicate asynchronously with the server.

3. Set the `onreadystatechange` function to monitor changes in the request state:

- `readyState == 4` ensures the request is complete.

- `status == 200` checks if the request was successful.

4. If both conditions are met, retrieve the data using `this.responseText`.

5. Update the inner HTML content of the element with the ID "demo" with the server response.

6. Use `open()` to configure the request (HTTP method, file, and asynchronous mode).

7. Use `send()` to send the request to the server.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AJAX Example</title>

</head>

<body>

<h2>AJAX Example</h2>

<button type="button" onclick="loadDoc()">Fetch Data</button>

<p id="demo"></p>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

}

</script>

</body>

</html>

**Explanation:**

The `loadDoc()` function initiates an AJAX request to fetch data from `ajax\_info.txt`.

Once the data is successfully retrieved (i.e., when the `readyState == 4` and `status == 200`), the content of the paragraph element with the ID `demo` is updated with the server's response.

**RESULT:**

Thus the AJAX function is illustrated.

|  |  |
| --- | --- |
| LA11 01.10.2024 | **Implementation of a semester fee form** |

**AIM:**

To create a webpage implementing a semester fee form.

**ALGORITHM:**

1. HTML Structure for the Form:

- Create a form with fields for Student Name, Student ID, and Department.

- Include a Submit button to trigger validation and display the fee record.

2. JavaScript Validation:

- Check if all fields are filled.

- Ensure the Student ID is alphanumeric and at least 5 characters long.

- Display an alert for any validation issues.

3. Display the Fee Record:

- If validation passes, display the fee record with the student’s details and a "Pay" button.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Semester Fee Submission Form</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f0f0f0;

}

.form-container {

width: 400px;

margin: 0 auto;

padding: 20px;

background-color: white;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 10px;

font-weight: bold;

}

input, select {

width: 100%;

padding: 8px;

margin: 8px 0;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

width: 100%;

padding: 10px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.fee-record {

display: none;

margin-top: 20px;

padding: 20px;

background-color: #e2f0d9;

border: 1px solid #ccc;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Shiv Nadar University</h2>

<h3>Semester Fee Submission Form</h3>

<form id="feeForm" onsubmit="return validateForm()">

<label for="name">Student Name:</label>

<input type="text" id="name" name="name" required>

<label for="studentId">Student ID:</label>

<input type="text" id="studentId" name="studentId" required pattern="[A-Za-z0-9]{5,}">

<label for="department">Department:</label>

<select id="department" name="department" required>

<option value="">--Select Department--</option>

<option value="IoT">IoT</option>

<option value="AIDS">AIDS</option>

<option value="Cyber Security">Cyber Security</option>

</select>

<button type="submit">Submit</button>

</form>

<div id="feeRecord" class="fee-record">

<h4>Fee Record for <span id="displayName"></span></h4>

<p><strong>ID:</strong> <span id="displayId"></span></p>

<p><strong>Department:</strong> <span id="displayDept"></span></p>

<p><strong>Total Fee:</strong> ₹ 1,50,000</p>

<button onclick="payFee()">Pay</button>

</div>

</div>

<script>

function validateForm() {

var name = document.getElementById("name").value;

var studentId = document.getElementById("studentId").value;

var department = document.getElementById("department").value;

if (name == "" || studentId == "" || department == "") {

alert("Please fill out all fields.");

return false;

}

var idPattern = /^[A-Za-z0-9]{5,}$/;

if (!idPattern.test(studentId)) {

alert("Student ID must be alphanumeric and at least 5 characters long.");

return false;

}

displayFeeRecord();

return false; // Prevent form submission to server

}

function displayFeeRecord() {

var name = document.getElementById("name").value;

var studentId = document.getElementById("studentId").value;

var department = document.getElementById("department").value;

document.getElementById("displayName").innerText = name;

document.getElementById("displayId").innerText = studentId;

document.getElementById("displayDept").innerText = department;

document.getElementById("feeRecord").style.display = "block";

}

function payFee() {

alert("Payment processing...");

}

</script>

</body>

</html>

**OUTPUT:**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**RESULT:**

Thus a webpage implementing a semester fee form is created.