1. A. For a Student management system, create a simple form and validate the following using JavaScript password e-mail username validation mobile number validation

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Management System</title>

</head>

<body>

<h2>Student Registration Form</h2>

<form id="registrationForm">

  <div>

    <label for="username">Username:</label>

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

    <span id="usernameError"></span>

  </div>

  <div>

    <label for="email">Email:</label>

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

    <span id="emailError"></span>

  </div>

  <div>

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

    <input type="password" id="password" name="password">

    <span id="passwordError"></span>

  </div>

  <div>

    <label for="mobile">Mobile Number:</label>

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

    <span id="mobileError"></span>

  </div>

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

</form>

<script>

  const form = document.getElementById('registrationForm');

  form.addEventListener('submit', function(event) {

    event.preventDefault();

    if (validateForm()) {

      alert('Form submitted successfully!');

      // You can add code here to submit the form data to your backend server

    }

  });

  function validateForm() {

    const username = document.getElementById('username').value.trim();

    const email = document.getElementById('email').value.trim();

    const password = document.getElementById('password').value.trim();

    const mobile = document.getElementById('mobile').value.trim();

    let isValid = true;

    // Username validation

    if (username === '') {

      document.getElementById('usernameError').innerText = 'Username is required';

      isValid = false;

    } else {

      document.getElementById('usernameError').innerText = '';z

    }

    // Email validation

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!emailRegex.test(email)) {

      document.getElementById('emailError').innerText = 'Invalid email format';

      isValid = false;

    } else {

      document.getElementById('emailError').innerText = '';

    }

    // Password validation

    if (password.length < 6) {

      document.getElementById('passwordError').innerText = 'Password must be at least 6 characters';

      isValid = false;

    } else {

      document.getElementById('passwordError').innerText = '';

    }

    // Mobile number validation

    const mobileRegex = /^\d{10}$/;

    if (!mobileRegex.test(mobile)) {

      document.getElementById('mobileError').innerText = 'Mobile number must be 10 digits';

      isValid = false;

    } else {

      document.getElementById('mobileError').innerText = '';

    }

    return isValid;

  }

</script>

</body>

</html>

B. Write a program to demonstrate the client and server communication using http request and response.

HTML File :

!DOCTYPE html

html lang=en

head

meta charset=

 quot;UTF-8&quot;&gt;

&lt;meta http-equiv=&quot;X-UA-Compatible&quot; content=&quot;IE=edge&quot;&gt;

&lt;meta name=&quot;viewport&quot; content=&quot;width=device-width, initial-scale=1.0&quot;&gt;

&lt;title&gt;Document&lt;/title&gt;

&lt;/head&gt;

&lt;body style=&quot;background-color:azure;&quot;&gt;

&lt;form action=&quot;http://localhost:3000/area.js&quot; method=&quot;get&quot;&gt;

&lt;h3&gt;Rectangle&lt;/h3&gt;&lt;br&gt; Enter length : &lt;input type=&quot;number&quot; name=&quot;t1&quot;&gt;

&lt;br&gt;&lt;br&gt; Enter breadth : &lt;input type=&quot;number&quot; name=&quot;t2&quot;&gt;

&lt;br&gt;&lt;br&gt; &lt;h3&gt;Circle&lt;/h3&gt;&lt;br&gt; Enter radius : &lt;input type=&quot;number&quot; name=&quot;t3&quot;&gt;

&lt;br&gt;&lt;br&gt;

&lt;h3&gt;Square &lt;/h3&gt;&lt;br&gt; Enter side : &lt;input type=&quot;number&quot; name=&quot;t4&quot;&gt;

&lt;br&gt;&lt;br&gt;&lt;button type=&quot;submit&quot;&gt;Area&lt;/button&gt;

&lt;/form&gt;

&lt;/body&gt;

&lt;/html&gt;

area.js

const http = require(&#39;http&#39;);

url = require(&#39;url&#39;);

http.createServer((req, res) =&gt; {

console.log(req.url);

var q = url.parse(req.url, true).query;

console.log(q);

console.log(q.t1);

console.log(q.t2);

console.log(q.t3);

console.log(q.t4);

var a = Number(q.t1);

var b = Number(q.t2);

var c = Number(q.t3);

var d = Number(q.t4);

var area = a \* b;

var circle = 3.14 \* c \* c;

var sqr = d \* d;

res.write(&quot;Area of the rectangle is &quot; + area + &quot;\n&quot;);

res.write(&quot;Area of the circle is &quot; + circle + &quot;\n&quot;);

res.write(&quot;Area of the square is &quot; + sqr);

res.end();

}).listen(3000);

1. A. Design a simple form to collect the personal information of a user of a banking application and process the data using GET method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Banking Application - Personal Information</title>

</head>

<body>

<h2>Personal Information Form</h2>

<form action="/process\_data" method="GET">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="address">Address:</label>

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

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const app = express();

app.get('/process\_data', (req, res) => {

    const fullName = req.query.full\_name;

    const email = req.query.email;

    const address = req.query.address;

    const phone = req.query.phone;

    // Process the data (e.g., store it in a database)

    // For demonstration purposes, we'll just log the data

    console.log('Full Name:', fullName);

    console.log('Email:', email);

    console.log('Address:', address);

    console.log('Phone Number:', phone);

    // Send a response back to the client

    res.send('Data received successfully!');

});

app.listen(3000, () => {

    console.log('Server is running on http://localhost:3000');

});

B. Collect the personal information of a user of a Student management system and store the data in MongoDB database and process it using Update and delete operations.

const mongoose = require('mongoose');

// Connect to MongoDB database

mongoose.connect('mongodb://localhost:27017/student\_management', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

// Define student schema

const studentSchema = new mongoose.Schema({

  fullName: String,

  email: String,

  address: String,

  phoneNumber: String,

});

// Create student model

const Student = mongoose.model('Student', studentSchema);

// Function to add a new student

async function addStudent(data) {

  try {

    const student = new Student(data);

    await student.save();

    console.log('Student added successfully!');

  } catch (error) {

    console.error('Error adding student:', error);

  }

}

// Function to update student information

async function updateStudent(id, newData) {

  try {

    await Student.findByIdAndUpdate(id, newData);

    console.log('Student updated successfully!');

  } catch (error) {

    console.error('Error updating student:', error);

  }

}

// Function to delete a student

async function deleteStudent(id) {

  try {

    await Student.findByIdAndDelete(id);

    console.log('Student deleted successfully!');

  } catch (error) {

    console.error('Error deleting student:', error);

  }

}

// Example usage

const newStudent = {

  fullName: 'John Doe',

  email: 'john.doe@example.com',

  address: '123 Main Street',

  phoneNumber: '555-123-4567',

};

addStudent(newStudent);

// Example update operation

updateStudent('student\_id\_here', { address: '456 Elm Street' });

// Example delete operation

deleteStudent('student\_id\_here');

1. . A. Design a simple form to collect the personal information of a user of a banking application and process the data using POST method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Banking Application - Personal Information</title>

</head>

<body>

<h2>Personal Information Form</h2>

<form action="/process\_data" method="POST">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="address">Address:</label>

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

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const bodyParser = require('body-parser');

 [

const app = express();

const PORT = process.env.PORT || 3000;

// Middleware to parse form data

app.use(bodyParser.urlencoded({ extended: true }));

// Route to handle form submission

app.post('/process\_data', (req, res) => {

  const fullName = req.body.full\_name;

  const email = req.body.email;

  const address = req.body.address;

  const phone = req.body.phone;

  // Process the data (e.g., store it in a database)

  // For demonstration purposes, we'll just log the data

  console.log('Full Name:', fullName);

  console.log('Email:', email);

  console.log('Address:', address);

  console.log('Phone Number:', phone);

  // Send a response back to the client

  res.send('Data received successfully!');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

B. For a civil construction management application, Design a simple webpages using React JS and navigate from one component to another component

npm install react react-dom react-router-dom

// App.js

import React from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

import Home from './components/Home';

import Projects from './components/Projects';

import Resources from './components/Resources';

function App() {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li>

              <Link to="/">Home</Link>

            </li>

            <li>

              <Link to="/projects">Projects</Link>

            </li>

            <li>

              <Link to="/resources">Resources</Link>

            </li>

          </ul>

        </nav>

        <Route path="/" exact component={Home} />

        <Route path="/projects" component={Projects} />

        <Route path="/resources" component={Resources} />

      </div>

    </Router>

  );

}

export default App;

// components/Home.js

import React from 'react';

function Home() {

  return (

    <div>

      <h2>Welcome to Civil Construction Management Application</h2>

      <p>This is the home page of the application.</p>

    </div>

  );

}

export default Home;

// components/Projects.js

import React from 'react';

function Projects() {

  return (

    <div>

      <h2>Projects</h2>

      <p>List of construction projects will be displayed here.</p>

    </div>

  );

}

export default Projects;

// components/Resources.js

import React from 'react';

function Resources() {

  return (

    <div>

      <h2>Resources</h2>

      <p>Useful resources for construction management will be displayed here.</p>

    </div>

  );

}

export default Resources;

1. A. For a banking management system, create a simple form and validate the following using JavaScript

password

e-mail

username validation

mobile number validation

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Banking Management System</title>

</head>

<body>

<h2>User Registration Form</h2>

<form id="registrationForm">

  <div>

    <label for="username">Username:</label>

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

    <span id="usernameError" class="error"></span>

  </div>

  <div>

    <label for="email">Email:</label>

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

    <span id="emailError" class="error"></span>

  </div>

  <div>

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

    <input type="password" id="password" name="password">

    <span id="passwordError" class="error"></span>

  </div>

  <div>

    <label for="mobile">Mobile Number:</label>

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

    <span id="mobileError" class="error"></span>

  </div>

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

</form>

<script>

  const form = document.getElementById('registrationForm');

  form.addEventListener('submit', function(event) {

    event.preventDefault();

    if (validateForm()) {

      alert('Form submitted successfully!');

      // You can add code here to submit the form data to your backend server

    }

  });

  function validateForm() {

    const username = document.getElementById('username').value.trim();

    const email = document.getElementById('email').value.trim();

    const password = document.getElementById('password').value.trim();

    const mobile = document.getElementById('mobile').value.trim();

    let isValid = true;

    // Username validation

    if (username === '') {

      document.getElementById('usernameError').innerText = 'Username is required';

      isValid = false;

    } else {

      document.getElementById('usernameError').innerText = '';

    }

    // Email validation

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!emailRegex.test(email)) {

      document.getElementById('emailError').innerText = 'Invalid email format';

      isValid = false;

    } else {

      document.getElementById('emailError').innerText = '';

    }

    // Password validation

    if (password.length < 6) {

      document.getElementById('passwordError').innerText = 'Password must be at least 6 characters';

      isValid = false;

    } else {

      document.getElementById('passwordError').innerText = '';

    }

    // Mobile number validation

    const mobileRegex = /^\d{10}$/;

    if (!mobileRegex.test(mobile)) {

      document.getElementById('mobileError').innerText = 'Mobile number must be 10 digits';

      isValid = false;

    } else {

      document.getElementById('mobileError').innerText = '';

    }

    return isValid;

  }

</script>

</body>

</html>

B. Same as 1 B

1. A. Design a simple form to collect the personal information of a student application and process the data using GET method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Application Form</title>

</head>

<body>

<h2>Student Application Form</h2>

<form action="/process\_data" method="GET">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="dob">Date of Birth:</label>

    <input type="date" id="dob" name="dob" required>

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const app = express();

const PORT = process.env.PORT || 3000;

// Route to handle form submission

app.get('/process\_data', (req, res) => {

  const fullName = req.query.full\_name;

  const email = req.query.email;

  const dob = req.query.dob;

  const phone = req.query.phone;

  // Process the data (e.g., store it in a database)

  // For demonstration purposes, we'll just log the data

  console.log('Full Name:', fullName);

  console.log('Email:', email);

  console.log('Date of Birth:', dob);

  console.log('Phone Number:', phone);

  // Send a response back to the client

  res.send('Data received successfully!');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

B. Collect the personal information of a user of a student application system and store the data in MongoDB database. Perform single insertion and bulk insertion using it.

const mongoose = require('mongoose');

// Connect to MongoDB database

mongoose.connect('mongodb://localhost:27017/student\_application', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

// Define student schema

const studentSchema = new mongoose.Schema({

  fullName: String,

  email: String,

  dob: Date,

  phone: String,

});

// Create student model

const Student = mongoose.model('Student', studentSchema);

// Function to insert a single student

async function insertSingleStudent(data) {

  try {

    const student = new Student(data);

    await student.save();

    console.log('Single student inserted successfully!');

  } catch (error) {

    console.error('Error inserting single student:', error);

  }

}

// Function to insert multiple students

async function insertBulkStudents(data) {

  try {

    await Student.insertMany(data);

    console.log('Bulk insertion of students successful!');

  } catch (error) {

    console.error('Error inserting bulk students:', error);

  }

}

// Example usage for single insertion

const singleStudentData = {

  fullName: 'John Doe',

  email: 'john.doe@example.com',

  dob: new Date('1990-01-01'),

  phone: '555-123-4567',

};

insertSingleStudent(singleStudentData);

// Example usage for bulk insertion

const bulkStudentsData = [

  {

    fullName: 'Jane Smith',

    email: 'jane.smith@example.com',

    dob: new Date('1992-03-15'),

    phone: '555-987-6543',

  },

  {

    fullName: 'David Johnson',

    email: 'david.johnson@example.com',

    dob: new Date('1995-07-20'),

    phone: '555-456-7890',

  },

];

insertBulkStudents(bulkStudentsData);

1. A. Design a simple form to collect the personal information of a student for library application and process the data using POST method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Library Application - Student Information</title>

</head>

<body>

<h2>Student Information Form</h2>

<form action="/process\_data" method="POST">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="student\_id">Student ID:</label>

    <input type="text" id="student\_id" name="student\_id" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const PORT = process.env.PORT || 3000;

// Middleware to parse form data

app.use(bodyParser.urlencoded({ extended: true }));

// Route to handle form submission

app.post('/process\_data', (req, res) => {

  const fullName = req.body.full\_name;

  const studentId = req.body.student\_id;

  const email = req.body.email;

  const phone = req.body.phone;

  // Process the data (e.g., store it in a database)

  // For demonstration purposes, we'll just log the data

  console.log('Full Name:', fullName);

  console.log('Student ID:', studentId);

  console.log('Email:', email);

  console.log('Phone Number:', phone);

  // Send a response back to the client

  res.send('Data received successfully!');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

B. For a library management application, Design a simple webpages using React JS and  navigate from one component to another component

npm install react react-dom react-router-dom

// App.js

import React from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

import Home from './components/Home';

import Books from './components/Books';

import Members from './components/Members';

function App() {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li>

              <Link to="/">Home</Link>

            </li>

            <li>

              <Link to="/books">Books</Link>

            </li>

            <li>

              <Link to="/members">Members</Link>

            </li>

          </ul>

        </nav>

        <Route path="/" exact component={Home} />

        <Route path="/books" component={Books} />

        <Route path="/members" component={Members} />

      </div>

    </Router>

  );

}

export default App;

// components/Home.js

import React from 'react';

function Home() {

  return (

    <div>

      <h2>Welcome to Library Management Application</h2>

      <p>This is the home page of the application.</p>

    </div>

  );

}

export default Home;

// components/Books.js

import React from 'react';

function Books() {

  return (

    <div>

      <h2>Books</h2>

      <p>List of books available in the library will be displayed here.</p>

    </div>

  );

}

export default Books;

// components/Members.js

import React from 'react';

function Members() {

  return (

    <div>

      <h2>Members</h2>

      <p>List of library members will be displayed here.</p>

    </div>

  );

}

export default Members;

1. SAME AS Q1
2. A. Design a simple form to collect the personal information of a customer for an online shopping application and process the data using GET method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Online Shopping - Customer Information</title>

</head>

<body>

<h2>Customer Information Form</h2>

<form action="/process\_data" method="GET">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="address">Address:</label>

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

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const app = express();

const PORT = process.env.PORT || 3000;

// Route to handle form submission

app.get('/process\_data', (req, res) => {

  const fullName = req.query.full\_name;

  const email = req.query.email;

  const address = req.query.address;

  const phone = req.query.phone;

  // Process the data (e.g., store it in a database)

  // For demonstration purposes, we'll just log the data

  console.log('Full Name:', fullName);

  console.log('Email:', email);

  console.log('Address:', address);

  console.log('Phone Number:', phone);

  // Send a response back to the client

  res.send('Data received successfully!');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

B. Collect the personal information of a user of a customer for an online shopping application and store the data in MongoDB database and process it using getone operation.

const mongoose = require('mongoose');

// Connect to MongoDB database

mongoose.connect('mongodb://localhost:27017/online\_shopping', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

// Define customer schema

const customerSchema = new mongoose.Schema({

  fullName: String,

  email: String,

  address: String,

  phone: String,

});

// Create customer model

const Customer = mongoose.model('Customer', customerSchema);

// Function to insert a customer

async function insertCustomer(data) {

  try {

    const customer = new Customer(data);

    await customer.save();

    console.log('Customer inserted successfully!');

  } catch (error) {

    console.error('Error inserting customer:', error);

  }

}

// Function to find one customer by email

async function findOneCustomer(email) {

  try {

    const customer = await Customer.findOne({ email });

    if (customer) {

      console.log('Customer found:', customer);

    } else {

      console.log('Customer not found');

     }

  } catch (error) {

    console.error('Error finding customer:', error);

  }

}

// Example usage to insert a customer

const newCustomer = {

  fullName: 'John Doe',

  email: 'john.doe@example.com',

  address: '123 Main Street',

  phone: '555-123-4567',

};

insertCustomer(newCustomer);

// Example usage to find one customer by email

const customerEmail = 'john.doe@example.com';

findOneCustomer(customerEmail);

1. A. Design a simple form to collect the personal information of a user of a banking application and process the data using POST method

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Banking Application - Personal Information</title>

</head>

<body>

<h2>Personal Information Form</h2>

<form action="/process\_data" method="POST">

  <div>

    <label for="full\_name">Full Name:</label>

    <input type="text" id="full\_name" name="full\_name" required>

  </div>

  <div>

    <label for="email">Email:</label>

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

  </div>

  <div>

    <label for="address">Address:</label>

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

  </div>

  <div>

    <label for="phone">Phone Number:</label>

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

  </div>

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

</form>

</body>

</html>

JS:

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const PORT = process.env.PORT || 3000;

// Middleware to parse form data

app.use(bodyParser.urlencoded({ extended: true }));

// Route to handle form submission

app.post('/process\_data', (req, res) => {

  const fullName = req.body.full\_name;

  const email = req.body.email;

  const address = req.body.address;

  const phone = req.body.phone;

  // Process the data (e.g., store it in a database)

  // For demonstration purposes, we'll just log the data

  console.log('Full Name:', fullName);

  console.log('Email:', email);

  console.log('Address:', address);

  console.log('Phone Number:', phone);

  // Send a response back to the client

  res.send('Data received successfully!');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

B.

npm install react react-dom react-router-dom

// App.js

import React from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

import Home from './components/Home';

import Students from './components/Students';

import AddStudent from './components/AddStudent';

function App() {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li>

              <Link to="/">Home</Link>

            </li>

            <li>

              <Link to="/students">Students</Link>

            </li>

            <li>

              <Link to="/add-student">Add Student</Link>

            </li>

          </ul>

        </nav>

        <Route path="/" exact component={Home} />

        <Route path="/students" component={Students} />

        <Route path="/add-student" component={AddStudent} />

      </div>

    </Router>

  );

}

export default App;

// components/Home.js

import React from 'react';

function Home() {

  return (

    <div>

      <h2>Welcome to Student Management Application</h2>

      <p>This is the home page of the application.</p>

    </div>

  );

}

export default Home;

// components/Students.js

import React from 'react';

function Students() {

  return (

    <div>

      <h2>Students</h2>

      <p>List of students will be displayed here.</p>

    </div>

  );

}

export default Students;

// components/AddStudent.js

import React from 'react';

function AddStudent() {

  return (

    <div>

      <h2>Add Student</h2>

      <p>Form to add a new student will be displayed here.</p>

    </div>

  );

}

export default AddStudent;

1. SAME AS Q1

1. A. Write a JavaScript code to create an array called ‘scores’ containing the scores of students in the class and perform the following operations:

Double each score in the scores array

Filter out the scores which are abive 85

Calculate the total score and average score

// Create an array called 'scores' containing the scores of students in the class

const scores = [70, 85, 90, 60, 95, 80];

// Double each score in the scores array

const doubledScores = scores.map(score => score \* 2);

// Filter out the scores which are above 85

const filteredScores = scores.filter(score => score > 85);

// Calculate the total score

const totalScore = scores.reduce((total, score) => total + score, 0);

// Calculate the average score

const averageScore = totalScore / scores.length;

console.log('Original Scores:', scores);

console.log('Doubled Scores:', doubledScores);

console.log('Filtered Scores (Above 85):', filteredScores);

console.log('Total Score:', totalScore);

console.log('Average Score:', averageScore);

B. Write a JavaScript program to display the factors of the given number

function displayFactors(number) {

    const factors = [];

    for (let i = 1; i <= number; i++) {

        if (number % i === 0) {

            factors.push(i);

        }

    }

    return factors;

}

const number = 12;

const factors = displayFactors(number);

console.log(`Factors of ${number}:`, factors);

1. A. Write a JavaScript program to display the content as “Kongu Engineering College” while you move the mouseout. When performing mousemove, “Kongu Engineering College” should be changed to “KEC”

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Mouse Event Example</title>

<style>

  #content {

    border: 1px solid black;

    padding: 20px;

    margin: 20px;

    width: 300px;

    text-align: center;

  }

</style>

</head>

<body>

<div id="content">Kongu Engineering College</div>

<script>

  const contentDiv = document.getElementById('content');

  contentDiv.addEventListener('mouseover', function() {

    contentDiv.textContent = 'KEC';

  });

  contentDiv.addEventListener('mouseout', function() {

    contentDiv.textContent = 'Kongu Engineering College';

  });

</script>

</body>

</html>

B. Write JavaScript code to perform any four math operations using callback.

function add(a, b, callback) {

    callback(a + b);

}

function subtract(a, b, callback) {

    callback(a - b);

}

function multiply(a, b, callback) {

    callback(a \* b);

}

function divide(a, b, callback) {

    if (b === 0) {

        callback('Error: Division by zero');

    } else {

        callback(a / b);

    }

}

// Example usage

add(5, 3, result => console.log('Addition:', result));

subtract(5, 3, result => console.log('Subtraction:', result));

multiply(5, 3, result => console.log('Multiplication:', result));

divide(5, 0, result => console.log('Division:', result)); // Will output an error

divide(10, 2, result => console.log('Division:', result));

1. A. Consider an array called subjects which contain five subjects. Write a JavaScript program to show the subjects in a webpage as an unordered list.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Subjects List</title>

</head>

<body>

<h2>Subjects List</h2>

<ul id="subjectsList"></ul>

<script>

  const subjects = ['Mathematics', 'Science', 'History', 'English', 'Computer Science'];

  const subjectsList = document.getElementById('subjectsList');

  subjects.forEach(subject => {

    const listItem = document.createElement('li');

    listItem.textContent = subject;

    subjectsList.appendChild(listItem);

  });

</script>

</body>

</html>

B. You're building an online shopping website. Describe how you would use mouse events to implement a feature where users can move over product images to see a larger version along with the product description. When the user move out of the image then its size should be normal and description has to be disappeared.

HTML:

<div class="product" onmouseover="showDescription(this)" onmouseout="hideDescription(this)">

    <img src="product-image.jpg" alt="Product Image">

    <p class="description">Product Description</p>

</div>

CSS:

.product {

    position: relative;

}

.description {

    display: none;

    position: absolute;

    bottom: 0;

    left: 0;

    right: 0;

    background-color: rgba(0, 0, 0, 0.5);

    color: #fff;

    padding: 10px;

    text-align: center;

}

JS:

function showDescription(element) {

    const description = element.querySelector('.description');

    description.style.display = 'block';

}

function hideDescription(element) {

    const description = element.querySelector('.description');

    description.style.display = 'none';

}

1. A. Create the NodeJS server which consists of pages like home, aboutus and contact, display the content based on the URL.

const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

    // Parse URL

    const url = req.url === '/' ? '/home' : req.url;

    const filePath = path.join(\_\_dirname, 'pages', `${url}.html`);

    // Check if file exists

    fs.access(filePath, fs.constants.F\_OK, (err) => {

        if (err) {

            // File not found, send 404 response

            res.writeHead(404, { 'Content-Type': 'text/html' });

            res.end('<h1>404 Page Not Found</h1>');

            return;

        }

        // Read the file and send it as response

        fs.readFile(filePath, (err, data) => {

            if (err) {

                // Error reading file, send 500 response

                res.writeHead(500, { 'Content-Type': 'text/html' });

                res.end('<h1>500 Internal Server Error</h1>');

                return;

            }

            // Send the file content as response

            res.writeHead(200, { 'Content-Type': 'text/html' });

            res.end(data);

        });

    });

});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

B. Write a JavaScript program for bank application in which the users transfer money from one account to another. How will use error handling mechanism to handle insufficient balance?

class BankAccount {

    constructor(accountNumber, balance) {

        this.accountNumber = accountNumber;

        this.balance = balance;

    }

    transfer(amount, receiverAccount) {

        if (this.balance >= amount) {

            this.balance -= amount;

            receiverAccount.balance += amount;

            console.log(`Transfer successful! Amount transferred: ${amount}`);

        } else {

            console.log('Insufficient balance. Transfer failed.');

        }

    }

}

// Create sender and receiver accounts

const senderAccount = new BankAccount(123456, 1000);

const receiverAccount = new BankAccount(654321, 500);

// Perform transfer with error handling

senderAccount.transfer(700, receiverAccount); // Insufficient balance

senderAccount.transfer(500, receiverAccount); // Successful transfer

console.log('Sender Account Balance:', senderAccount.balance);

console.log('Receiver Account Balance:', receiverAccount.balance);

1. A. Write the NodeJS to implement area of circle using GET and POST method

const http = require('http');

const url = require('url');

const querystring = require('querystring');

const server = http.createServer((req, res) => {

    const reqUrl = url.parse(req.url, true);

    if (reqUrl.pathname === '/area' && req.method === 'GET') {

        // GET method: Display form to input radius

        res.writeHead(200, { 'Content-Type': 'text/html' });

        res.end(`

            <form action="/area" method="post">

                <label for="radius">Enter the radius of the circle:</label>

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

                <button type="submit">Calculate Area</button>

            </form>

        `);

    } else if (reqUrl.pathname === '/area' && req.method === 'POST') {

        // POST method: Calculate area and send response

        let body = '';

        req.on('data', chunk => {

            body += chunk.toString();

        });

        req.on('end', () => {

            const postData = querystring.parse(body);

            const radius = parseFloat(postData.radius);

            const area = Math.PI \* Math.pow(radius, 2);

            res.writeHead(200, { 'Content-Type': 'text/plain' });

            res.end(`Area of the circle with radius ${radius} is ${area.toFixed(2)}`);

        });

    } else {

        // Invalid route

        res.writeHead(404, { 'Content-Type': 'text/plain' });

        res.end('404 Not Found');

    }

});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

B. Write a query for the following using MongoDB

To insert a document in a student collection which has atleast 4 key value pair

To update the mark of the student whose name is ‘Ram’

To count the number of rows in a student collection

const MongoClient = require('mongodb').MongoClient;

// MongoDB connection URI

const uri = 'mongodb://localhost:27017';

// Database name

const dbName = 'your\_database\_name';

// Collection name

const collectionName = 'student';

// Insert document with at least 4 key-value pairs

const insertDocument = async () => {

    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

    try {

        await client.connect();

        const db = client.db(dbName);

        const collection = db.collection(collectionName);

        const result = await collection.insertOne({

            name: 'John Doe',

            age: 20,

            mark: 85,

            grade: 'A'

            // Add more key-value pairs as needed

        });

        console.log('Document inserted successfully:', result);

    } catch (error) {

        console.error('Error inserting document:', error);

    } finally {

        client.close();

    }

};

// Update the mark of the student whose name is 'Ram'

const updateMark = async () => {

    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

    try {

        await client.connect();

        const db = client.db(dbName);

        const collection = db.collection(collectionName);

        const result = await collection.updateOne(

            { name: 'Ram' },

            { $set: { mark: 90 } }

        );

        console.log('Mark updated successfully:', result);

    } catch (error) {

        console.error('Error updating mark:', error);

    } finally {

        client.close();

    }

};

// Count the number of rows in the student collection

const countRows = async () => {

    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

    try {

        await client.connect();

        const db = client.db(dbName);

        const collection = db.collection(collectionName);

        const count = await collection.countDocuments();

        console.log('Number of rows:', count);

    } catch (error) {

        console.error('Error counting rows:', error);

    } finally {

        client.close();

    }

};

// Call the functions

insertDocument();

updateMark();

countRows();

1. A. Discuss the concept of creating web servers with HTTP request and response. Create the NodeJS server to respond back to the client for supermarket. When a client request for “/product” as url to display product information. When a client request for  “/about” as url to display information about supermarket. When a client request for “/contact” as url to display contact number.

const http = require('http');

const server = http.createServer((req, res) => {

    // Set response headers

    res.writeHead(200, { 'Content-Type': 'text/html' });

    // Routing based on the requested URL

    if (req.url === '/product') {

        // Display product information

        res.end('<h1>Product Information</h1><p>This is the product page.</p>');

    } else if (req.url === '/about') {

        // Display information about the supermarket

        res.end('<h1>About Us</h1><p>This is the about page.</p>');

    } else if (req.url === '/contact') {

        // Display contact information

        res.end('<h1>Contact Us</h1><p>Contact us at: 123-456-7890</p>');

    } else {

        // Handle unknown URLs with a 404 Not Found response

        res.writeHead(404, { 'Content-Type': 'text/html' });

        res.end('<h1>404 Not Found</h1><p>The requested URL was not found on this server.</p>');

    }

});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

B. Design a HTML page which contains a name, emailid and age of a person. Create a server to handle events using GET and POST method. The server needs to respond back to the client at the port number as 8000 to display user information such as name, emailed and age of a person in a web page and explain it.

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User Information</title>

</head>

<body>

  <h2>User Information</h2>

  <form action="http://localhost:8000" method="POST">

    <label for="name">Name:</label>

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

    <label for="email">Email:</label>

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

    <label for="age">Age:</label>

    <input type="number" id="age" name="age" required><br>

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

  </form>

</body>

</html>

JS:

const http = require('http');

const fs = require('fs');

const server = http.createServer((req, res) => {

    if (req.method === 'GET') {

        // Serve the HTML page

        fs.readFile('index.html', (err, data) => {

            if (err) {

                res.writeHead(500, { 'Content-Type': 'text/html' });

                res.end('<h1>500 Internal Server Error</h1>');

            } else {

                res.writeHead(200, { 'Content-Type': 'text/html' });

                res.end(data);

            }

        });

    } else if (req.method === 'POST') {

        // Handle POST request

        let body = '';

        req.on('data', chunk => {

            body += chunk.toString();

        });

        req.on('end', () => {

            const formData = new URLSearchParams(body);

            const name = formData.get('name');

            const email = formData.get('email');

            const age = formData.get('age');

            // Respond with user information

            res.writeHead(200, { 'Content-Type': 'text/html' });

            res.end(`<h2>User Information</h2>

                    <p>Name: ${name}</p>

                    <p>Email: ${email}</p>

                    <p>Age: ${age}</p>`);

        });

    } else {

        // Handle other HTTP methods

        res.writeHead(405, { 'Content-Type': 'text/html' });

        res.end('<h1>405 Method Not Allowed</h1>');

    }

});

const PORT = 8000;

server.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

1. A. Write the Javascript code to display a alert message when mouse moved over a text.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Mouseover Alert</title>

</head>

<body>

<p id="text" onmouseover="showAlert()">Move your mouse over this text.</p>

<script>

function showAlert() {

    alert('Mouse moved over the text!');

}

</script>

</body>

</html>

B. Create a component in React JS that has the following employee information as default props.

Emp-Name : String

Emp-Id : Number

Perform props validation for the above detail and display the same using ReactJS.

import React from 'react';

import PropTypes from 'prop-types';

const EmployeeInfo = ({ empName, empId }) => {

  return (

    <div>

      <h2>Employee Information</h2>

      <p>Name: {empName}</p>

      <p>ID: {empId}</p>

    </div>

  );

};

// Default props

EmployeeInfo.defaultProps = {

  empName: 'John Doe',

  empId: 12345

};

// Props validation

EmployeeInfo.propTypes = {

  empName: PropTypes.string.isRequired,

  empId: PropTypes.number.isRequired

};

export default EmployeeInfo;

1. A. Write the JavaScript to read the name of the user and display greeting message to user based on the system time when a button is clicked (Example [in case time is before 11AM]: Good morning Raja)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Greeting Message</title>

</head>

<body>

<label for="nameInput">Enter your name:</label>

<input type="text" id="nameInput">

<button onclick="displayGreeting()">Get Greeting</button>

<script>

function displayGreeting() {

    const name = document.getElementById('nameInput').value;

    const now = new Date();

    const hour = now.getHours();

    let greeting;

    if (hour < 12) {

        greeting = 'Good morning';

    } else if (hour < 18) {

        greeting = 'Good afternoon';

    } else {

        greeting = 'Good evening';

    }

    alert(`${greeting}, ${name}!`);

}

</script>

</body>

</html>

B. Create a component named Employee to display the employee details like name, age, address.

import React from 'react';

const Employee = ({ name, age, address }) => {

  return (

    <div>

      <h2>Employee Details</h2>

      <p>Name: {name}</p>

      <p>Age: {age}</p>

      <p>Address: {address}</p>

    </div>

  );

};

export default Employee;

19. A. 1. Write the mongodb query to retrieve all the student details

2. Write the mongodb query to retrieve the student details in the student collection for the department as CSE

3. Write the mongodb query to retrieve the student details in the student collection whose CGPA is greater than 9

4.Write the mongodb query to retrieve the student details in the student collection whose CGPA is less than equal to 7.5

5.Write the mongodb query to retrieve the IInd year and cse student details in the student collection.

6.Write the mongodb query to retrieve the student details from the student collection who belongs to CSE or EEE

db.student.find({})

db.student.find({ department: "CSE" })

db.student.find({ cgpa: { $gt: 9 } })

db.student.find({ cgpa: { $lte: 7.5 } })

db.student.find({ year: "II", department: "CSE" })

db.student.find({ $or: [ { department: "CSE" }, { department: "EEE" } ] })

|  |
| --- |
| B. Create a module to implement the accces the current date using NodeJS    // currentDate.js  const getCurrentDate = () => {      const currentDate = new Date();      return currentDate.toDateString();  };    module.exports = getCurrentDate;  // main.js  const getCurrentDate = require('./currentDate');    const currentDate = getCurrentDate();  console.log('Current date:', currentDate);    20.  A. Write a query for the following using MongoDB  To insert a document in a student collection which has atleast 4 key value pair  To update the mark of the student whose name is ‘Ram’  To count the number of rows in a student collection  To display the last five documents in a collection  To increment the mark of all students by 10  To change the default id of a document while inserting a document in a collection  To display name and age of a student from a collection  To update the mark of the student whose name is ‘Raju’ if a document doesn’t exist insert into a collection  To retrieve document from a collection  To delete a student collection from a database      db.student.insertOne({      name: "John Doe",      age: 20,      department: "CSE",      marks: 85  })    db.student.updateOne({ name: "Ram" }, { $set: { marks: 90 } })    db.student.countDocuments()    db.student.find().sort({ \_id: -1 }).limit(5)    db.student.updateMany({}, { $inc: { marks: 10 } })    db.student.insertOne({      \_id: "custom\_id",      name: "Jane Smith",      age: 22,      department: "EEE",      marks: 75  })    db.student.find({}, { \_id: 0, name: 1, age: 1 })    db.student.updateOne(      { name: "Raju" },      { $set: { marks: 95 } },      { upsert: true }  )    db.student.find({})    db.student.drop() |
| B. Write ExpressJS code to illustrate the usage of regular expression in providing routes.    const express = require('express');  const app = express();    // Route with regular expression to match any path starting with '/users/'  app.get(/^\/users\/(.\*)$/, (req, res) => {      res.send(`User details for: ${req.params[0]}`);  });    // Route with regular expression to match any path ending with '.html'  app.get(/.\*\.html$/, (req, res) => {      res.send(`HTML file requested: ${req.url}`);  });    // Route with regular expression to match any path containing 'admin' as a substring  app.get(/admin/, (req, res) => {      res.send(`Admin page requested: ${req.url}`);  });    const PORT = process.env.PORT || 3000;  app.listen(PORT, () => {      console.log(`Server is running on http://localhost:${PORT}`);  }); |