

Ex.No: 6	Practice Exercise : Servlets and Node.js
Date: 23/1/25	

Aim:

To practice the given problems using Node.js and servlets.

1) A web application that takes a name as input and on submit it shows a hello <name> page where name is taken from the request. It shows the start time at the right top corner of the page and provides a logout button. On clicking this button, it should show a logout page with Thank You <name> message with the duration of usage (hint: Use session to store name and time).

Step 1: Installation of Node.js and verification by running the following commands

```
PS D:\Projects\travelers-radio\node_projects> npm -v
10.8.2
PS D:\Projects\travelers-radio\node_projects> node -v
v20.18.0
```

Step 2: Installation of Express.js (via npm) which is a web framework for Node.js that simplifies routing and handling HTTP requests.

```
PS D:\Projects\travelers-radio\node_projects> npm install express
added 69 packages, and audited 70 packages in 4s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step 3: Installation of other dependencies

```
PS D:\Projects\travelers-radio\node_projects> npm install express-session
added 6 packages, and audited 76 packages in 2s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step 4: Initializing the project

```
PS D:\Projects\travelers-radio\node_projects> npm init -y
Wrote to D:\Projects\travelers-radio\node_projects\package.json:
```

```
{
  "name": "node_projects",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

```
node_projects > JS app.js > app.get('/') callback
```

```
1  const express = require('express');
2  const session = require('express-session');
3  const app = express();
4  const port = 3000;
5
6  app.use(express.urlencoded({ extended: true }));
7  app.use(session({
8    secret: 'of88w3yrf7wg3r9f',
9    resave: false,
10   saveUninitialized: false
11 }));
12
13 app.get('/', (req, res) => {
14   res.send(`
15     <html>
16     <body>
17       <form action="/submit" method="POST">
18         <label for="name">Name: </label>
19         <input type="text" id="name" name="name" required>
20         <input type="submit" value="Submit">
21       </form>
22     </body>
23   </html>`;
24 });
25
26 app.post('/submit', (req, res) => {
27   const name = req.body.name;
28   req.session.name = name;
29   req.session.time = Date.now();
30   res.send(`
31     <html>
32     <body>
33       <h1>Hello ${name}</h1>
34       <p>Start Time: ${new Date(req.session.time).toLocaleString()}</p>
35       <a href="/logout">Logout</a>
36     </body>
37   </html>`);
38 });
39
40
```

```

41  app.get('/logout',(req,res)=>{
42      const endtime = Date.now();
43      const duration = ((endtime - req.session.time)/1000).toFixed(2);
44      res.send(`
45          <html>
46          <body>
47              <h1>Thank You ${req.session.name}</h1>
48              <p>Time Taken${duration}</p>
49          </body>
50          </html>`);
51      req.session.destroy();
52  });
53
54
55  app.listen(port, () => {
56      console.log(`Server running at http://localhost:${port}`);
57  });

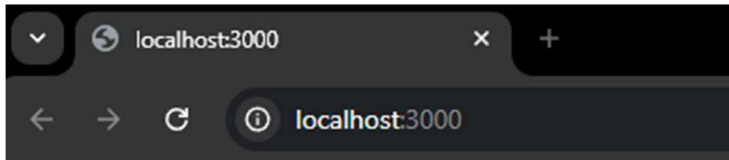
```

Output:

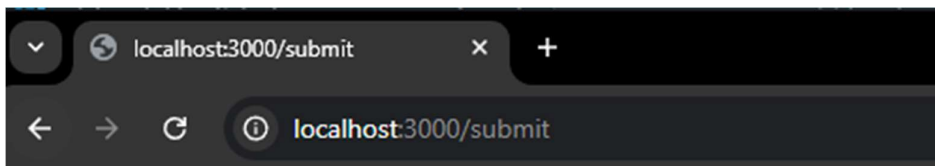
```

PS D:\Projects\travelers-radio\node_projects> node app.js
Server running at http://localhost:3000

```



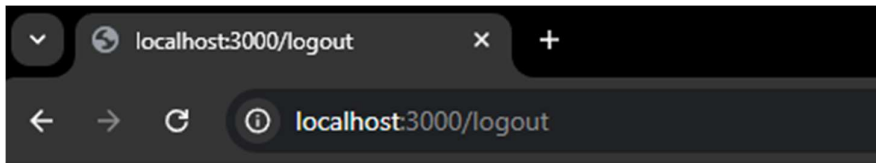
Name:



Hello Sreeraghavan

Start Time: 29/1/2025, 11:01:48 pm

[Logout](#)



Thank You Sreeraghavan

Time Taken1.97

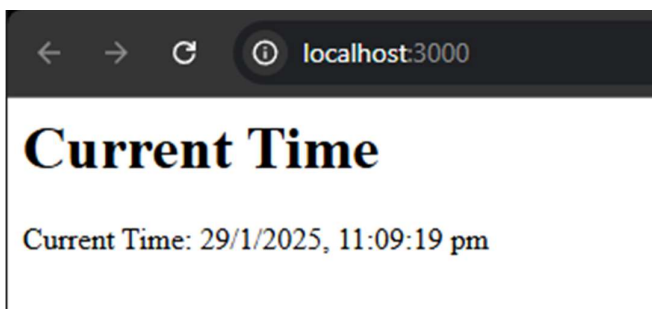
2)Write a Servlet application to print the current date and time.

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

app.get('/', (req, res) => {
  res.send(`
    <html>
    <body>
    <h1>Current Time</h1>
    <p>Current Time: ${new Date().toLocaleString()}</p>
    </body>
  </html>
`);
});

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

Output:



3) A web application that takes name and age from an HTML page. If the age is less than 18, it should send a page with “Hello <name>, you are not authorized to visit the site” message, where <name> should be replaced with the entered name. Otherwise it should send “Welcome <name> to this site” message.

agelogin.html

```
JS app2.js U X  <> agelogin.html U X
node_projects > <> agelogin.html > html > body > form > input#age
1  <html>
2    <body>
3      <form action="/submit" method="POST">
4        <label for="name">Name: </label>
5        <input type="text" id="name" name="name" required>
6        <label for="age">Age: </label>
7        <input type="number" id="age" name="age" required>
8        <input type="submit" value="Submit">
9      </form>
10   </body>
11 </html>
```

app2.js

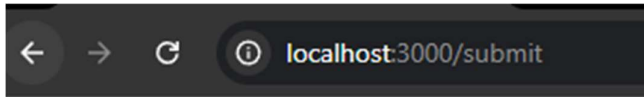
```
JS app2.js U X  <> agelogin.html U
node_projects > JS app2.js > ...
1  const express = require('express');
2  const parser = require('body-parser');
3  const app = express();
4  const port = 3000;
5
6  app.use(parser.urlencoded({ extended: true }));
7
8
9  app.get('/', (req, res) => {
10    res.sendFile(__dirname + '/agelogin.html');
11  });
12
13  app.post('/submit', (req, res) => {
14    const { name, age } = req.body;
15    if (age < 18) {
16      res.send(`
17        <html>
18        <body>
19          Hello ${name}, you are not authorized to use this site
20        </body>
21      </html>`);
22    } else {
23      res.send(`
24        <html>
25        <body>
26          Welcome ${name} to this site
27        </body>
28      </html>`);
29    }
30  });
31
32
33  app.listen(port, () => {
34    console.log(`Server running at http://localhost:${port}`);
35  });
```

Output:



localhost:3000

Name: Age:



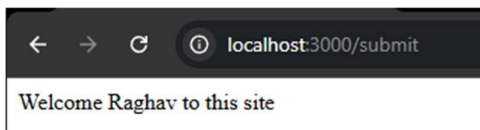
localhost:3000/submit

Hello Raghav, you are not authorized to use this site



localhost:3000

Name: Age:



localhost:3000/submit

Welcome Raghav to this site

4) A web application that lists all cookies stored in the browser on clicking “List Cookies” button. Add cookies if necessary.

app3.js

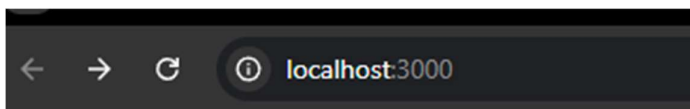
```
node_projects > JS app3.js > ...
1  const express = require('express');
2  const cookieParser = require('cookie-parser');
3  const path = require('path');
4
5  const app = express();
6  const port = 3000;
7
8  app.use(cookieParser());
9
10 ∨ app.get('/', (req, res) => {
11   |   res.sendFile(path.join(__dirname, 'cookiepage.html'));
12   | });
13
14 ∨ app.get('/set-cookies', (req, res) => {
15   |   res.cookie('username', 'JohnDoe');
16   |   res.cookie('theme', 'dark');
17   |   res.send('Cookies set! <a href="/">Go back</a>');
18   | });
19
20
21 ∨ app.listen(port, () => {
22   |   console.log('Server running at http://localhost:${port}');
23   | });
24
```

index1.html

cookiepage.html

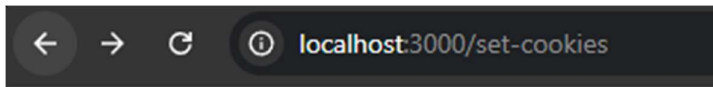
```
node_projects > <> cookiepage.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial
6    <title>Cookie List</title>
7  </head>
8  <body>
9    <h1>Cookies in this page:</h1>
10   <ul id="cookie-list"></ul>
11
12   <script>
13     function listCookies() {
14       const cookies = document.cookie.split('; ');
15       const list = document.getElementById('cookie-list');
16       list.innerHTML = '';
17
18       if (cookies[0] === "") {
19         list.innerHTML = '<li>No cookies found</li>';
20         return;
21       }
22
23       cookies.forEach(cookie => {
24         const [name, value] = cookie.split('=');
25         const li = document.createElement('li');
26         li.textContent = `${name}: ${value}`;
27         list.appendChild(li);
28       });
29     }
30
31     listCookies();
32   </script>
33 </body>
34 </html>
```

Output:

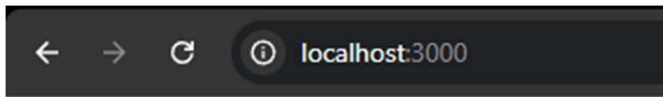


Cookies in this page:

- No cookies found



Cookies set! [Go back](#)



Cookies in this page:

- username: JohnDoe
- theme: dark

5. A user validation web application, where the user submits the login name & password to the server. The name and password are checked against the data already available in Database and if the data matches, a successful login page is returned. Otherwise failure message is shown to the user.

server.js


```

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

const app = express();
const port = 3000;

mongoose.connect('mongodb://127.0.0.1:27017/userDB', { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('MongoDB Connection Error:', err));

const userSchema = new mongoose.Schema({
  username: String,
  password: String
});
const User = mongoose.model('User', userSchema);

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

app.get('/', (req, res) => {
  res.send(`
    <html>
    <body>
      <h1>Login</h1>
      <form action="/login" method="POST">
        <label>Username:</label>
        <input type="text" name="username" required><br><br>
        <label>Password:</label>
        <input type="password" name="password" required><br><br>
        <button type="submit">Login</button>
      </form>
    </body>
    </html>
  `);
});

```

```

app.post('/login', async (req, res) => {
  const { username, password } = req.body;

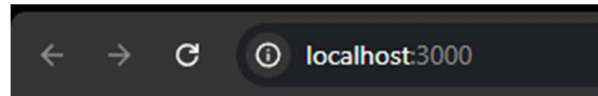
  const user = await User.findOne({ username, password });
  if (user) {
    res.send(`
      <html>
      <body>
        <h1> Login Successful</h1>
        <p>Welcome, ${username}!</p>
      </body>
      </html>
    `);
  } else {
    res.send(`
      <html>
      <body>
        <h1> Invalid Credentials</h1>
        <p>Try again.</p>
        <a href="/">Go back</a>
      </body>
      </html>
    `);
  }
});

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

```

Output:

```
PS D:\Projects\travelers-radio\node_projects> node .\server.js
(node:12200) [MONGODB DRIVER] Warning: useNewUrlParser is a depre
(Use `node --trace-warnings ...` to show where the warning was cr
(node:12200) [MONGODB DRIVER] Warning: useUnifiedTopology is a de
Server running at http://localhost:3000
Connected to MongoDB
```

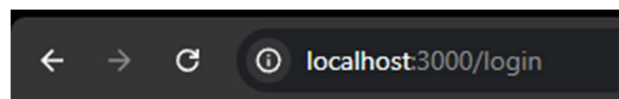


Login

Username:

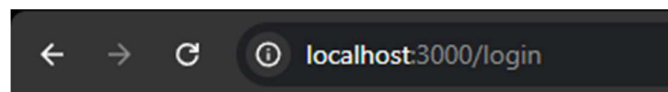
Password:

Login



Login Successful

Welcome, testuser!



Invalid Credentials

Try again.

[Go back](#)

The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS (1)' panel lists the connection '127.0.0.1:27017' with a tree view showing databases 'admin', 'config', 'local', and 'userDB'. Under 'userDB', there are collections 'idk' and 'users'. The 'users' collection is selected. The main panel shows the 'Documents' tab for the 'users' collection. It displays a single document with the following fields: `_id` (ObjectId), `username` ('testuser'), and `password` ('password123'). Above the document, there are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'.

Result:

Servers with express were created and successfully tested