



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE ENGINEERING
AND INFORMATION SYSTEMS**

WINTER SEMESTER 2024-2025

PMCA601P – FULL STACK WEB DEVELOPMENT LAB

NODEJS REACTJS - EXERCISE

SUBMITTED ON: 14 – MAR - 2025

SUBMITTED BY-

AKASH KUMAR BANIK

PROGRAM: MCA

REGISTER No.: 24MCA0242

1. Assume that the server-side Node.js script maintains the details of various employees in an array. Employee details include EMPLOYEE NAME, DEPARTMENT, and SALARY. Design a form to select various employees and facilitate total salary calculation at the server end upon submitting the form.

CODE:

Index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Employee Salary Calculator</title>

</head>

<body>

  <h1>Select Employees to Calculate Total Salary</h1>

  <form id="employee-form" action="http://localhost:3000/calculate" method="POST">

    <div>

      <input type="checkbox" id="emp1" name="employees" value="5000">

      <label for="emp1">John Doe (Marketing) - $5000</label>

    </div>

    <div>

      <input type="checkbox" id="emp2" name="employees" value="4500">

      <label for="emp2">Jane Smith (HR) - $4500</label>

    </div>

    <div>

      <input type="checkbox" id="emp3" name="employees" value="6000">

      <label for="emp3">Alice Brown (Engineering) - $6000</label>

    </div>

    <div>

      <input type="checkbox" id="emp4" name="employees" value="4000">

      <label for="emp4">Bob White (Sales) - $4000</label>

    </div> <br>
```

```
        <button type="submit">Calculate Total Salary</button>
    </form>
</body>
</html>
```

Server.js:

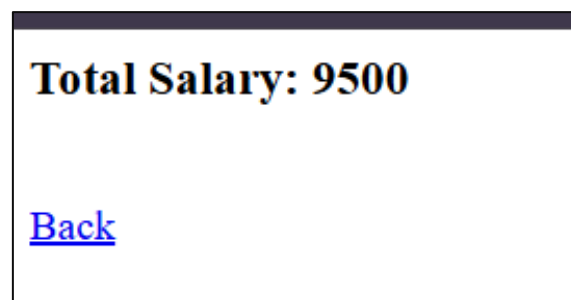
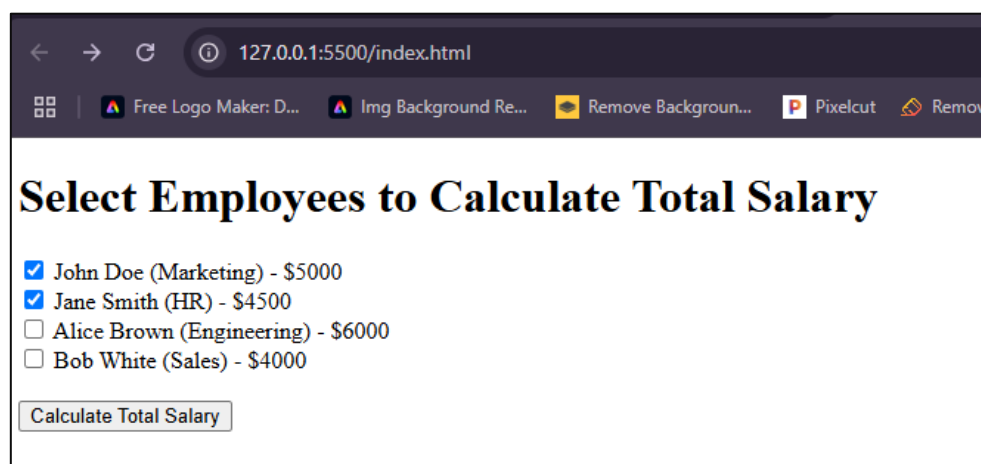
```
const express = require("express");
const bodyParser = require("body-parser");

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

let employees = [
    { id: 1, name: "Alice", department: "HR", salary: 50000 },
    { id: 2, name: "Bob", department: "IT", salary: 70000 },
    { id: 3, name: "Charlie", department: "Finance", salary: 60000 }
];

app.get("/", (req, res) => {
    let formHtml = `
        <h2>Select Employees</h2>
        <form action="/calculate" method="POST">
            ${employees.map(emp =>
                `<label>
                    <input type="checkbox" name="employees" value="${emp.salary}">
                    ${emp.name} (${emp.department})
                </label><br>`).join("")}
            <button type="submit">Calculate Salary</button>
        </form>
    `;
    res.send(formHtml);
});
```

```
app.post("/calculate", (req, res) => {  
  let totalSalary = (req.body.employees || [])  
    .map(Number) // Convert selected salaries to numbers  
    .reduce((sum, salary) => sum + salary, 0);  
  res.send(`<h3>Total Salary: ${totalSalary}</h3><br><a href="/">Back</a>`);  
});  
app.listen(3000, () => console.log("Server running on port 3000"));
```

OUTPUT:

2. Design a single Node.js web application with different URLs to manage the session process as follows:

- A form to collect the session key and value.**
- A button to either create a new session variable if it does not exist or update its value if it already exists. Display the corresponding details.**
- A button to list all active session variables, displaying each session key and its value when clicked.**

CODE:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Session Manager</title>
</head>
<body>
  <h1>Manage Sessions</h1>
  <form action="http://localhost:3000/set-session" method="POST">
    <label for="key">Session Key:</label>
    <input type="text" id="key" name="key" required>
    <label for="value">Session Value:</label>
    <input type="text" id="value" name="value" required>
    <button type="submit">Set/Update Session</button>
  </form> <br>
  <form action="/get-sessions" method="GET">
    <button type="submit">List All Sessions</button>
  </form>
</body>
</html>
```

Server.js:

```
const express = require("express");
const session = require("express-session");
const bodyParser = require("body-parser");
const app = express();
```

Middleware to parse form data

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

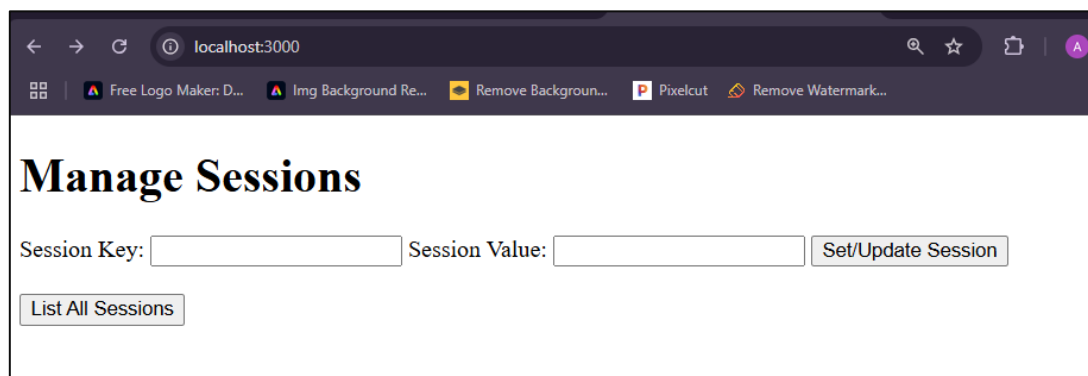
```
secret: "mySecretKey",
resave: false,
saveUninitialized: true
});

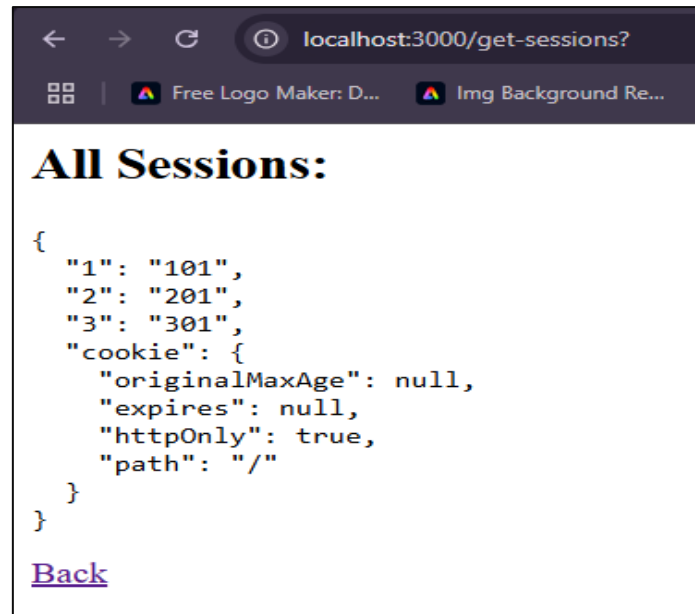
app.use(express.static(__dirname));
app.post("/set-session", (req, res) => {
  const { key, value } = req.body;

  if (key && value) {
    req.session[key] = value; // Store in session
    res.send(`<h2>Session Updated: ${key} = ${value}</h2><a href="/">Back</a>`);
  } else {
    res.send(`<h2>Invalid Input. Try again.</h2><a href="/">Back</a>`);
  }
});

app.get("/get-sessions", (req, res) => {
  res.send(`<h2>All Sessions:</h2><pre>${JSON.stringify(req.session, null, 2)}</pre><a href="/">Back</a>`);
});

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

OUTPUT:



3. Develop a Node JS application to authenticate the credentials data supplied through HTML form using the JSON array maintained in the script. Ensure JSON array consists of keys named UserID, Password, Username and Failure_Attempt. During each unsuccessful attempt increment Failure_Attempt by 1 and display the login page. In case Failure_Attempt reaches 3 , display warning message alone.(Use NodeJS session)

CODE:

Index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Login</title>

</head>

<body>

  <h1>Login</h1>

  <form action="http://localhost:3000/login" method="POST">

    <label for="userid">User ID:</label>

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

    <br>

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

```
<input type="password" id="password" name="password" required>
<br>
<button type="submit">Login</button>
</form>
<br>
<a href="http://localhost:3000/reset">Reset Attempts</a>
</body>
</html>
```

Server.js:

```
const express = require("express");
const session = require("express-session");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
  secret: "secureSessionKey",
  resave: false,
  saveUninitialized: true
}));
const users = [
  { UserID: "user1", Password: "pass123", Username: "John Doe", Failure_Attempt: 0 },
  { UserID: "user2", Password: "secret456", Username: "Alice Smith", Failure_Attempt: 0 },
  { UserID: "user3", Password: "welcome789", Username: "Bob Johnson", Failure_Attempt: 0 }
];

app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});
```



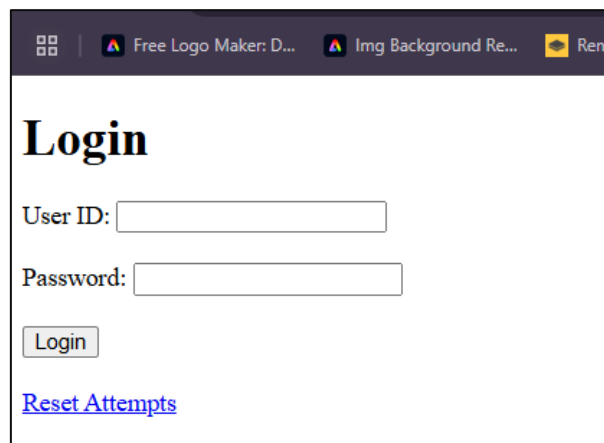
```
app.post("/login", (req, res) => {
  const { userid, password } = req.body;
  let user = users.find(u => u.UserID === userid);

  if (!user) {
    return res.send(`<h2>User not found.</h2><a href="/">Back</a>`);
  }
  if (user.Failure_Attempt >= 3) {
    return res.send(`<h2>Account Locked! Too many failed attempts.</h2>`);
  }

  if (user.Password === password) {
    req.session.username = user.Username; // Store session data
    user.Failure_Attempt = 0; // Reset failed attempts
    res.send(`<h2>Welcome, ${user.Username}!</h2><a href="/">Logout</a>`);
  } else {
    user.Failure_Attempt++; // Increment failure attempts
    res.send(`<h2>Incorrect password. Attempt ${user.Failure_Attempt}/3.</h2><a href="/">Try Again</a>`);
  }
});

app.get("/reset", (req, res) => {
  users.forEach(user => user.Failure_Attempt = 0);
  res.send(`<h2>Failure attempts reset!</h2><a href="/">Back</a>`);
});

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

OUTPUT:

Login

User ID:

Password:

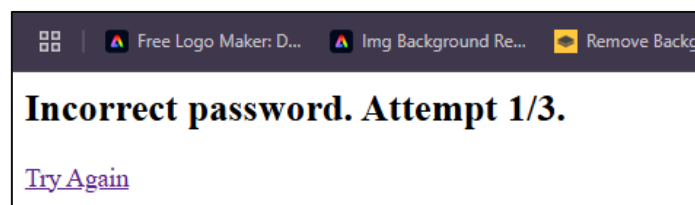
Login

[Reset Attempts](#)



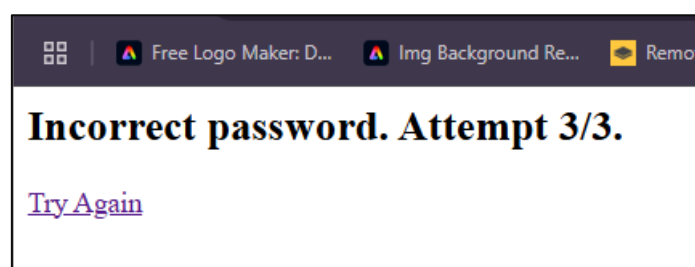
Welcome, John Doe!

[Logout](#)



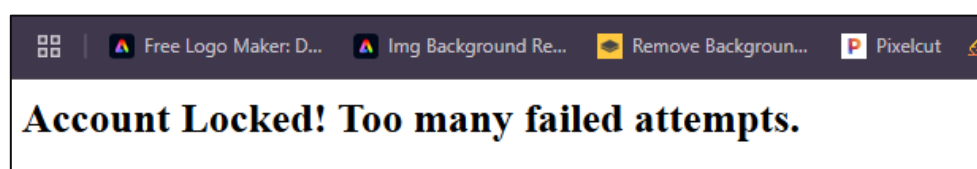
Incorrect password. Attempt 1/3.

[Try Again](#)



Incorrect password. Attempt 3/3.

[Try Again](#)



Account Locked! Too many failed attempts.

4. Write a Node.js application that allows users to select a file through a user interface and click the 'UPLOAD' button. Validate that the selected file type is 'pdf' before uploading it to the server. Finally, display a message indicating whether the file is a valid PDF and if it was uploaded successfully.

CODE:

Index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>File Upload</title>

</head>

<body>

  <h2>Upload a PDF File</h2>

  <form action="http://localhost:3000/upload" method="POST" enctype="multipart/form-
data">

    <input type="file" name="file" accept=".pdf" required> <br>

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

  </form>

</body>

</html>
```

Server.js:

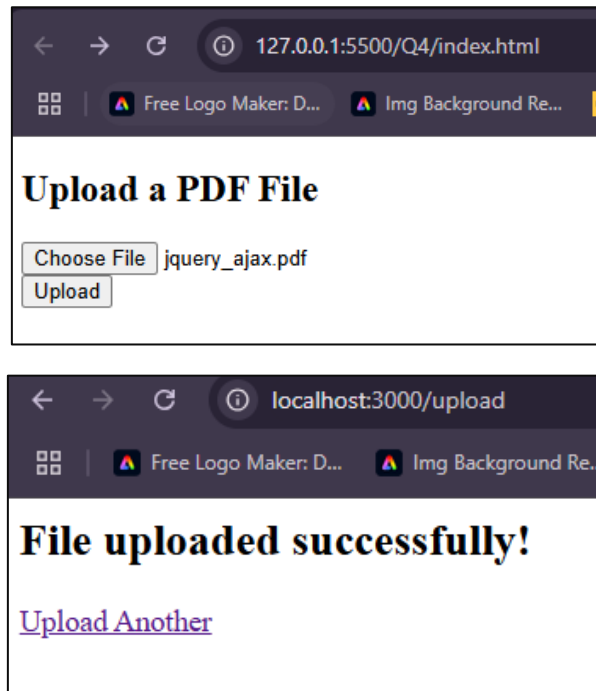
```
const express = require("express");
const multer = require("multer");
const path = require("path");

const app = express();
app.use(express.static(__dirname));

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, "uploads/"); // Files will be saved in 'uploads' folder
  },
  filename: (req, file, cb) => {
    cb(null, file.fieldname + "-" + Date.now() + path.extname(file.originalname));
  }
});
```

```
    }  
  });  
  
const fileFilter = (req, file, cb) => {  
  if (file.mimetype === "application/pdf") {  
    cb(null, true);  
  } else {  
    cb(new Error("Invalid file type! Only PDFs are allowed."), false);  
  }  
};  
  
const upload = multer({  
  storage: storage,  
  limits: { fileSize: 2 * 1024 * 1024 }, // 2MB max  
  fileFilter: fileFilter  
});  
  
app.post("/upload", (req, res) => {  
  upload.single("file")(req, res, (err) => {  
    if (err) {  
      return res.send(`<h2>${err.message}</h2><a href="/">Go Back</a>`);  
    }  
    res.send(`<h2>File uploaded successfully!</h2><a href="/">Upload Another</a>`);  
  });  
});  
  
const fs = require("fs");  
if (!fs.existsSync("uploads")) {  
  fs.mkdirSync("uploads");  
}  
app.listen(3000, () => {
```

```
console.log("Server running at http://localhost:3000");  
});
```

OUTPUT:

5. Design the React based Controlled form components for capturing the details of the vehicle. Properties of Vehicle include Model, Color, Date of Registration(use Date Control) and Vehicle Type(2/3/4 Wheeler, use Drop Down).

On submission of the form display the details captured as JSON format in an alert box. Ensure the form is displayed with default values.

CODE:**App.js file:**

```
import React, { useState } from "react";
```

```
const VehicleForm = () => {  
  // Define state for form inputs  
  const [vehicle, setVehicle] = useState({  
    model: "Honda City",  
    color: "Red",  
    registrationDate: "2023-01-01",  
    type: "4 Wheeler"
```

```
});
```

```
const handleChange = (e) => {  
  const { name, value } = e.target;  
  setVehicle((prevVehicle) => ({  
    ...prevVehicle,  
    [name]: value  
  }));  
};
```

```
const handleSubmit = (e) => {  
  e.preventDefault();  
  alert(JSON.stringify(vehicle, null, 2)); // Show form data as JSON  
};
```

```
return (  
  <div>  
    <h2>Vehicle Registration Form</h2>  
    <form onSubmit={handleSubmit}>  
      {/* Vehicle Model */}  
      <label>Model:</label>  
      <input type="text" name="model" value={vehicle.model} onChange={handleChange}  
required /> <br/><br/>  
  
      {/* Vehicle Color */}  
      <label>Color:</label>  
      <input type="text" name="color" value={vehicle.color} onChange={handleChange}  
required /> <br/><br/>  
  
      {/* Date of Registration */}  
      <label>Date of Registration:</label>
```

```
<input type="date" name="registrationDate" value={vehicle.registrationDate}
onChange={handleChange} required /> <br/><br/>
```

```
{/* Vehicle Type Dropdown */}
```

```
<label>Vehicle Type:</label>
```

```
<select name="type" value={vehicle.type} onChange={handleChange} required>
```

```
<option value="2 Wheeler">2 Wheeler</option>
```

```
<option value="3 Wheeler">3 Wheeler</option>
```

```
<option value="4 Wheeler">4 Wheeler</option>
```

```
</select>
```

```
<br /><br />
```

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

```
</form>
```

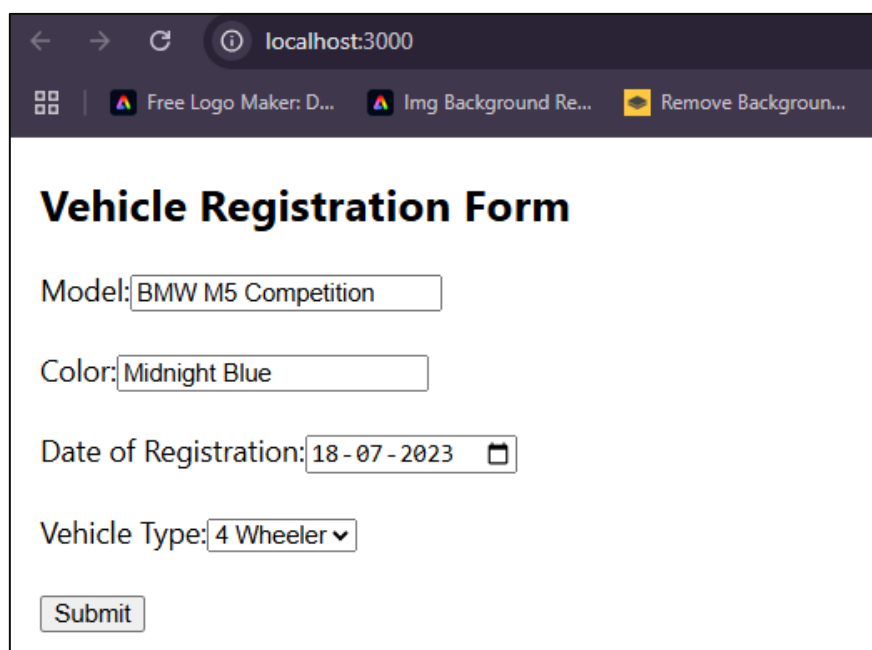
```
</div>
```

```
);
```

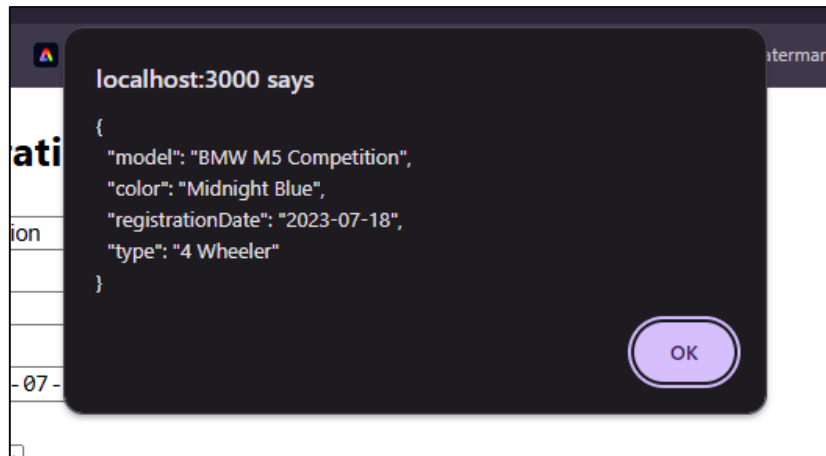
```
};
```

```
export default VehicleForm;
```

OUTPUT:



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The browser's tab bar includes 'Free Logo Maker: D...', 'Img Background Re...', and 'Remove Backgroun...'. The main content area features a form titled 'Vehicle Registration Form'. The form has four input fields: 'Model:' with the value 'BMW M5 Competition', 'Color:' with the value 'Midnight Blue', 'Date of Registration:' with the value '18-07-2023' and a calendar icon, and 'Vehicle Type:' with a dropdown menu showing '4 Wheeler'. A 'Submit' button is located at the bottom of the form.



6. Create a React Class Component named 'Customer' with properties: name, age, gender, designation, and years of experience, all assigned default values. Write a React.js script to receive data from 'index.html', validate the age, and display the message in the following format:

"Data of Mr./Ms. <Name> is valid" if the age is above 25.

"Data of Mr./Ms. <Name> is invalid" if the age is below 25.

CODE:

Customer.js:

```
import React from "react";
```

```
class Customer extends React.Component {
```

```
  render() {
```

```
    const { name, age, gender, designation, yearsOfExperience } = this.props;
```

```
    const message = age > 25
```

```
      ? `Data of Mr./Ms. ${name} is valid`
```

```
      : `Data of Mr./Ms. ${name} is invalid`;
```

```
    return (
```

```
      <div>
```

```
        <h2>Customer Details</h2>
```

```
        <p><strong>Name:</strong> {name}</p>
```

```
        <p><strong>Age:</strong> {age}</p>
```



```
<p><strong>Gender:</strong> {gender}</p>
<p><strong>Designation:</strong> {designation}</p>
<p><strong>Years of Experience:</strong> {yearsOfExperience}</p>
<h3>{message}</h3>
</div>

);
}
}

// Default Props
Customer.defaultProps = {
  name: "John Doe",
  age: 30,
  gender: "Male",
  designation: "Software Engineer",
  yearsOfExperience: 5,
};
export default Customer;

App.js:
import React from "react";
import Customer from "../Customer";

function App() {
  return (
    <div>
      <h1>Customer Information</h1>
      {/* Passing Props from index.html or using defaultProps */}
      <Customer name="Alice Smith" age={22} gender="Female" designation="Manager"
yearsOfExperience={3} />
    </div>
  );
}
```

```
);  
}
```

```
export default App;
```

Index.js:

```
import React from "react";  
import ReactDOM from "react-dom/client";  
import App from "./App";  
  
const root = ReactDOM.createRoot(document.getElementById("root"));  
root.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>  
);
```

OUTPUT: