Ex.No:- 1	Write a program to create a simple webpage using HTML
Date:-	

AIM:

To design a simple HTML webpage that includes headings, paragraphs, hyperlinks, lists, a button, centered text, and a table.

ALGORITHM:

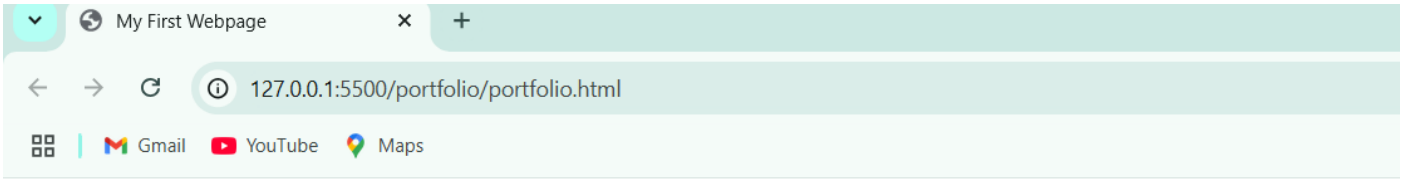
1. Start Visual Studio Code.
2. Create a new folder for your project (e.g., MyWebpage).
3. Inside VS Code, open that folder (File → Open Folder).
4. Create a new file and save it as index.html.
5. Write the following HTML code:
6. Save the file (Ctrl + S).
7. Right-click on the index.html file and choose "Open with Live Server" *(you must have the Live Server extension installed)*.
8. The page will open in your browser.
9. Stop.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
 <title>My First Webpage</title>
</head>
<body>
 <h1>Welcome to My Webpage</h1>
 <h2>This is a subheading (H2)</h2>
 <h3>This is H3 heading</h3>
 <h4>This is H4 heading</h4>
 <h5>This is H5 heading</h5>
 <h6>This is H6 heading</h6>
 <p>This is a paragraph of text on my webpage.</p>
 <a href="https://www.google.com">Visit Google page </a><br><br>
 <h3>Ordered List:</h3>
 <ol>
  <li>First item</li>
  <li>Second item</li>
```

```
    <li>Third item</li>
  </ol>
  <h3>Unordered List:</h3>
  <ul>
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
  </ul>
  <center>This text is centered.</center>
  <br>
  <button type="button">Click Me</button>
  <h3>Simple Table:</h3>
  <table border="1">
    <tr>
      <th>Name</th>
      <th>Age</th>
    </tr>
    <tr>
      <td>Alice</td>
      <td>25</td>
    </tr>
    <tr>
      <td>Bob</td>
      <td>30</td>
    </tr>
  </table>
 </body>
</html>
```

OUTPUT:



Welcome to My Webpage

This is a subheading (H2)

This is H3 heading

This is H4 heading

This is H5 heading

This is H6 heading

This is a paragraph of text on my webpage.

Visit Google page

Ordered List:

1. First item
2. Second item
3. Third item

Unordered List:

- Apple
- Banana
- Cherry

This text is centered.

Click Me

Simple Table:

Name	Age
Alice	25
Bob	30

RESULT:-

Thus, the HTML code has been executed and verified successfully.

Ex.No: 02	Write a program to create a website using HTML CSS and JavaScript
Date:	

AIM:

To create a basic interactive website using HTML for structure, CSS for styling and JavaScript for dynamic behaviour.

ALGORITHM:-

1. Start
2. Create the structure of the web page using HTML:
   - Add <header> with a title and navigation menu.
   - Add <main> section with a welcome message and a button.
   - Add <footer> with copyright.
3. Style the webpage using CSS:
   - Set the background colour, font and layout.
   - Style the header , navigation links , buttons and footer.
4. Add interactivity using JavaScript:
   - Write a function showMessage() to display a message when the button is clicked.
   - Use the onlick event in the HTML to trigger the function.
5. Link the style.css and script.js files to the index.html file.
6. Save all files and open index.html in a browser to view the website.
7. End.

PROGRAM:

index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta cgarset="UTF-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title> My Simple  Website</title>
  <link rel="stylesheet" href="style.css"/>
 </head>
 <body>
  <header>
   <h1>Welcome to my website</h1>
   <nav>
    <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
```

```
        </ul>
      </nav>
    </header>
    <main>
      <section>
        <h2>Hello there!</h2>
        <p>This is a simple website made with html,css and javascript.</p>
        <button onclick="showmessage()">Click me..!</button>
      </section>
    </main>
    <footer>
      <p>&copy; 2025 My website. All rights reserved.</p>
    </footer>
    <script src="script.js"></script>
  </body>
</html>
```

Style.css

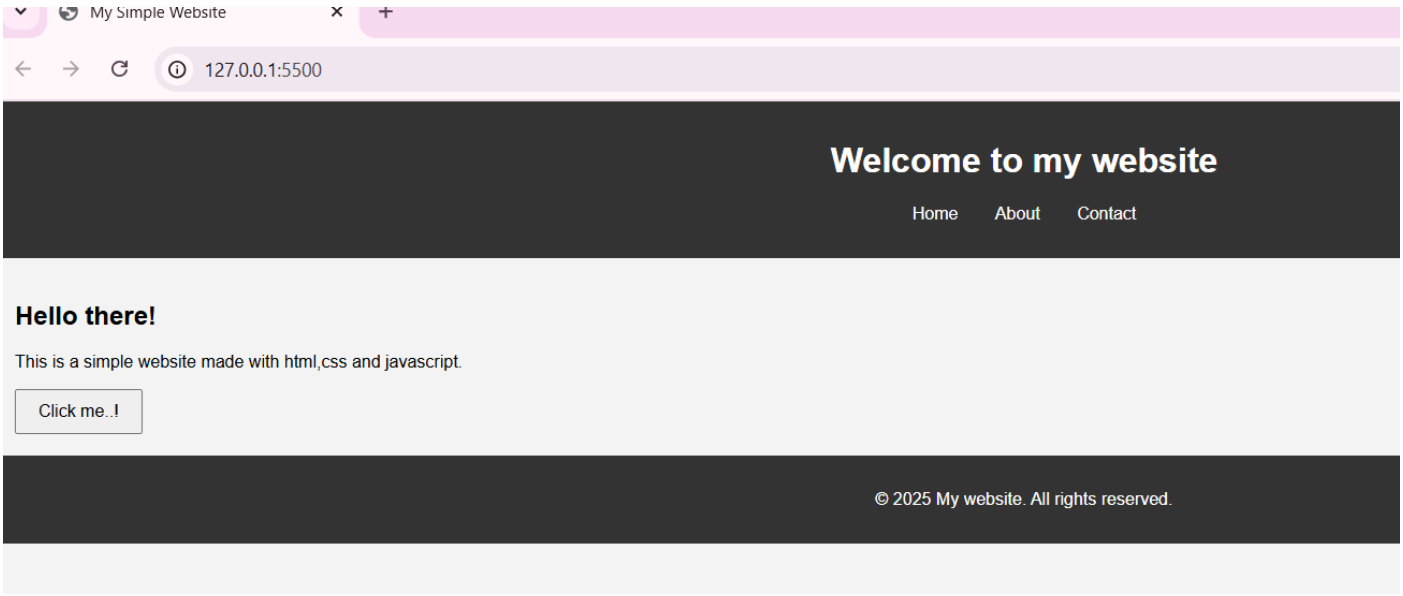
```
body{
    font-family: Arial, Helvetica, sans-serif;
    background-color: whitesmoke;
}
header{
    background-color: #333;
    color:white;
    padding: 1rem;
    text-align:center;
}
nav ul{
    list-style: none;
    padding: 0;
}
nav ul li{
    display: inline;
    margin: 0 15px;
```

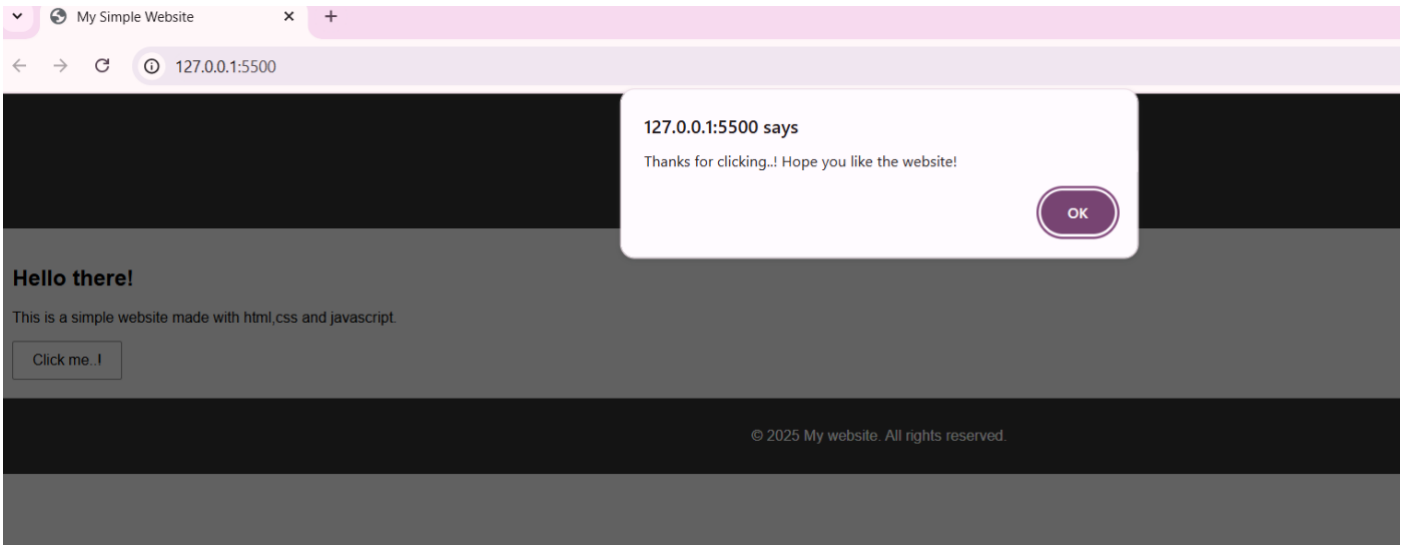
```
}
nav ul li a{
  color: white;
  text-decoration: none;
}
main{
  padding: 20px;
  font-size: 16px;
}
button{
  padding: 10px 20px;
  font-size: 16px;
  cursor:pointer;
}
footer{
  background-color: #333;
  color:white;
  text-align: center;
  padding: 1rem;
}
```

script.js

```
function showmessage(){
  alert("Thanks for clicking..! Hope you like the website!");
}
```

OUTPUT:





RESULT:-

Thus, the HTML,CSS,JavaScript code has been executed and verified successfully.

Ex.No:- 3	Write a program to build a chat module using HTML, CSS and JavaScript
Date:-	

AIM:

To design and develop a simple chat module using HTML,CSS and JavaScript .

ALGORITHM:-

1. Start
2. Create the HTML layout for the chat module in container for chat message , An input field for typing messages,A button to send the message.
3. Apply CSS to style the chat interface .Make it responsive and visually clean.
4. Write JavaScript to handle chat logic.
5. Repeat step 4 as long as the user inputs messages.
6. End.

PROGRAM:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
     content="width=device-width, initial-scale=1.0">
  <title>ChatBot</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="chatBot">
    <header>
      <h2>ChatBot</h2>
      <span alt="Close"
         id="cross"
         onclick="cancel()">X</span>
    </header>
    <ul class="chatbox">
      <li class="chat-incoming chat">
        <p>Hey! How can I assist you today?</p>
      </li>
    </ul>
```

```
      <div class="chat-input">
        <textarea rows="0" cols="17"
                placeholder="Enter a message..."></textarea>
        <button id="sendBTN">Send</button>
      </div>
    </div>
    <script src="script.js" defer></script>
</body>
</html>
```

Style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
  font-family: "Poppins", sans-serif;
  font-weight: 400;
  font-style: normal;
  background-color: #f7f7f7;
}
.chatBot {
  border: 3px solid #2F8D46;
  border-radius: 10px;
  margin: 50px auto;
  overflow: hidden;
  width: 500px;
  overflow-y: clip;
  height: 600px;
  background: rgb(255, 255, 255) url(gfg-gg-logo.svg);
  background-size: contain;
  box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
```

```
  background-repeat: no-repeat;
  background-position: center;
}
header {
  background-color: #2F8D46;
  text-align: center;
  padding: 10px 0;
  border-radius: 7px 7px 0 0;
}
header h2 {
  color: #fff;
  margin: 0;
}
.chatbox {
  padding: 15px;
  list-style: none;
  overflow-y: auto;
  height: 400px;
}
.chatbox li {
  margin-bottom: 10px;
}
.chat p {
  padding: 10px;
  border-radius: 10px;
  max-width: 70%;
  word-wrap: break-word;
}
.chat-outgoing p {
  background-color: #162887;
  align-self: flex-end;
  color: #fff;
}
.chat-incoming p {
```

```
  background-color: #eaeaea;
}
.chat-input {
  padding: 10px;
  border-top: 1px solid #ccc;
}
.chat-input textarea {
  width: 522px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 7px;
  resize: none;
  outline: none;
  overflow-y: scroll;
  background-color: #dcdcdc85;
  font-size: 16px;
  color: green;
  font-weight: 600;
  margin-top: -10px;
  margin-left: -15px;
  height: 71px;
}
#cross {
  float: right;
  position: relative;
  top: -38px;
  left: -15px;
  cursor: pointer;
  color: white;
  font-weight: bolder;
  font-size: 28px;
}
#cross:hover {
  color: red;
```

```
  transition: all .5s;
}
.chatbox .chat p.error {
  color: #ffffff;
  background-color: #ff3737e8;
}
#sendBTN {
  width: 100%;
  padding: 8px;
  border: 0;
  outline: none;
  font-size: 20px;
  font-weight: 600;
  border-radius: 7px;
  background-color: #2F8D46;
  cursor: pointer;
  color: white;
  margin-top: 12px;
}
.lastMessage {
  margin-top: 50px;
  font-size: 35px;
  font-weight: 600;
  color: darkgreen;
  margin-left: 550px;
}
```

Index.js

```
// script.js
const chatInput =
    document.querySelector('.chat-input textarea');
const sendChatBtn =
    document.querySelector('.chat-input button');
const chatbox = document.querySelector(".chatbox");
```

```
let userMessage;
const API_KEY =
  "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";
//OpenAI Free APIKey
const createChatLi = (message, className) => {
  const chatLi = document.createElement("li");
  chatLi.classList.add("chat", className);
  let chatContent =
    className === "chat-outgoing" ? `<p>${message}</p>` : `<p>${message}</p>`;
  chatLi.innerHTML = chatContent;
  return chatLi;
}
const generateResponse = (incomingChatLi) => {
  const API_URL = "https://api.openai.com/v1/chat/completions";
  const messageElement = incomingChatLi
  .querySelector("p");
  const requestOptions = {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${API_KEY}`
    },
    body: JSON.stringify({
      "model": "gpt-3.5-turbo",
      "messages": [
        {
          role: "user",
          content: userMessage
        }
      ]
    })
  };

  fetch(API_URL, requestOptions)
```

```
        .then(res => {
            if (!res.ok) {
                throw new Error("Network response was not ok");
            }
            return res.json();
        })
        .then(data => {
            messageElement
            .textContent = data.choices[0].message.content;
        })
        .catch((error) => {
            messageElement
            .classList.add("error");
            messageElement
            .textContent = "Oops! Something went wrong. Please try again!";
        })
        .finally(() => chatbox.scrollTo(0, chatbox.scrollHeight));
};
const handleChat = () => {
    userMessage = chatInput.value.trim();
    if (!userMessage) {
        return;
    }
    chatbox
    .appendChild(createChatLi(userMessage, "chat-outgoing"));
    chatbox
    .scrollTo(0, chatbox.scrollHeight);
    setTimeout(() => {
        const incomingChatLi = createChatLi("Thinking...", "chat-incoming")
        chatbox.appendChild(incomingChatLi);
        chatbox.scrollTo(0, chatbox.scrollHeight);
        generateResponse(incomingChatLi);
    }, 600);
}
```

```
sendChatBtn.addEventListener("click", handleChat);
function cancel() {
    let chatbotcomplete = document.querySelector(".chatBot");
    if (chatbotcomplete.style.display != 'none') {
        chatbotcomplete.style.display = "none";
        let lastMsg = document.createElement("p");
        lastMsg.textContent = 'Thanks for using our Chatbot!';
        lastMsg.classList.add('lastMessage');
        document.body.appendChild(lastMsg)
    }
}
```

Output:-

RESULT:-

Thus, the HTML,CSS,JavaScript code has been executed and verified successfully.

Ex.no:- 4	Write a program to create a simple calculator Application using React JS.
Date:-	

AIM:-

To create a calculator that performs basic arithmetic operations using React components and state management.

ALGORITHM:-

1. Create a calculatorTitle.js file for showing the title of the calculator and paste the code given below for this file.
2. Then create a file outputScreenRow.js for taking input and showing the output of the calculation, code of this file is given below.
3. Create an outputScreen.js file and import the outputScreenRow.js file. The code of this file is given below.
4. Create a button.js file
5. create a calculator.js file and import calculatorTitle.js, outputScreen.js, and button.js files.
6. Inside the index.js file import, the calculator.js file

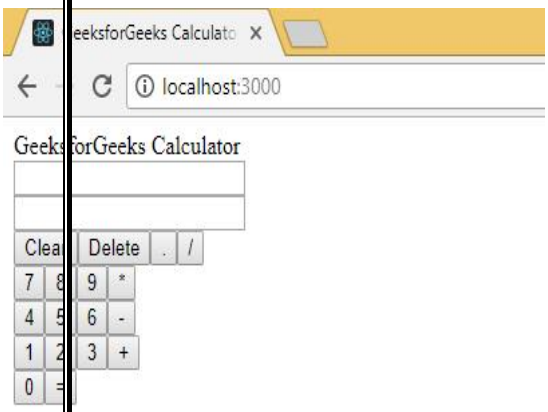
PROGRAM:-

calculatorTitle.js

```
import React from "react";
const CalculatorTitle = (props) => {
   return (
   <div className="calculator-title">{props.value}</div>
   );
};
export default CalculatorTitle;
import React from "react";
const OutputScreenRow = () => {
   return (
      <div className="screen-row">
         <input type="text" readOnly />
      </div>
   );
};
export default OutputScreenRow;
import React from "react";
import OutputScreenRow from "./outputScreenRow.js"; // Import Output Screen Row.
```

```
// Functional Component.
// Use to hold two Screen Rows.
const OutputScreen = () => {
   return (
      <div className="screen">
         <OutputScreenRow />
         <OutputScreenRow />
      </div>
   );
};
export default OutputScreen; // Export Output Screen.
// button.js File
import React from "react"; // Import React (Mandatory Step)

// Create our Button component as a functional component.
const Button = (props) => {
   return (
   <input type="button" value={props.label} />
   );
};
export default Button; // Export our button component
// calculator.js File
// Imports.
import React from "react";
import CalculatorTitle from "./calculatorTitle.js";
import OutputScreen from "./outputScreen.js";
import Button from "./button.js";

class Calculator extends React.Component {
   render() {
      return (
         <div className="frame">
            <CalculatorTitle value="GeeksforGeeks Calculator" />
```

```
        <div class="mainCalc">
          <OutputScreen />
          <div className="button-row">
            <Button label={"Clear"} />
            <Button label={"Delete"} />
            <Button label={"."} />
            <Button label={"/"} />
          </div>
          <div className="button-row">
            <Button label={"7"} />
            <Button label={"8"} />
            <Button label={"9"} />
            <Button label={"*"} />
          </div>
          <div className="button-row">
            <Button label={"4"} />
            <Button label={"5"} />
            <Button label={"6"} />
            <Button label={"-"} />
          </div>
          <div className="button-row">
            <Button label={"1"} />
            <Button label={"2"} />
            <Button label={"3"} />
            <Button label={"+"} />
          </div>
          <div className="button-row">
            <Button label={"0"} />
            <Button label={"="} />
          </div>
        </div>
      </div>
    );
  }
```



}

export default Calculator;

ndex.js File

import React from "react";

import ReactDOM from "react-dom";

import Calculator from "./components/calculator.js";

ReactDOM.render(<Calculator />, document.getElementById("root"));

OUTPUT:-

RESULT:-

Thus, the simple calculator application using React JS has been built and tested successfully

Ex.no:-5	Write a program to create a voting application using React JS
Date:-	

AIM:-

To create a voting app where users can vote for options and view live results using React state.

ALGORITHM:-

1. Start
2. Create React project
3. Create options as buttons or cards
4. Use useState() to track vote counts
5. On vote, increment respective count
6. Show total and percentage votes
7. Reset option (optional)
8. End

PROGRAM:-

App.js

```
import React,{Component} from 'react';
import './App.css';

class App extends Component{
	constructor(props){
		super(props);
		this.state = {
			languages : [
				{name: "Jvascript", votes: 0},
				{name: "Python", votes: 0},
				{name: "kotlin", votes: 0},
				{name: "Java", votes: 0}
			]
		}
	}

	vote (i) {
		let newLanguages = [...this.state.languages];
		newLanguages[i].votes++;
		function swap(array, i, j) {
			var temp = array[i];
			array[i] = array[j];
			array[j] = temp;
		}
		this.setState({languages: newLanguages});

	}

	render(){
		return(
			<>
<h1>Vote Your Language!</h1>
```

```
<div className="languages">
                                        {
                                                this.state.languages.map((lang, i) =>
                    <div key={i} className="language">
                    <div className="voteCount">
                                                                        {lang.votes}
                                                                </div>
<div className="languageName">
                                                                {lang.name}
                                                                </div>
                                                                <button onClick={this.vote.bind(this, i)}>Click
Here</button>
            </div>
            )}
            </div>
        </>
            );
            }
}
export default App;
```

```
*{
  margin: 0;
  padding: 0;
}

body {
  text-align: center;
  color: #222;
  font-size: 24px;
  font-family: sans-serif;
}

h1 {
  margin: 30px;
}

.languages {
  height: 400px;
  width: 400px;
  margin: 10px auto;
  display: flex;
  flex-direction: column;
}

.language {
  flex: 1;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 40px;
  background-color: blanchedalmond;
  border: 1px solid #222;
```

```
  margin: 2px;
}

.voteCount {
  border-radius: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.language button {
  color: blueviolet;
  background-color: #0000;
  border: none;
  font-size: 30px;
  outline: none;
  cursor: pointer;
}
```

OUTPUT:-

RESULT:-

Thus, the voting application using React JS has been implemented and verified successfully.

Ex.no:- 6	Write a program to create and Build a Password Strength Check using Jquery
Date:-	

AIM:-

To validate the strength of a user's password input based on criteria like length, case, symbols using jQuery.

ALGORITHM:-

1. Start
2. Create HTML form with password field
3. Include jQuery
4. Write a function to: Detect input change,Check criteria (length, uppercase, number, symbol),Update strength indicator (weak, strong)
5. Display color-coded result
6. End

PROGRAM:-

```
<!-- Update progress-bar whenever input field is updated.-->

<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="utf-8" />

   <!-- Required CDN's -->

   <link rel="stylesheet"

        href=

"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

   <script src=

"https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">

 </script>

   <script src=

"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js">

 </script>

</head>

<style>

   .container-fluid {

      background-color: #ffffff;

      border-radius: 8px;

      border: 1px solid lightgrey;

      padding: 16px;

      -webkit-box-shadow: 0px 0px 12px 2px rgba(0, 0, 0, 0.75);
```

```
    -moz-box-shadow: 0px 0px 12px 2px rgba(0, 0, 0, 0.75);
    box-shadow: 0px 0px 12px 2px rgba(0, 0, 0, 0.75);
  }
  .input-group {
    width: 80%;
    height: auto;
    padding: 4px;
  }
  .progress {
    height: 4px;
  }
  .progress-bar {
    background-color: green;
  }
</style>
<body>
  <br>
  <br>
  <div class="container">
    <div class="row">
      <div class="col-sm-3">   </div>
      <div class="col-sm-6">
        <div class="container-fluid">
          <center>
            <h2 class="text-success">
              Enter Password:
            </h2>
            <br>
            <br>
            <div class="form-group">
              <div class="input-group">
                <span class="input-group-addon">
                  <i class="glyphicon glyphicon-user">
                  </i>
```

```
      </span>
       <input id="email"
            type="text"
            class="form-control"
            name="email"
            placeholder="User-Email">
     </div>
     <div class="input-group">
       <span class="input-group-addon">
        <i class="glyphicon glyphicon-lock">
        </i>
      </span>
       <input id="password"
            type="password"
            class="form-control"
            name="password"
            placeholder="Enter Password">
     </div>
   </div>
   <div class="input-group">
     <div class="progress">
       <div class="progress-bar"
          role="progressbar"
          aria-valuenow="0"
          aria-valuemin="0"
          aria-valuemax="100"
          style="width:0%">
       </div>
     </div>
   </div>
   <br>
   <br>
   <div class="input-group">
     <button class="btn btn-success btn-block">
```

```
                    Register Now!
                  </button>
                 </div>
               </center>
               <br>
             </div>
           </div>
           <div class="col-sm-3">   </div>
         </div>
       </div>
       <script>
         var percentage = 0;
         function check(n, m) {
           if (n < 6) {
             percentage = 0;
             $(".progress-bar").css("background", "#dd4b39");
           } else if (n < 8) {
             percentage = 20;
             $(".progress-bar").css("background", "#9c27b0");
           } else if (n < 10) {
             percentage = 40;
             $(".progress-bar").css("background", "#ff9800");
           } else {
             percentage = 60;
             $(".progress-bar").css("background", "#4caf50");
           }          if ((m.match(/[a-z]/) != null))
           {
             percentage += 10;
           }
           if ((m.match(/[A-Z]/) != null))
           {
             percentage += 10;
           }
                 if ((m.match(/0|1|2|3|4|5|6|7|8|9/) != null))
```

```
        {
            percentage += 10;
        }
            if ((m.match(/\W/) != null) && (m.match(/\D/) != null))
        {
            percentage += 10;
        }
        $(".progress-bar").css("width", percentage + "%");
    }
    $(document).ready(function() {
        $("#password").keyup(function() {
            var m = $(this).val();
            var n = m.length;
            check(n, m);
        });
    });
  </script>
</body>
</html>
```

Output:

RESULT:-

Thus, the password strength checker using jQuery has been created and validated successfully.

Ex.no:- 7 Date:-	Write a program to create and Build a star rating system using Jquery

AIM:-

To implement a star-based rating system using jQuery with mouse hover and click events.

ALGORITHM:-

1. Start
2. Create HTML with 5 star icons (★)
3. Include jQuery
4. Add event listeners:
    - hover to highlight stars
    - click to select rating
5. Show selected rating visually
6. Store/print rating value
7. End

PROGRAM:-

```
<!DOCTYPE html>
<html>

<head>
  <title>rating</title>
  <link rel="stylesheet"
      type="text/css"
      href="jquery.rateyo.min.css">
</head>

<body>
  <div style="width: 600px; margin: 30px auto">
    <div id="rateYo"></div>
  </div>
  <script type="text/javascript"
      src="jquery.min.js"></script>
  <script type="text/javascript"
      src="jquery.rateyo.min.js"></script>
  <script>
    $("#rateYo").rateYo({
      rating: 1.5,
      spacing: "10px",
      numStars: 5,
      minValue: 0,
      maxValue: 5,
      normalFill: 'black',
      ratedFill: 'orange',
    })
  </script>
```

```
</body>
</html>
```

OUTPUT

RESULT:-

Thus, the star rating system using jQuery has been designed and executed successfully.

Ex no:- 8	Create a Simple Login form using React JS
Date:-	

AIM:-

To build a functional login form using React with form validation and conditional rendering.

ALGORITHM:-

1. Start
2. Create React component with email and password fields
3. Use useState() for form data and error messages
4. On submit: Validate input,Show error or success
5. Optionally simulate login logic
6. End

PROGRAM:-

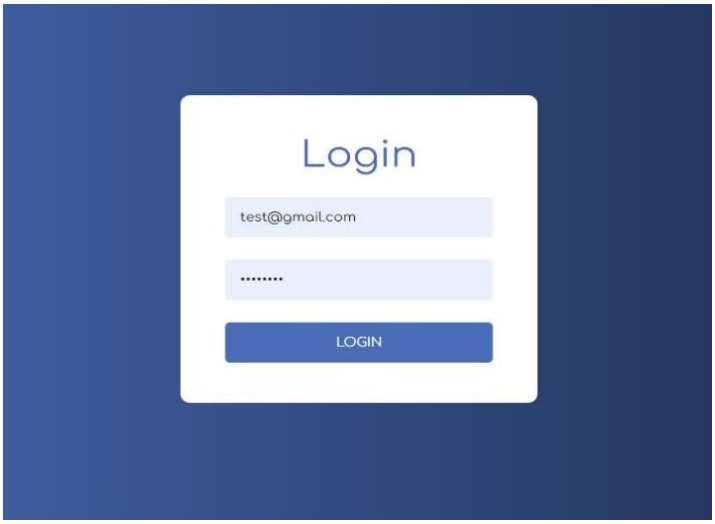
LoginForm.js

```
import React, { useState } from 'react';
const LoginForm = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [errors, setErrors] = useState({});
  const validate = () => {
    const errors = {};
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!email.trim()) {
      errors.email = 'Email is required';
    } else if (!emailRegex.test(email)) {
      errors.email = 'Enter a valid email';
    }
    if (!password.trim()) {
      errors.password = 'Password is required';
    }
    return errors;
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    const formErrors = validate();
    setErrors(formErrors);
    if (Object.keys(formErrors).length === 0) {
      console.log('Form submitted:', { email, password });
      // Reset form
      setEmail('');
      setPassword('');
    }
  };
  return (
    <div className="login-container">
      <h2>Login</h2>
      <form onSubmit={handleSubmit} className="login-form">
        <div>
```

```
        <label>Email:</label>
        <input
          type="text"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
        {errors.email && <span className="error">{errors.email}</span>}
      </div>
      <div>
        <label>Password:</label>
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
        {errors.password && <span className="error">{errors.password}</span>}
      </div>
      <button type="submit">Login</button>
    </form>
  </div>
 );
};
export default LoginForm;
App.js
import React from 'react';
import LoginForm from './LoginForm';
import './App.css';
function App() {
 return (
  <div className="App">
   <LoginForm />
  </div>
 );
}
export default App;
App.css
body {
 font-family: Arial, sans-serif;
 background-color: #f4f4f4;
}
.login-container {
 max-width: 400px;
 margin: 80px auto;
 padding: 30px;
 background: white;
 border-radius: 10px;
 box-shadow: 0 5px 15px rgba(0,0,0,0.1);
}
.login-form div {
 margin-bottom: 15px;
}
.login-form label {
 display: block;
```

```
  margin-bottom: 5px;
 }
 .login-form input {
  width: 100%;
  padding: 8px;
  box-sizing: border-box;
 }
 .login-form button {
  width: 100%;
  padding: 10px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
  border-radius: 5px;
 }
 .error {
  color: red;
  font-size: 0.8em;
 }
```

OUTPUT:-



RESULT:-

Thus, the login form using React JS has been implemented and tested successfully.

Ex.no:- 9	Using the CMS users must be able to design a web page using the drag and drop method
Date:-	

Aim:

To develop a web-based Content Management System (CMS) that allows users to design custom web pages using a drag-and-drop interface, enabling them to place and organize components such as text boxes, images, and buttons.

Algorithm:

1. **Start the CMS application**
2. Display a toolbox with pre-defined components (e.g., Button, Image, Text block)
3. Allow user to **drag** a component from the toolbox
4. Detect the **drop** event in the page builder area
5. On drop:
   - Clone the dragged component
   - Append it to the drop area
6. Enable users to rearrange, remove, or customize components
7. Optionally, allow saving/exporting the layout
8. **End**

PROGRAM:-

1. App.js

```
jsx
Copy code
import React, { useState } from 'react';
import './App.css';

const componentsList = [
 { type: 'text', label: 'Text Box' },
 { type: 'button', label: 'Button' },
 { type: 'image', label: 'Image' }
];

function App() {
 const [canvasItems, setCanvasItems] = useState([]);

 const handleDrop = (e) => {
  const componentType = e.dataTransfer.getData('componentType');
  const newItem = { type: componentType, id: Date.now() };
  setCanvasItems((items) => [...items, newItem]);
 };

 const handleDragOver = (e) => {
  e.preventDefault();
 };

 const renderComponent = (item) => {
  switch (item.type) {
   case 'text':
    return <input key={item.id} placeholder="Enter text" />;
   case 'button':
```

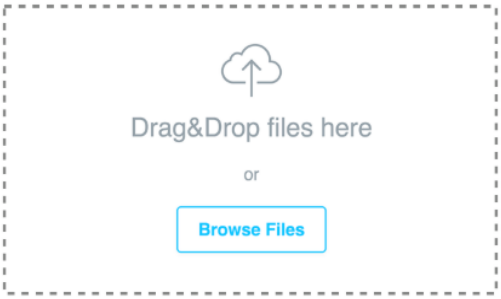
```
      return <button key={item.id}>Click Me</button>;
    case 'image':
      return <img key={item.id} src="https://via.placeholder.com/100" alt="Placeholder" />;
    default:
      return null;
  }
 };

 return (
  <div className="app">
   <div className="toolbox">
    <h3>Toolbox</h3>
    {componentsList.map((item) => (
     <div
      key={item.type}
      className="toolbox-item"
      draggable
      onDragStart={(e) => e.dataTransfer.setData('componentType', item.type)}
     >
      {item.label}
     </div>
    ))}
   </div>

   <div className="canvas" onDrop={handleDrop} onDragOver={handleDragOver}>
    <h3>Canvas</h3>
    {canvasItems.map((item) => renderComponent(item))}
   </div>
  </div>
 );
}

export default App;
```

2. App.css

```
css
Copy code
.app {
 display: flex;
 gap: 20px;
 padding: 20px;
}

.toolbox {
 width: 200px;
 padding: 15px;
 background: #f0f0f0;
 border-radius: 8px;
}

.toolbox-item {
 padding: 10px;
 background-color: #ddd;
 margin-bottom: 10px;
```

```
  cursor: grab;
  border-radius: 4px;
  text-align: center;
 }

 .canvas {
  flex: 1;
  min-height: 400px;
  padding: 15px;
  background: #ffffff;
  border: 2px dashed #aaa;
  border-radius: 8px;
 }
```

OUTPUT:-



RESULT:-

Thus, the drag-and-drop webpage design module using CMS features has been built and verified successfully.

Ex.no:-10	Connecting our TODO React js Projectwith Firebase
Date:-	

AIM:-

To connect a React TODO app to Firebase for real-time task storage, retrieval, and sync across devices.

ALGORITHM:-

1. Start
2. Setup Firebase project and enable Firestore
3. Create React TODO App with useState and useEffect
4. Install Firebase SDK and configure
5. Use Firestore for: Adding tasks (addDoc),Deleting tasks (deleteDoc),Fetching tasks (onSnapshot)
6. Sync UI with database changes
7. Deploy
8. End

PROGRAM:-

firebase.js in src/

```
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore';
const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "your-app.firebaseapp.com",
  projectId: "your-app-id",
  storageBucket: "your-app.appspot.com",
  messagingSenderId: "YOUR_SENDER_ID",
  appId: "YOUR_APP_ID"
};
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);
export { db };
```

App.js

```
import React, { useEffect, useState } from 'react';
import { db } from './firebase';
import { collection, addDoc, getDocs, deleteDoc, doc } from 'firebase/firestore';

function App() {
  const [task, setTask] = useState('');
  const [tasks, setTasks] = useState([]);

  const tasksCollectionRef = collection(db, 'tasks');

  const getTasks = async () => {
    const data = await getDocs(tasksCollectionRef);
    setTasks(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
  };

  const addTask = async () => {
    if (task.trim()) {
      await addDoc(tasksCollectionRef, { text: task });
      setTask('');
```

```
    getTasks();
  }
};

const deleteTask = async (id) => {
  const taskDoc = doc(db, 'tasks', id);
  await deleteDoc(taskDoc);
  getTasks();
};

useEffect(() => {
  getTasks();
}, []);

return (
  <div className="App">
    <h2>Firebase TODO</h2>
    <input
      value={task}
      onChange={(e) => setTask(e.target.value)}
      placeholder="Enter task..."
    />
    <button onClick={addTask}>Add Task</button>
    <ul>
      {tasks.map((t) => (
        <li key={t.id}>
          {t.text}
          <button onClick={() => deleteTask(t.id)}>X </button>
        </li>
      ))}
    </ul>
  </div>
);
}

export default App;
```

OUTPUT:-

RESULT:-

Thus, the TODO application integrated with React JS and Firebase has been developed and tested successfully.