1.Demonstrate to display Message using Node js

```
console.log("Hellow world");
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\1no.js"
Hellow world
PS C:\Users\Admin\Desktop\node 19 prac>
```

2.Demonstrate a Node js. Programme to create the TodoList

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

const app = express();

const PORT = 3000;

app.use(express.json()); // Middleware to parse JSON data

// Sample Todo List

let todos = [{ id: 1, task: "Learn Node.js", completed: false }];

// 
 GET all todos

app.get("/todos", (req, res) => {

   res.json(todos);

});

// 
 POST a new todo

app.post("/todos", (req, res) => {

   const newTodo = { id: todos.length + 1, task: req.body.task, completed: false };

   todos.push(newTodo);

   res.status(201).json(newTodo);

});

// Start server

app.listen(PORT, () => {

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

});
```

localhost:3000/todos

Pretty-print


```
[
  {
    "id": 1,
    "task": "Learn Node.js",
    "completed": false
  }
]
```

3.Create a NodeJS based script file that provides implementation for pwd command from Node shell.

```
const CurrentDirectory = process.cwd();

console.log("Current Work Directory:", CurrentDirectory);

// Example: Handling an error in a file read operation

const fs = require('fs');

fs.readFile('somefile.txt', 'utf8', (err, data) => {

  if (err) {

    console.error('Error reading file:', err); // Now err is properly defined in the callback

    return;

  }

  console.log('File content:', data);

});
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\3no.js"
Current Work Directory: C:\Users\Admin\Desktop\node 19 prac
File content: hellow world!
PS C:\Users\Admin\Desktop\node 19 prac>
```

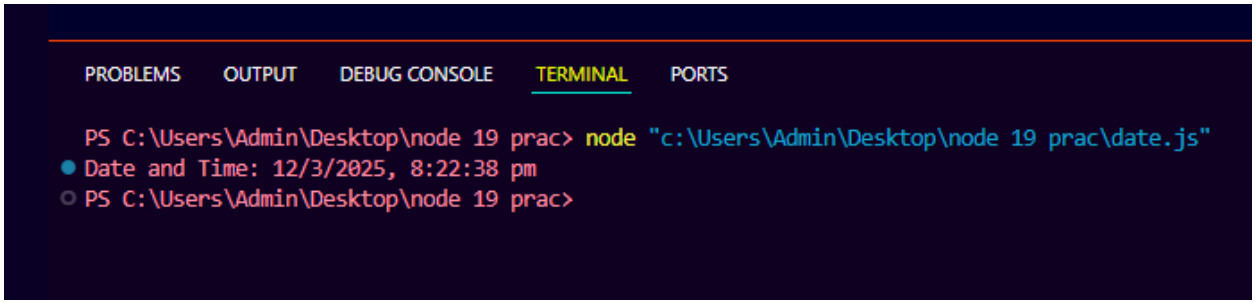
4.Demonstrate a user defined date module that can give you the current date and time.

Mydatetime.js

```
module.exports.getDateTime = () => {

   return new Date().toLocaleString();

};
```

////////////////////***date.js***///////////////

```
const myDate = require("./mydatetime");

console.log("Date and Time:", myDate.getDateTime());
```



5.Create a user defined module named Math with four functions Addition, Subtraction, Multiplication, Division and export them. Import Math module form other Node JS Script file and invoke all the four functions to perform operations on given input command from Node shell.

Math.js

```
function Addition(a, b) {
   return a + b;
}
function subtraction(a, b) {
   return a - b;
}
function multiplication(a, b) {
   return a * b;
}


function division(a, b) {
   if (b == 0) {
      return "Error: Division by zero";
   }
   return a / b;
}


module.exports = {
   Addition,
```

```
    subtraction,
    multiplication,
    division,
};
```

////////////////////////***app.js***/////////////////////

```
const math = require('./math');
const operation = process.argv[2];
const num1 = parseFloat(process.argv[3]);
const num2 = parseFloat(process.argv[4]);
let result;
switch (operation) {
    case 'add':
        result = math.Addition(num1, num2);
        break;
    case 'sub':
        result = math.subtraction(num1, num2);
        break;
    case 'mul':
        result = math.multiplication(num1, num2);
        break;
    case 'div':
        result = math.division(num1, num2);
        break;
    default:
        result = 'Invalid operation. Please use "add", "sub", "mul", or "div".';
```

```
}

console.log(`Result: ${result}`);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\aapp.js"
Result: Invalid operation. Please use "add", "sub", "mul", or "div".
PS C:\Users\Admin\Desktop\node 19 prac> node aapp.js add 20 8
Result: 28
PS C:\Users\Admin\Desktop\node 19 prac> node aapp.js mul 20 5
Result: 100
PS C:\Users\Admin\Desktop\node 19 prac> node aapp.js div 100 10
Result: 10
PS C:\Users\Admin\Desktop\node 19 prac> node aapp.js sub 50 20
Result: 30
PS C:\Users\Admin\Desktop\node 19 prac>
```

6. Demonstrate a Node JS Script that displays a message Welcome to Node JS through loop, with delay in between the iterations Using setTimeOut()

```
function displayMessage(i,times){
  if (i<times){
  setTimeout(()=>
  {
  console.log("welcome to Node js");
  displayMessage(i + 1,times);
  },1000);
  }
  }
  const iterations = 10;
  displayMessage(0,iterations);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\6no.js"
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
welcome to Node js
```

7.Demonstrate a Node.js file that will convert the output "Hello World!"

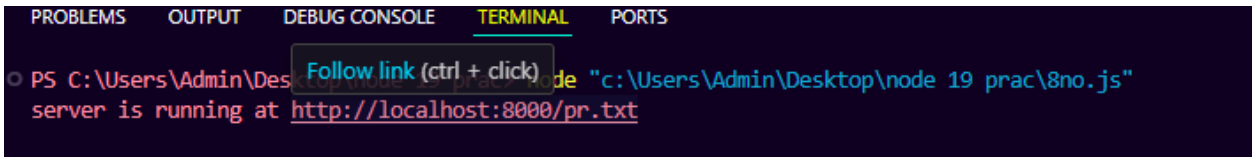
into upper-case letters Using Node js.

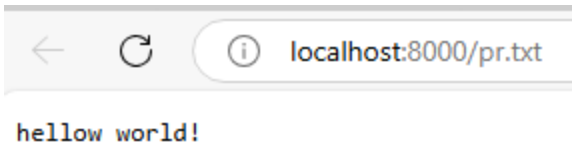
```
const message ="Hellow World";

const UpperCaseMessage =message.toUpperCase();

console.log(UpperCaseMessage);



const LowerCaseMessage =message.toLocaleLowerCase();

console.log(LowerCaseMessage);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\7no.js"
HELLOW WORLD
hellow world
PS C:\Users\Admin\Desktop\node 19 prac>
```

8.Demonstrate a Node.js file that open the requested file and return the content to the client Using Node js.

```
const http = require("http");

const fs = require ("fs");

const path = require ("path");

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

   const filepath = path.join(__dirname,"pr.txt");

   fs.readFile(filepath,"utf8",(err,data)=>{

      if(err){

         res.writeHead(500,{"content.type ":"text/plain"});

         res.end("Error reading file ");

       }else{

         res.writeHead(200,{"content.type":"text/plain " });

         res.end(data);

      }

   })

})

const PORT=8000;

server.listen(PORT,()=>{

   console.log("server is running at http://localhost:8000/pr.txt");

});
```




localhost:8000/pr.txt
hellow world!

9. Demonstrate a Node.js built-in File System module Using Node js.

```
const fs = require("fs");

const fileName ="demo.txt";

fs.writeFile(fileName,"hello,this is a Node.js file system demo",(err)=>{

   if (err)throw  err;

   console.log("file constent and data written");

   fs .readFile(fileName,"utf8",(err,data)=>{

      if (err)throw err;

      console.log("data appended");

      const newfileName="new_demo.txt";

      fs.rename(fileName,newfileName,(err)=>{

         if(err)throw err;

         console.log("file renamed");

         fs.unlink(newfileName,(err)=>{

            if (err)throw err;

            console.log("file deleted");

         });


      });

   });

});
```

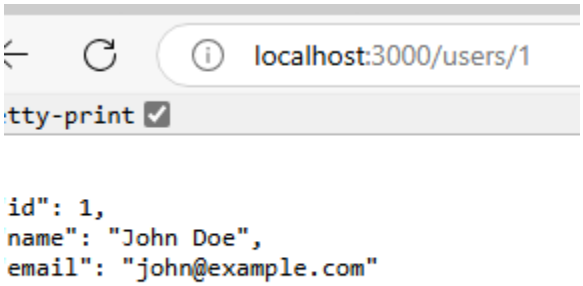
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node 19 prac> node "c:\Users\Admin\Desktop\node 19 prac\no9.js"
file constent and data written
data appended
file renamed
file deleted
PS C:\Users\Admin\Desktop\node 19 prac>
```

10.To demonstrate REST API in Node JS

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

const app = express();

const PORT = 3000;


app.use(express.json()); // Middleware to parse JSON request bodies


// Sample data (acts as a temporary database)

let users = [

  { id: 1, name: "John Doe", email: "john@example.com" },

  { id: 2, name: "Jane Doe", email: "jane@example.com" }

];


// 
 GET request - Fetch all users

app.get('/users', (req, res) => {

  res.json(users);

});


// 
 GET request - Fetch a single user by ID

app.get('/users/:id', (req, res) => {

  const user = users.find(u => u.id === parseInt(req.params.id));

  if (!user) return res.status(404).json({ message: "User not found" });

  res.json(user);

});
```

```
// 
 POST request - Add a new user
app.post('/users', (req, res) => {
  const newUser = {
    id: users.length + 1,
    name: req.body.name,
    email: req.body.email
  };
  users.push(newUser);
  res.status(201).json(newUser);
});

// 
 PUT request - Update a user by ID
app.put('/users/:id', (req, res) => {
  const user = users.find(u => u.id === parseInt(req.params.id));
  if (!user) return res.status(404).json({ message: "User not found" });

  user.name = req.body.name || user.name;
  user.email = req.body.email || user.email;

  res.json(user);
});

// 
 DELETE request - Remove a user by ID
app.delete('/users/:id', (req, res) => {
  users = users.filter(u => u.id !== parseInt(req.params.id));
```

```
  res.json({ message: "User deleted successfully" });
});


// Start the server
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```



localhost:3000/users/1-h

Pretty-print ☑


```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com"
}
```


localhost:3000/users/1%20-H%20-d

Pretty-print ☐


```
{"id":1,"name":"John Doe","email":"john@example.com"}
```


localhost:3000/users/2

Pretty-print ☑


```
{
  "id": 2,
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

11. Create Node JS Application for to stored student information in databas

```
var mysql = require('mysql');

var con = mysql.createConnection({

  host: "localhost",

  user: "yourusername",

  password: "yourpassword"

});

con.connect(function(err) {

  if (err) throw err;

  console.log("Connected!");

});
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:217:24) {
  code: 'ER_NOT_SUPPORTED_AUTH_MODE',
  errno: 1251,
  sqlMessage: 'Client does not support authentication protocol requested by server; consider upgrading MySQL client',
  sqlState: '08004',
  fatal: true
}

Node.js v22.13.1
PS C:\Users\Admin\Desktop\New folder>
```

12. Create Node JS Application for login credentials.

App.js

```
// Import dependencies

const express = require('express');

const session = require('express-session');

// Initialize Express app

const app = express();

// Set up the port

const port = 3000;

// Middleware to handle form data

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

app.use(express.json());

// Set up session handling

app.use(session({

  secret: 'yourSecretKey',

  resave: false,

  saveUninitialized: true

}));

// Sample users (replace with a database in a real application)

const users = {

  admin: { username: 'admin', password: 'admin123' },

  user: { username: 'user', password: 'user123' }

};

// Serve static files (like CSS)

app.use(express.static('public'));
```

```
// Home route
app.get('/', (req, res) => {
  if (req.session.user) {
    return res.redirect('/dashboard');
  }
  res.sendFile(__dirname + '/views/login.html');
});
// Login route
app.post('/login', (req, res) => {
  const { username, password } = req.body;
    // Check if the username exists
  if (users[username] && users[username].password === password) {
    req.session.user = users[username]; // Store user in session
    return res.redirect('/dashboard');
  } else {
    return res.send('Invalid credentials. Please try again.');
  }
});
// Dashboard route (after successful login)
app.get('/dashboard', (req, res) => {
  if (!req.session.user) {
    return res.redirect('/');
  }
  res.send(`<h1>Welcome ${req.session.user.username}</h1><a href="/logout">Logout</a>`);
});
```

```
// Logout route
app.get('/logout', (req, res) => {
  req.session.destroy((err) => {
    if (err) {
      return res.redirect('/dashboard');
    }
    res.redirect('/');
  });
});
// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

////////////////////////////**login.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>
    <link rel="stylesheet" href="style.css">


</head>
<body>
    <h2>Login Form</h2>
```

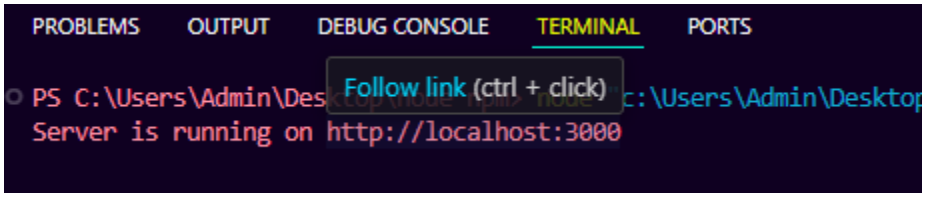
```
    <form action="/login" method="POST">
        <div>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

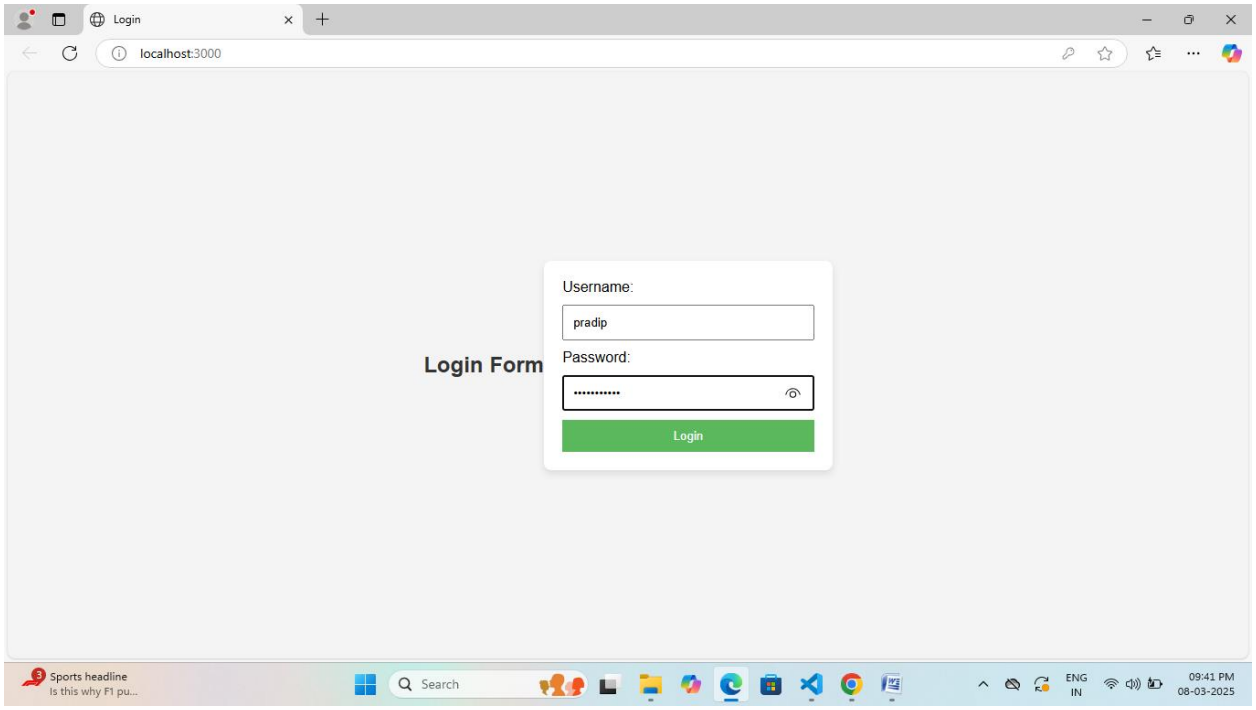
////////////////////////////***style.css***////////////////////////

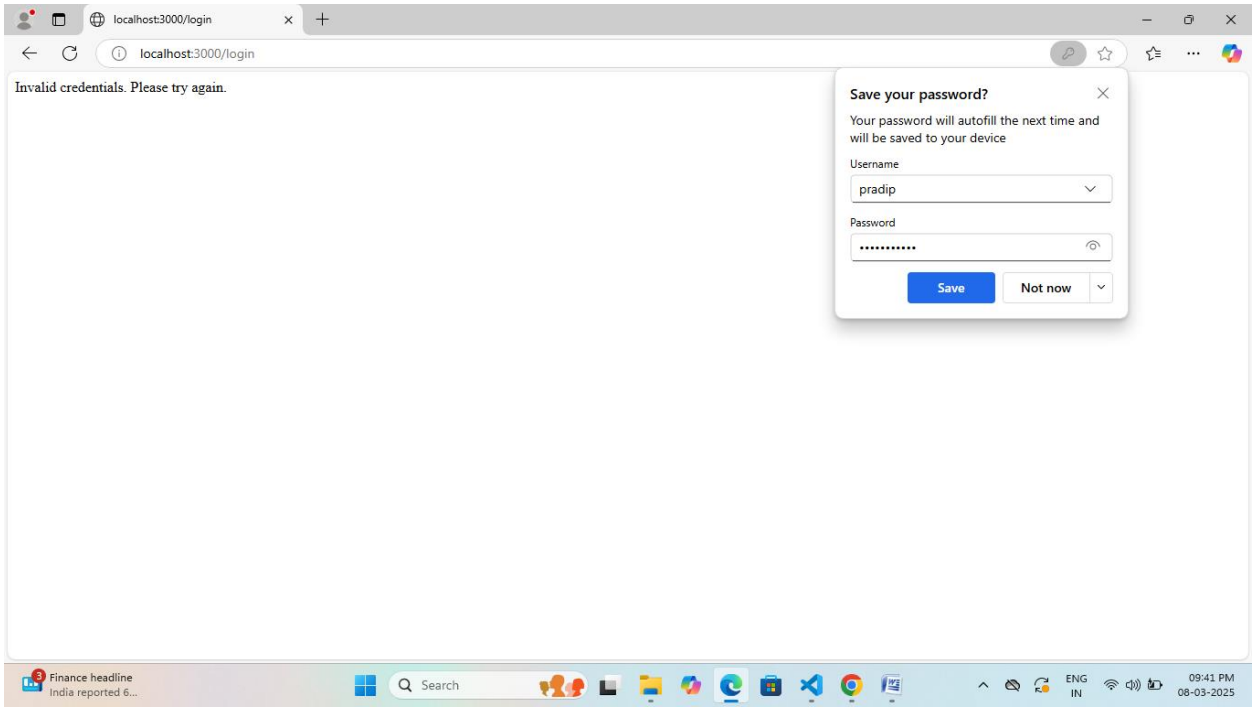
```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

```
h2 {
    color: #333;
}

form {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
}

input {
    padding: 10px;
    margin: 10px 0;
    width: 100%;
    box-sizing: border-box;
}

button {
    padding: 10px 20px;
    background-color: #5cb85c;
    border: none;
    color: white;
    cursor: pointer;
    width: 100%;
```

```
}

button:hover {
   background-color: #4cae4c;
}
```



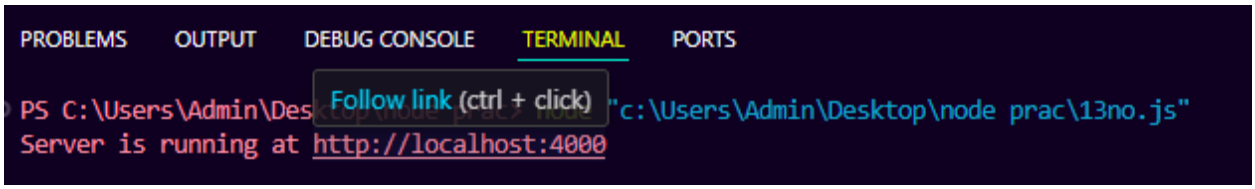


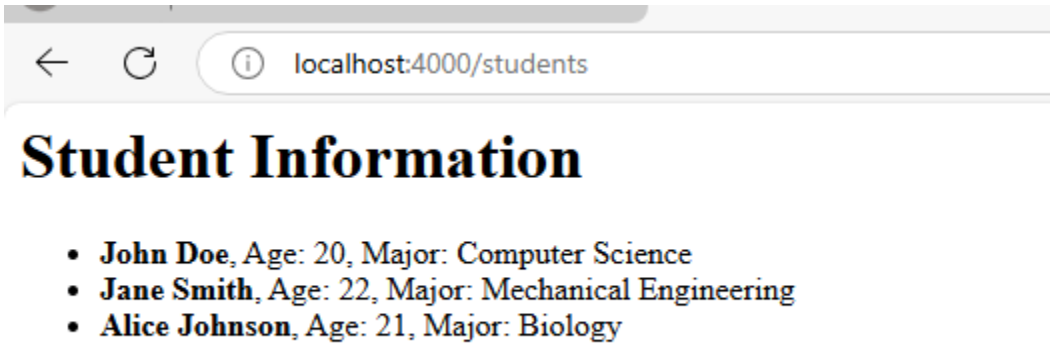

localhost:3000/login
localhost:3000/login
Invalid credentials. Please try again.
Save your password?
Your password will autofill the next time and will be saved to your device
Username
pradip
Password
Save
Not now
Finance headline
India reported 6...
Search
ENG
IN
09:41 PM
08-03-2025

13. Create Node JS Application to display student information.

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

const app = express();

const port = 4000;

// Sample student data

const students = [

  { id: 1, name: 'John Doe', age: 20, major: 'Computer Science' },

  { id: 2, name: 'Jane Smith', age: 22, major: 'Mechanical Engineering' },

  { id: 3, name: 'Alice Johnson', age: 21, major: 'Biology' },

];

// Route to display all students

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

  res.send('<h1>Welcome to the Student Information Page!</h1><a href="/students">View
Student Information</a>');

});

// Route to display student information

app.get('/students', (req, res) => {

  let studentList = '<h1>Student Information</h1><ul>';

  students.forEach(student => {

    studentList += `<li><strong>${student.name}</strong>, Age: ${student.age}, Major:
${student.major}</li>`;

  });

  studentList += '</ul>';

  res.send(studentList);

});
```

```
// Start the server

app.listen(port, () => {

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

});
```





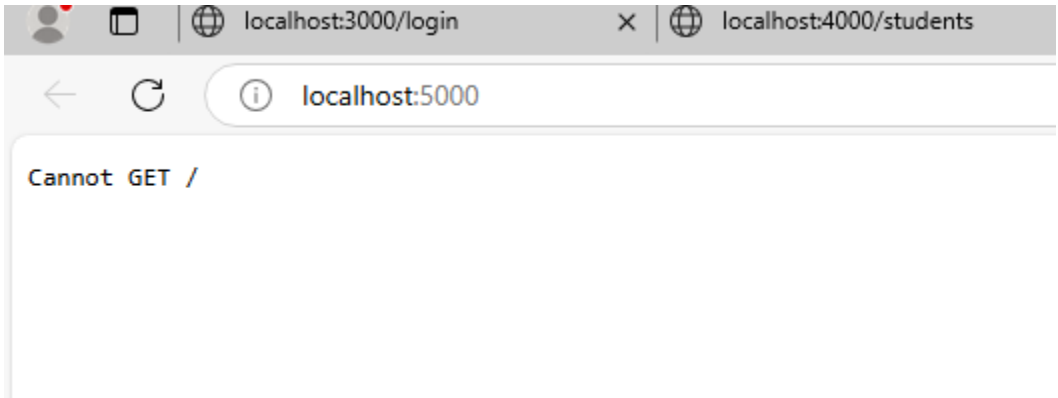
14. Create Node JS Application to update, display and delete student information

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 5000;
// Middleware to parse JSON request bodies
app.use(bodyParser.json());
// Sample in-memory student data
let students = [
  { id: 1, name: 'John Doe', age: 20, major: 'Computer Science' },
  { id: 2, name: 'Jane Smith', age: 22, major: 'Mechanical Engineering' },
  { id: 3, name: 'Alice Johnson', age: 21, major: 'Biology' },
];
// Route to display all students
app.get('/students', (req, res) => {
  res.json(students);
});
// Route to display a single student by ID
app.get('/students/:id', (req, res) => {
  const studentId = parseInt(req.params.id);
  const student = students.find(s => s.id === studentId);
    if (!student) {
    return res.status(404).send('Student not found');
```

```
  }


  res.json(student);
});
// Route to add a new student
app.post('/students', (req, res) => {
  const { name, age, major } = req.body;
  const newStudent = {
    id: students.length + 1,  // Incremental ID for simplicity
    name,
    age,
    major
  };
    students.push(newStudent);
  res.status(201).json(newStudent);
});
// Route to update an existing student
app.put('/students/:id', (req, res) => {
  const studentId = parseInt(req.params.id);
  const { name, age, major } = req.body;
    const student = students.find(s => s.id === studentId);
    if (!student) {
    return res.status(404).send('Student not found');
  }
  student.name = name || student.name;
```

```
  student.age = age || student.age;

  student.major = major || student.major;

   res.json(student);

});

// Route to delete a student

app.delete('/students/:id', (req, res) => {

  const studentId = parseInt(req.params.id);

    const studentIndex = students.findIndex(s => s.id === studentId);

    if (studentIndex === -1) {

    return res.status(404).send('Student not found');

  }

  students.splice(studentIndex, 1);

  res.status(200).send('Student deleted');

});

// Start the server

app.listen(port, () => {

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

});
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Des  Follow link (ctrl + click)  "c:\Users\Admin\Desktop\node prac\14no.js"
Server is running at http://localhost:5000
```


localhost:3000/login
localhost:4000/students
localhost:5000
Cannot GET /

15. Verify how to execute different functions successfully in the Node.js platform.

```
// Example Function 1: Simple Greeting

function greet(name) {

  console.log(`Hello, ${name}!`);

 }

  // Example Function 2: Sum of Two Numbers

 function sum(a, b) {

  return a + b;

 }

 function sub(a, b) {

  return a - b;

 }

 function mul(a, b) {

  return a * b;

 }

 function div(a, b) {

  return a / b;

 }

  // Call the functions

 greet('Alice');

 console.log('Sum of 3 and 4:', sum(3, 4));

 console.log('Sub of 3 and 4:', sub(3, 4));

 console.log('Mul of 3 and 4:', mul(3, 4));

 console.log('Div of 3 and 4:', div(3, 4));
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node prac> node "c:\Users\Admin\Desktop\node prac\15no.js"
Hello, Alice!
Sum of 3 and 4: 7
Sub of 3 and 4: -1
Mul of 3 and 4: 12
Div of 3 and 4: 0.75
PS C:\Users\Admin\Desktop\node prac>
```

16. Write a program to show the workflow of JavaScript code executable by creating web server in Node.js.

```
const http = require('http');

// Create a web server

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

  // Set the response header to indicate a JSON response

  res.setHeader('Content-Type', 'application/json');



  // Create a workflow to handle different routes and responses

  if (req.url === '/') {

    res.statusCode = 200;

    res.end(JSON.stringify({ message: 'Welcome to the Node.js Web Server!' }));

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

    res.statusCode = 200;

    res.end(JSON.stringify({ message: 'This is a simple web server created with Node.js.' }));

  } else {

    res.statusCode = 404;

    res.end(JSON.stringify({ message: 'Not Found' }));

  }

});

const port = 8000;

server.listen(port, () => {

  console.log(`Server is running at http://localhost:${port}`); // ✅ Corrected backticks

});
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node prac> node "c:\Users\Admin\Desktop\node prac\16no.js"
Server is running at http://localhost:8000
```

Follow link (ctrl + click)

localhost:8000

retty-print ☑


```
"message": "Welcome to the Node.js Web Server!"
```


localhost:8000/about

Pretty-print ☐


```
{"message":"This is a simple web server created with Node.js."}
```


localhost:8000/abc

Pretty-print ☐


```
{"message":"Not Found"}
```

17 Write a program to show the workflow of restarting a Node application.

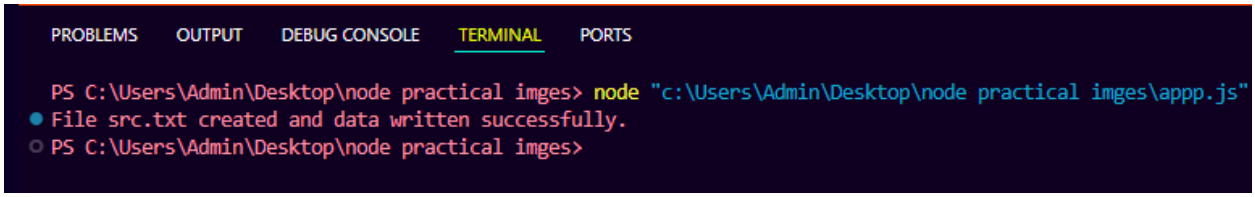
Index.js

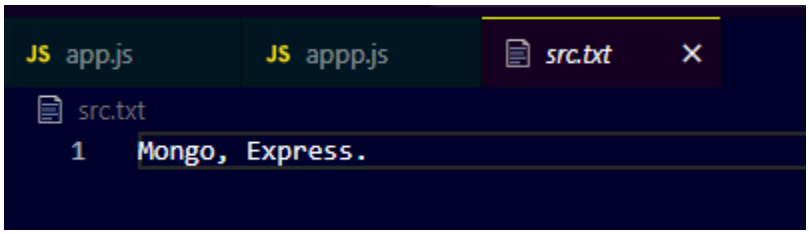
```
console.log("Node.js Application Running...");

// Simulate a long-running application (e.g., a server)

setInterval(() => {

  console.log("Still running...");

}, 5000);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin\Desktop\node1> node "c:\Users\Admin\Desktop\node1\index.js"
Node.js Application Running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
```

18. Create a text file src.txt and add the following data to it. Mongo, Express.

```
const fs = require('fs');

const filePath = 'src.txt';

const data = 'Mongo, Express.';

// Write data to src.txt

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

  if (err) {

    console.error('Error writing to file:', err);

  } else {

    console.log('File src.txt created and data written successfully.');

  }

});
```





19 Implement routing for the Adventure Trails application by embedding the necessary code in the routes/route.js file.

Route.js

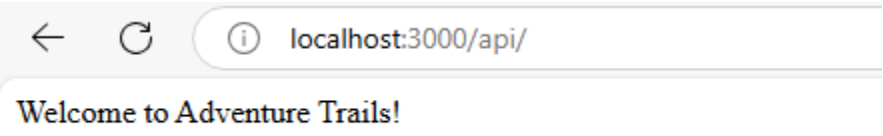
```
const express = require('express');

const router = express.Router();

// Route for the home page

router.get('/', (req, res) => {

  res.send('Welcome to Adventure Trails!');

});

// Route to list all trails

router.get('/trails', (req, res) => {

  const trails = [

    { name: 'Mountain Adventure', difficulty: 'Hard' },

    { name: 'River Exploration', difficulty: 'Medium' },

    { name: 'Forest Trek', difficulty: 'Easy' }

  ];

  res.json(trails);

});

// Route to show details of a specific trail

router.get('/trails/:id', (req, res) => {

  const trails = [

    { id: 1, name: 'Mountain Adventure', difficulty: 'Hard', description: 'A challenging mountain hike.' },

    { id: 2, name: 'River Exploration', difficulty: 'Medium', description: 'A river-side trek with scenic views.' },

    { id: 3, name: 'Forest Trek', difficulty: 'Easy', description: 'A peaceful walk through the forest.' }
```

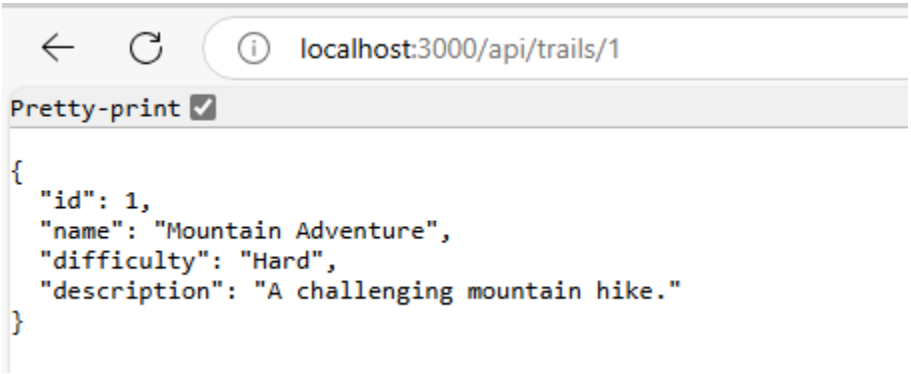
```
    ];
    const trail = trails.find(t => t.id === parseInt(req.params.id));
    if (!trail) {
        return res.status(404).send('Trail not found');
    }
    res.json(trail);
});
// Route to create a new trail (using POST method)
router.post('/trails', (req, res) => {
    const newTrail = req.body;
    // Add logic to save the new trail to a database (if necessary)
    res.status(201).send(`New trail added: ${newTrail.name}`);
});
module.exports = router;
```

////////////////////////////////////**app.js**/////////////////////////////////

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000;
// Middleware to parse JSON body
app.use(bodyParser.json());
// Import routes
const route = require('./routes/route');
```

```
// Use the routes defined in 'routes/route.js'

app.use('/api', route);

// Start the server

app.listen(port, () => {

  console.log(`Adventure Trails app running on http://localhost:${port}`);

});
```





localhost:3000/api/trails/99

Trail not found

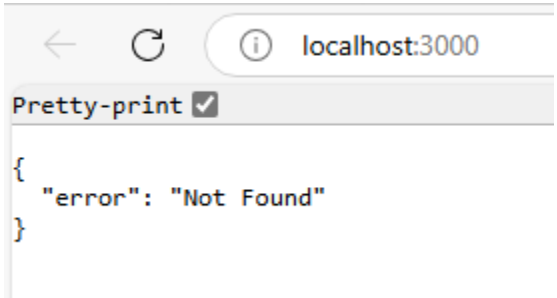
20. In myNotes application: (i) we want to handle POST submissions. (ii) display customized error messages. (iii) perform logging.

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

const morgan = require('morgan');  // For logging

const app = express();

// Middleware to parse JSON data

app.use(express.json());

// Use morgan for logging HTTP requests

app.use(morgan('combined'));

// Handle POST requests at '/submit'

app.post('/submit', (req, res) => {

  const { name, email } = req.body;

  // Check if the name or email is missing

  if (!name || !email) {

    return res.status(400).json({

      error: 'Name and email are required fields.',

    });

  }

  // Respond with success

  res.status(200).json({

    message: 'Submission received. Name: ${name}, Email: ${email}',

  });

});

// Handle 404 errors for undefined routes

app.use((req, res) => {

  res.status(404).json({ error: 'Not Found' });
```

```
});

// Start the server on port 3000

const port = 3000;

app.listen(port, () => {

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

});
```

////////////***login,json***//////////

```
{

   "name": "pradip",

   "email": "pradip@123"

   }
```



http://localhost:3000

POST http://localhost:3000 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
{
    "name": "pradip",
    "email": "dhole@123"
}
```

Cookies Headers (7) Test Results Status: 404 Not Found Time: 15 ms Size: 263 B

Pretty Raw Preview Visualize JSON

```
{
    "error": "Not Found"
}
```

http://localhost:3000/sub POST http://localhost:3000/su

http://localhost:3000/submit

POST http://localhost:3000/submit Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
{
    "name": "pradip",
    "email": "dhole@123"
}
```

Cookies Headers (7) Test Results Status: 200 OK Time: 13 ms Size: 300 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "Submission received. Name: pradip, Email: dhole@123"
}
```

21. Using Node js create a simple Web Designing Page.

```
// Import necessary modules
const express = require("express");
const path = require("path");

// Initialize Express app
const app = express();
const PORT = process.env.PORT || 3000;

// Set view engine to EJS
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));

// Serve static files from 'public' directory
app.use(express.static(path.join(__dirname, "public")));

// Define routes
app.get("/", (req, res) => {
   res.render("index", { title: "Web Designing Lab" });
});

// Start the server
app.listen(PORT, () => {
```

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

////////////////***index.ejs***/////////////

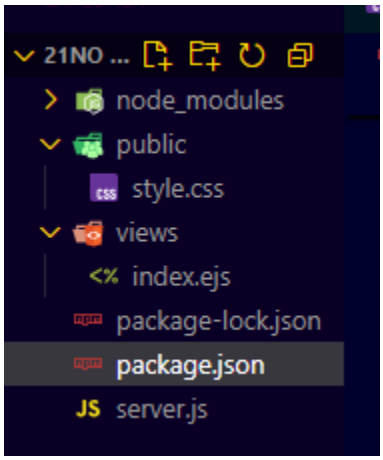
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
  <link rel="stylesheet" href="/style.css">
</head>
<body>
  <h1>welcome to node.js lab</h1>
</body>
</html>
```

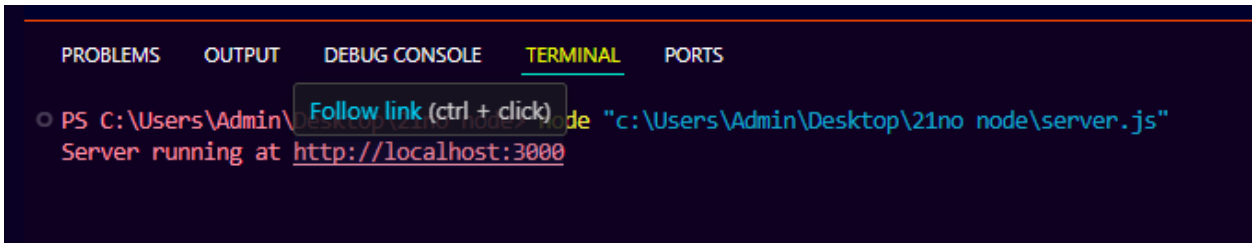
//////////////////////***style.css***/////////////////

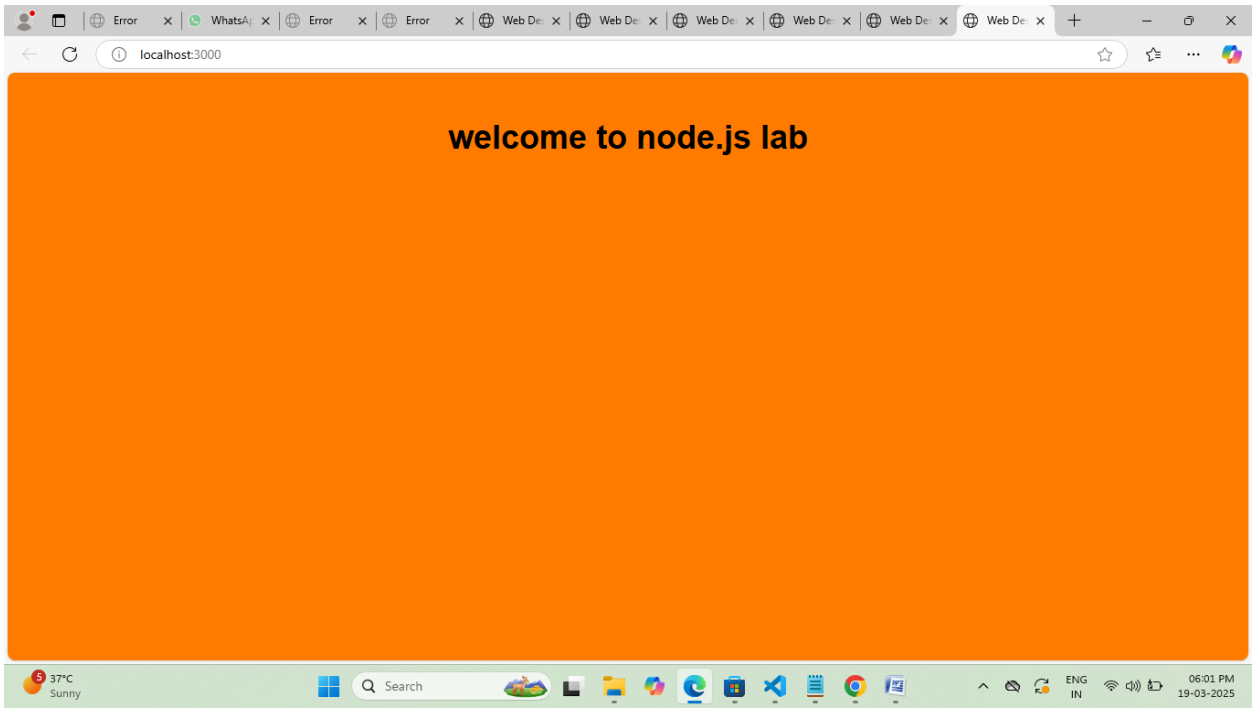
```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 50px;
  font-size: large;
  background-color: rgb(255, 123, 0);

}
```

//////////////////////***package.json***///////////////

```
{
  "name": "web-design-lab",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "cookie-parser": "^1.4.7",
    "ejs": "^3.1.10",
    "express": "^4.21.2"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```


21NO ...
node_modules
public
style.css
views
index.ejs
package-lock.json
package.json
server.js


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Follow link (ctrl + click)
PS C:\Users\Admin\Desktop\21no node> node "c:\Users\Admin\Desktop\21no node\server.js"
Server running at http://localhost:3000


localhost:3000
welcome to node.js lab

22. Dynamic Webpage with Date & Time

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

const app = express();

const PORT = 3000;

// Set view engine

app.set('view engine', 'ejs');

// Serve static files

app.use(express.static('public'));

// Route

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

   const currentTime = new Date().toLocaleString();

   res.render('datetime', { title: "Current Date & Time", time: currentTime });

});

// Start server

app.listen(PORT, () => {

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

});
```

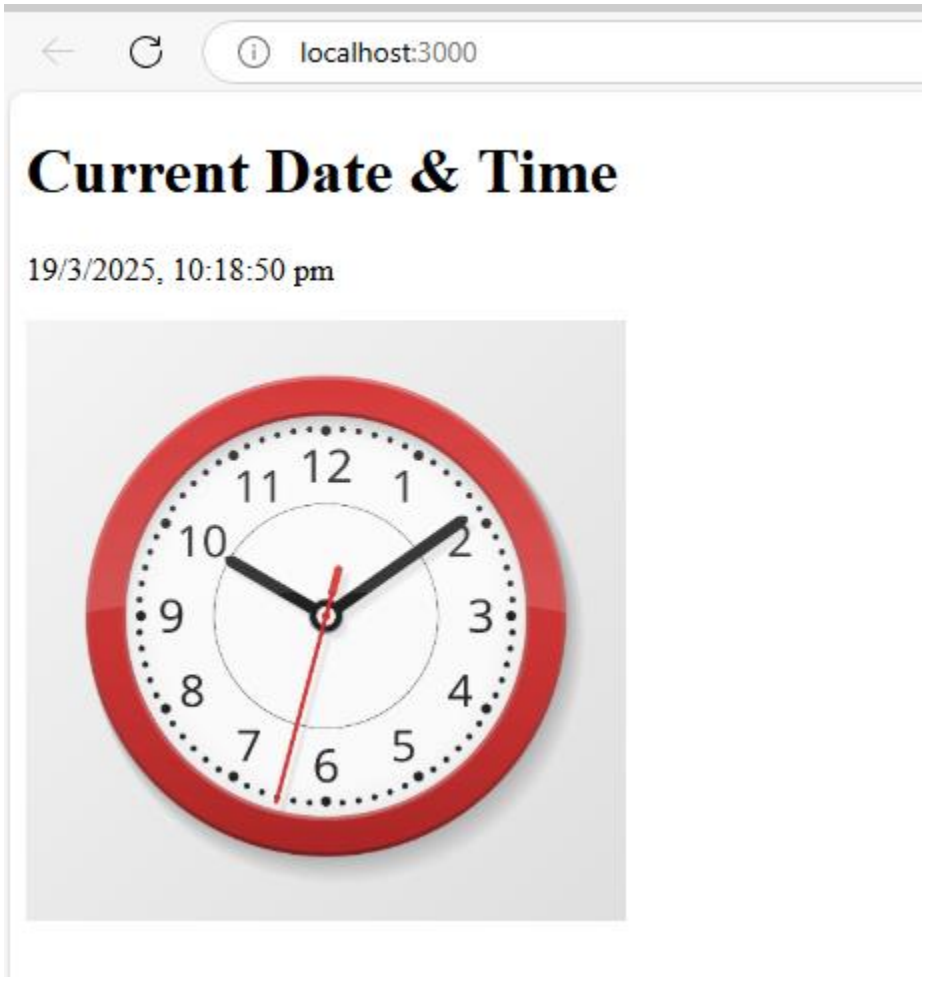
////////////////////datetime.ejs///////////////

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title><%= title %></title>

</head>
```

```
<body>
   <h1>Current Date & Time</h1>
   <p><%= time %></p>
   <img src="https://images.app.goo.gl/9u64JGsEtt2DZgwo9" alt="Watch Image" width="300">
</body>
</html>
```

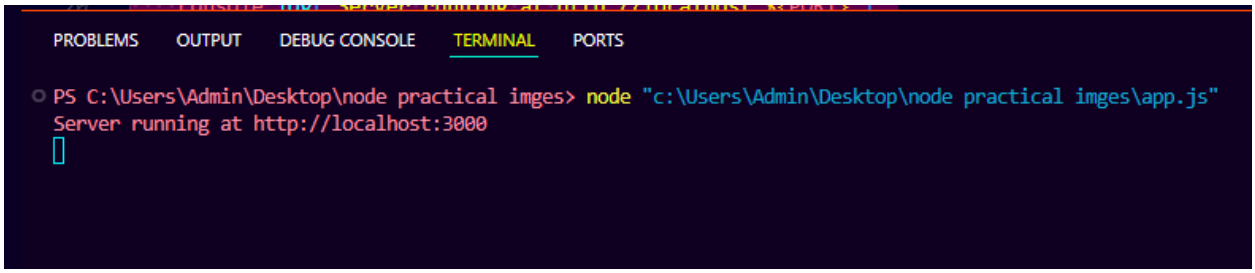
////////////////////////////

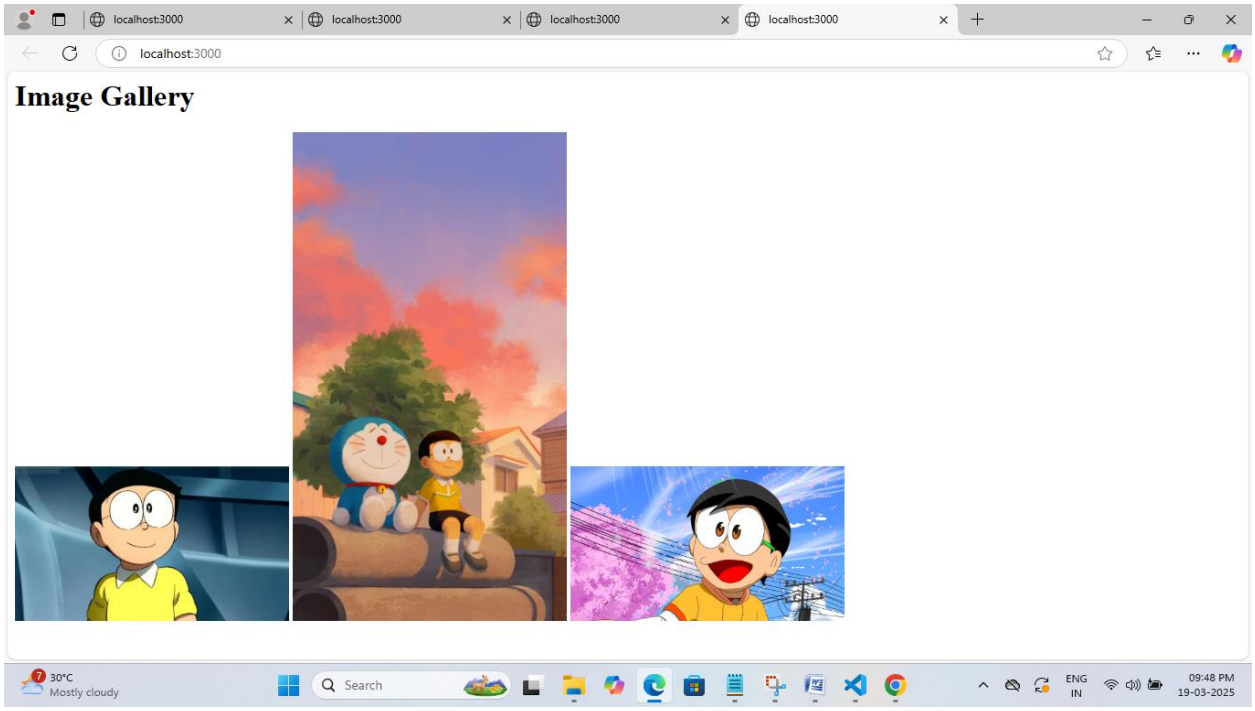
Install EJS:-npm install express ejs



23. Image Gallery Using Static Files

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

const app = express();

const PORT = 3000;

// Serve static files

app.use(express.static('public'));



// Route

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

  res.send(`

    <h1>Image Gallery</h1>

    <img src="/image/Nobita-Background-Photos.jpg"width="300">

    <img src="/image/Nobita-Mobile-Wallpaper.jpg"width="300">

    <img src="/image/Nobita-PC-Wallpaper.jpg"width="300">

  `);

});

// Start server

app.listen(PORT, () => {

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

});
```




localhost:3000
localhost:3000
localhost:3000
localhost:3000
localhost:3000
Image Gallery
30°C
Mostly cloudy
Search
ENG
IN
09:48 PM
19-03-2025

24. Write a program to explain session management using cookies

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

const cookieParser = require('cookie-parser');

const app = express();

const PORT = 3000;

// Use cookie-parser middleware

app.use(cookieParser());

// Route to set session cookie

app.get('/set-session', (req, res) => {

res.cookie('sessionData', JSON.stringify({ username: 'JohnDoe', role: 'admin' }), { maxAge:

60000, httpOnly: true });

res.send('Session cookie has been set!');

});

// Route to read session cookie

app.get('/get-session', (req, res) => {

const sessionCookie = req.cookies.sessionData;

if (sessionCookie) {

res.send(`Session Data: ${sessionCookie}`);

} else {

res.send('No session found');

}

});

// Route to clear session cookie

app.get('/clear-session', (req, res) => {

res.clearCookie('sessionData');
```

```
res.send('Session has been cleared');

});

app.listen(PORT, () => {

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

});
```

////////////////////

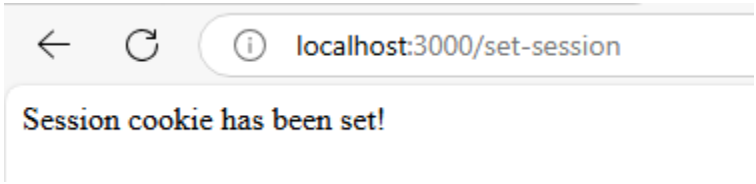
Install dependencies: npm install express cookie-parser

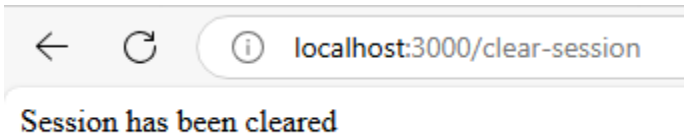
1. cookie-parser Middleware: Parses cookies from incoming requests.

2. Setting a Cookie (/set-session): Stores user session data in a cookie.

3. Reading a Cookie (/get-session): Retrieves and displays stored session data.

4. Clearing a Cookie (/clear-session): Deletes the session cookie





25 Write a program to explain session management using sessions.

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

const session = require('express-session');

const app = express();

const PORT = 3000;

// Use session middleware

app.use(session({

   secret: 'mySecretKey', // Secret key to sign the session ID

   resave: false, // Prevents session from being saved back if unmodified

   saveUninitialized: true, // Saves uninitialized sessions

   cookie: { maxAge: 60000 } // Session expiration time (1 min)

}));

// Route to create/update session data

app.get('/set-session', (req, res) => {

   req.session.username = 'JohnDoe';

   req.session.role = 'admin';

   res.send('Session data has been set!');

});

// Route to retrieve session data

app.get('/get-session', (req, res) => {

   if (req.session.username) {

      res.send(`Session Data: Username - ${req.session.username}, Role - ${req.session.role}`);

   } else {

      res.send('No session found');

   }
```

```
});

// Route to destroy session

app.get('/destroy-session', (req, res) => {

  req.session.destroy((err) => {

    if (err) {

      return res.send('Error destroying session');

    }

    res.send('Session has been destroyed');

  });

});

// Start the server

app.listen(PORT, () => {

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

});
```

//////////////////////

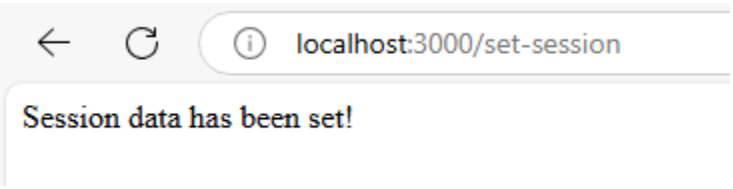
Install:- npm install express express-session

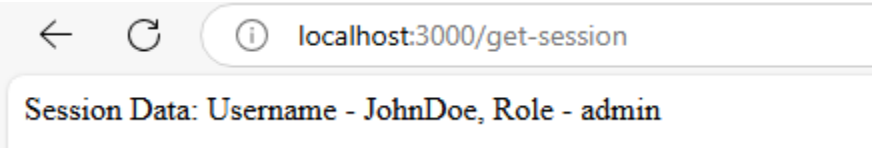
Behind the localhost:-

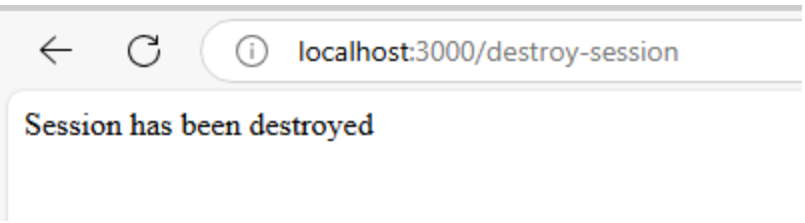
1. Setting a Session (**/set-session**): Stores session data (username & role).

2. Retrieving a Session (**/get-session**): Fetches stored session data.

3. Destroying a Session (**/destroy-session**): Deletes the session.






localhost:3000/destroy-session
Session has been destroyed