1.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>

   <style>.error { color: red; }</style>

</head>

<body>

   <h2>Registration Form</h2>

   <form id="studentForm">

      <label>Username:</label>

      <input type="text" id="username" required><span id="usernameError"
class="error"></span><br><br>

      <label>Email:</label>

      <input type="email" id="email" required><span id="emailError" class="error"></span><br><br>

      <label>Password:</label>

      <input type="password" id="password" required><span id="passwordError"
class="error"></span><br><br>

      <label>Mobile Number:</label>

      <input type="text" id="mobile" required><span id="mobileError"
class="error"></span><br><br>

      <button type="button" onclick="validateForm()">Submit</button>

   </form>


   <script>

      function validateForm() {

         document.querySelectorAll('.error').forEach(el => el.textContent = '');
```

```
    const username = document.getElementById('username').value;
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const mobile = document.getElementById('mobile').value;

    let isValid = true;

    if (username.length < 5 || username.length > 15) {
        document.getElementById('usernameError').textContent = 'Username must be 5-15 characters.';
        isValid = false;
    }

    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailPattern.test(email)) {
        document.getElementById('emailError').textContent = 'Invalid email address.';
        isValid = false;
    }

    if (password.length < 8) {
        document.getElementById('passwordError').textContent = 'Password must be at least 8 characters.';
        isValid = false;
    }

    const mobilePattern = /^\d{10}$/;
    if (!mobilePattern.test(mobile)) {
        document.getElementById('mobileError').textContent = 'Mobile number must be 10 digits.';
        isValid = false;
    }
```

```
        if (isValid) {
            alert('Form submitted successfully!');
            document.getElementById('studentForm').reset();
        }
    }
  </script>
</body>
</html>
```

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

```
// server.js
const http = require('http');

const server = http.createServer((req, res) => {
  if (req.method === 'GET' && req.url === '/data') {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify({ message: 'Hello from the server!' }));
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
  }
});

const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTTP Client</title>
    <script>
        function fetchData() {
            fetch('http://localhost:3000/data')
                .then(response => {
                    if (!response.ok) {
                        throw new Error('Network response was not ok ' + response.statusText);
                    }
                    return response.json();
                })
                .then(data => {
                    document.getElementById('message').textContent = data.message;
                })
                .catch(error => {
                    console.error('There was a problem with the fetch operation:', error);
                });
        }
    </script>
</head>
<body>
    <h1>HTTP Client-Server Communication</h1>
    <button onclick="fetchData()">Fetch Data from Server</button>
    <p id="message"></p>
</body>
</html>
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Document</title>
</head>
<body style="background-color:rgb(143, 216, 192)">
<h1>
<B>LOGIN PAGE</B>
</h1>
<form action="http://localhost:3000/login" method="get">
Enter your name
<input type="text" name="username" value=" " /><br>
<br>Enter your email <input type="text" name="email" value=" "
/><br><br>
<input type="submit" nmae="login" value="login" /> </form>
<br><br>

</body>
</html>
```

Get.js

```
http = require('http');
url = require('url');
querystring = require('querystring');
function onRequest(request, response) {
var path = url.parse(request.url).pathname;
console.log('Request for ' + path + 'received.');
var query = url.parse(request.url).query;
console.log(query);
var name = querystring.parse(query)["username"];
var email = querystring.parse(query)["email"];
response.write('Hello ' + name + ', email ' + email);
response.end();
}
http.createServer(onRequest).listen(3000);
console.log('Server started..');
```

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.

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');

const app = express();
const PORT = 3000;
app.use(bodyParser.json());
mongoose.connect('mongodb://localhost:27017/studentDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

const db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

const studentSchema = new mongoose.Schema({
```

```
    username: String,
    email: String,
    password: String,
    mobile: String
});

const Student = mongoose.model('Student', studentSchema);
app.post('/students', async (req, res) => {
    const student = new Student(req.body);
    try {
        await student.save();
        res.status(201).send(student);
    } catch (error) {
        res.status(400).send(error);
    }
});
app.put('/students/:id', async (req, res) => {
    try {
        const student = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true,
runValidators: true });
        if (!student) {
            return res.status(404).send();
        }
        res.send(student);
    } catch (error) {
        res.status(400).send(error);
    }
});
app.delete('/students/:id', async (req, res) => {
    try {
        const student = await Student.findByIdAndDelete(req.params.id);
        if (!student) {
            return res.status(404).send();
        }
        res.send(student);
    } catch (error) {
        res.status(500).send(error);
    }
});
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

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

Post.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
```

```
</head>
<body>
<form action="http://localhost:8080/login" method="post">
Enter your name
<input type="text" name="username" value=" " /><br><br>Enter your
email <input type="text" name="email" value=" " /><br>
<input type="submit" nmae="login" value="login" /> </form>
</body>
</html>
```

Post.js

```
var http = require('http');
var querystring = require('querystring');
var name, email;
http.createServer(function(req, res) {
var data1 = "";
req.on('data', function(chunk) {
console.log(chunk);
data1 += chunk;
console.log("Data is string" + data1);
});
req.on('end', function() {
q = querystring.parse(data1);
console.log(q);
name = q['username'];
email = q['email'];
res.write("Hello " + name + " your mail id is :" + email);
res.end();
});
}).listen(8080);
console.log("server has started..");
```

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

```
// src/components/Home.js
import React from 'react';

function Home() {
  return (
    <div>
      <h2>Home</h2>
      <p>Welcome to the Civil Construction Management Application!</p>
    </div>
  );
}

export default Home;
// src/components/Projects.js
import React from 'react';

function Projects() {
  return (
    <div>
      <h2>Projects</h2>
      <p>Here you can manage your construction projects.</p>
```

```
      </div>
    );
}

export default Projects;
// src/components/About.js
import React from 'react';

function About() {
  return (
    <div>
      <h2>About</h2>
      <p>Information about the Civil Construction Management Application.</p>
    </div>
  );
}

export default About;
// src/App.js
import React from 'react';
import { BrowserRouter as Router, Route, Link, Switch } from 'react-router-dom';
import Home from './components/Home';
import Projects from './components/Projects';
import About from './components/About';

function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/projects">Projects</Link>
            </li>
            <li>
              <Link to="/about">About</Link>
            </li>
          </ul>
        </nav>

        <Switch>
          <Route path="/" exact component={Home} />
          <Route path="/projects" component={Projects} />
          <Route path="/about" component={About} />
        </Switch>
      </div>
    </Router>
  );
}
```

```
export default App;
```
4. 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

```
const scores = [70, 85, 90, 78, 88, 92, 65, 80];
const doubledScores = scores.map(score => score * 2);
console.log('Doubled Scores:', doubledScores);
const filteredScores = doubledScores.filter(score => score > 85);
console.log('Filtered Scores (above 85):', filteredScores);
const totalScore = filteredScores.reduce((total, score) => total + score, 0);
console.log('Total Score:', totalScore);
const averageScore = filteredScores.length > 0 ? totalScore / filteredScores.length : 0;
console.log('Average Score:', averageScore);
```

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

```
function findFactors(num) {
    if (num <= 0) {
        console.log("Please enter a positive number.");
        return;
    }

    const factors = [];
    for (let i = 1; i <= num; i++) {
        if (num % i === 0) {
            factors.push(i);
        }
    }
    return factors;
}
const number = 28;
const factors = findFactors(number);
console.log(`The factors of ${number} are:`, factors);
```

5.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>

</head>

<body>

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

```
    <script>
        const contentDiv = document.getElementById('content');

        contentDiv.addEventListener('mousemove', () => {
            contentDiv.textContent = 'KEC';
        });

        contentDiv.addEventListener('mouseout', () => {
            contentDiv.textContent = 'Kongu Engineering College';
        });
    </script>
</body>
</html>
```

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

```
function add(a, b, callback) {
    const result = a + b;
    callback(result);
}
function subtract(a, b, callback) {
    const result = a - b;
    callback(result);
}

function multiply(a, b, callback) {
    const result = a * b;
    callback(result);
}
function divide(a, b, callback) {
    if (b === 0) {
        callback("Error: Division by zero");
```

```
    } else {
        const result = a / b;
        callback(result);
    }
}
function displayResult(result) {
    console.log("Result:", result);
}
add(5, 3, displayResult);
subtract(10, 4, displayResult);
multiply(7, 2, displayResult);
divide(20, 5, displayResult);
```

6. 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", "Physics", "Chemistry", "Biology", "Computer Science"];
        const subjectsList = document.getElementById("subjectsList");
        subjects.forEach(subject => {
            const listItem = document.createElement("li");
```

```
        listItem.textContent = subject;
        subjectsList.appendChild(listItem);
      });
    </script>
</body>
</html>
```

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.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Shopping Website</title>
</head>
<body>
    <div class="product-container" onmouseover="showDescription(1)" onmouseout="hideDescription(1)">
        <img class="product-image" src="product1.jpg" alt="Product 1">
        <p class="product-description" id="description1" style="display: none;">Product 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    </div>
    <div class="product-container" onmouseover="showDescription(2)" onmouseout="hideDescription(2)">
        <img class="product-image" src="product2.jpg" alt="Product 2">
        <p class="product-description" id="description2" style="display: none;">Product 2: Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    </div>
 <script>
        function showDescription(id) {
            document.getElementById(`description${id}`).style.display = 'block';
```

```
        }

        function hideDescription(id) {
            document.getElementById(`description${id}`).style.display = 'none';
        }
    </script>
</body>
</html>
```

7. 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) => {
    let filePath = '.' + req.url;
    if (filePath === './') filePath = './home.html';
    if (filePath === './aboutus') filePath = './aboutus.html';
    if (filePath === './contact') filePath = './contact.html';
    if (!filePath.endsWith('.html')) filePath = './404.html';

    fs.readFile(filePath, (err, content) => {
        if (err) {
            res.writeHead(500);
            res.end('Server Error');
            return;
        }
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(content, 'utf-8');
    });
});
```

```
server.listen(3000, () => {
  console.log('Server running on port 3000');
});
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  <h1>Home Page</h1>
  <p>Welcome to the home page!</p>
</body>
</html>
```

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?

```
function transferMoney(senderAccount, receiverAccount, amount)
  if (senderAccount.balance < amount) {
    throw new Error('Insufficient balance');
  }
  senderAccount.balance -= amount;
  receiverAccount.balance += amount;
  console.log(Transferred $${amount} from ${senderAccount.name}'s account to
${receiverAccount.name}'s account.);
}
const account1 = { name: 'John', balance: 1000 };
const account2 = { name: 'Alice', balance: 500 };
try {
  transferMoney(account1, account2, 700);
} catch (error) {
  console.error('Error:', error.message);
```

```
}
```

8. 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 calculateArea = (radius) => Math.PI * radius * radius;

const server = http.createServer((req, res) => {
  const parsedUrl = url.parse(req.url, true);

  if (req.method === 'GET' && parsedUrl.pathname === '/area') {
    const radius = parseFloat(parsedUrl.query.radius);
    if (radius && radius > 0) {
      const area = calculateArea(radius);
      res.writeHead(200, { 'Content-Type': 'application/json' });
      res.end(JSON.stringify({ radius, area }));
    } else {
      res.writeHead(400, { 'Content-Type': 'application/json' });
      res.end(JSON.stringify({ error: 'Invalid radius' }));
    }
  } else if (req.method === 'POST' && parsedUrl.pathname === '/area') {
    let body = '';
    req.on('data', chunk => body += chunk.toString());
    req.on('end', () => {
      const parsedBody = querystring.parse(body);
      const radius = parseFloat(parsedBody.radius);
      if (radius && radius > 0) {
        const area = calculateArea(radius);
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ radius, area }));
```

```
        } else {
            res.writeHead(400, { 'Content-Type': 'application/json' });
            res.end(JSON.stringify({ error: 'Invalid radius' }));
        }
    });
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
  }
});


server.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

Write a query for the following using MongoDB
1. To insert a document in a student collection which has atleast 4 key value pair
2. To update the mark of the student whose name is ‘Ram’
3. To count the number of rows in a student collection

```
db.student.insertOne({
  name: "Alice",
  age: 20,
  major: "Computer Science",
  marks: 85
});
db.student.updateOne(
  { name: "Ram" },
  { $set: { marks: 90 } }
);
db.student.countDocuments();
```

9. 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) => {
   const { url } = req;

   if (url === '/product') {
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end('<h1>Product Information</h1><p>Here are the details of our products.</p>');
   } else if (url === '/about') {
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end('<h1>About Us</h1><p>We are a leading supermarket chain.</p>');
   } else if (url === '/contact') {
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end('<h1>Contact Us</h1><p>Contact us at: (123) 456-7890</p>');
   } else {
      res.writeHead(404, { 'Content-Type': 'text/html' });
      res.end('<h1>404 Not Found</h1><p>The page you are looking for does not exist.</p>');
   }
});

const PORT = 3000;
server.listen(PORT, () => {
   console.log(`Server running at http://localhost:${PORT}/`);
});
```

10. 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">
   <title>Mouse Over Alert</title>
</head>
```

```
<body>
  <p id="hoverText">Hover over this text to see an alert message.</p>

  <script>
    document.getElementById('hoverText').addEventListener('mousemove', function() {
      alert('Mouse moved over the text!');
    });
  </script>
</body>
</html>
```

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.

Employeeinfo.js

```
import React from 'react';
import PropTypes from 'prop-types';

class EmployeeInfo extends React.Component {
  render() {
    const { empName, empId } = this.props;
    return (
      <div>
        <h2>Employee Information</h2>
        <p>Name: {empName}</p>
        <p>ID: {empId}</p>
      </div>
    );
  }
}

EmployeeInfo.defaultProps = {
```

```
  empName: "John Doe",
  empId: 12345
};

EmployeeInfo.propTypes = {
  empName: PropTypes.string.isRequired,
  empId: PropTypes.number.isRequired
};

export default EmployeeInfo;
```

APP.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import EmployeeInfo from './EmployeeInfo';

ReactDOM.render(
  <EmployeeInfo empName="Alice Smith" empId={98765} />,
  document.getElementById('root')
);
```

11. 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">
  <title>Greeting Message</title>
</head>
<body>
  <label for="userName">Enter your name: </label>
  <input type="text" id="userName">
  <button onclick="displayGreeting()">Display Greeting</button>
```

```
    <script>
        function displayGreeting() {
            const userName = document.getElementById('userName').value;
            const currentTime = new Date();
            const currentHour = currentTime.getHours();

            let greeting;

            if (currentHour < 11) {
                greeting = 'Good morning';
            } else if (currentHour < 17) {
                greeting = 'Good afternoon';
            } else {
                greeting = 'Good evening';
            }

            alert(`${greeting}, ${userName}`);
        }
    </script>
</body>
</html>
```

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

```
// Employee.js
import React from 'react';

class Employee extends React.Component {
    render() {
        const { name, age, address } = this.props;
        return (
```

```
        <div>
          <h2>Employee Details</h2>
          <p>Name: {name}</p>
          <p>Age: {age}</p>
          <p>Address: {address}</p>
        </div>
      );
  }
}

export default Employee;
// App.js
import React from 'react';
import ReactDOM from 'react-dom';
import Employee from './Employee';

const employeeData = {
  name: "John Doe",
  age: 30,
  address: "123 Main St, City, Country"
};

ReactDOM.render(
  <Employee {...employeeData} />,
  document.getElementById('root')
);
```

12. 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: 2, department: "CSE" });

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

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;

// app.js (example usage)

const getCurrentDate = require('./currentDate');


const currentDate = getCurrentDate();

console.log("Current Date:", currentDate);
```

13. Write a query for the following using MongoDB
1. To insert a document in a student collection which has atleast 4 key value pair
2. To update the mark of the student whose name is ‘Ram’
3. To count the number of rows in a student collection
4. To display the last five documents in a collection
5. To increment the mark of all students by 10
6. To change the default id of a document while inserting a document in a collection
7. To display name and age of a student from a collection
8. To update the mark of the student whose name is ‘Raju’ if a document doesn’t exist insert into a collection
9. To retrieve document from a collection
10. To delete a student collection from a database

```
db.student.insertOne({

   name: "Alice",
```

```
  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: ObjectId("newObjectIdHere"),
  name: "John",
  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({ name: "Alice" });
db.student.drop();
```

Write ExpressJS code to illustrate the usage of regular expression in providing routes.

```
const express = require('express');
const app = express();

app.get(/\/user/, (req, res) => {
```

```
  res.send('User route matched!');
});

app.get(/\/user\/\d+/, (req, res) => {
  res.send('Numeric ID route matched!');
});

app.get(/\/user\/[a-zA-Z0-9]+/, (req, res) => {
  res.send('Alphanumeric ID route matched!');
});

app.get(/^\/product\/.+$/, (req, res) => {
  res.send('Product route matched!');
});

app.get(/\binfo\b/, (req, res) => {
  res.send('Info route matched!');
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```