

EXP 1

use and demonstrate html5 tags css3 style to design web forms

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Form Example</title>
  <style>
    body { font-family: Arial; margin: 30px; justify-items: center; }
    form { background: #f2f2f2; padding: 20px; border-radius: 10px; width: 300px; }
    label { display: block; margin-top: 10px; }
    input, select, textarea { width: 100%; padding: 5px; margin-top: 5px; }
    button { margin-top: 10px; padding: 8px 15px; width:100%}
  </style>
</head>
<body>

  <h2>Registration Form</h2>

  <form>
    <label for="name">Full Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="example@mail.com"
required>

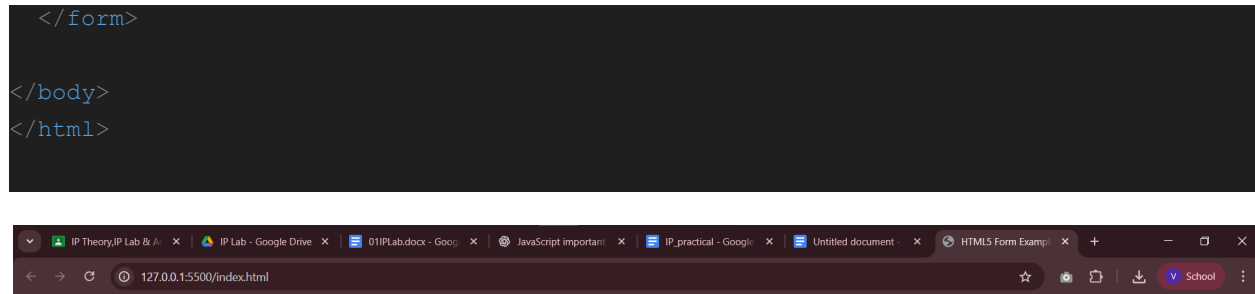
    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob">

    <label for="gender">Gender:</label>
    <select id="gender" name="gender">
      <option value="">Select</option>
      <option>Male</option>
      <option>Female</option>
      <option>Other</option>
    </select>

    <label for="bio">Short Bio:</label>
    <textarea id="bio" rows="3"></textarea>

    <label>
      <input type="checkbox" required> I agree to the terms
    </label>

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



Explanation:

- **HTML5 tags used:**
 - `<form>`, `<input>`, `<select>`, `<textarea>`, `<button>`, `<label>`, `<h2>`
- **Attributes demonstrated:** `type`, `placeholder`, `required`, `for`, `id`, `name`
- **CSS3 styling:** Used basic properties like `background`, `padding`, `border-radius`, and `margin` for simple, clean design.

EXP 2

Use and demonstrations of Javascript basic constructs and arrow function

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JavaScript Basics and Arrow Function</title>
</head>
<body>
  <h2>JavaScript Basic Constructs Example</h2>
  <button onclick="showResult()">Run Script</button>
  <p id="output"></p>

  <script>
    function showResult() {
      // ① Variables and Operators
      let a = 10;
      let b = 5;
      let sum = a + b;

      // ② Condition
      let message = (sum > 10) ? "Sum is greater than 10" : "Sum is 10 or less";

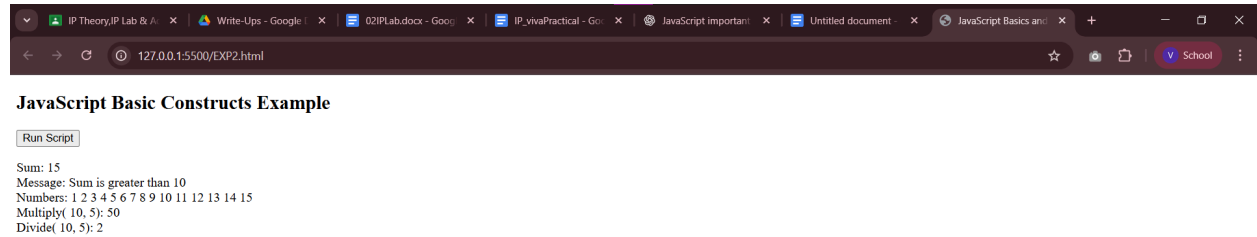
      // ③ Loop
      let numbers = "";
      for (let i = 1; i <= sum; i++) {
        numbers += i + " ";
      }

      // ④ Regular Function
      function multiply(x, y) {
        return x * y;
      }

      // ⑤ Arrow Function
      const divide = (x, y) => x / y;

      // Display output using DOM manipulation
      document.getElementById("output").innerHTML = `
        Sum: ${sum} <br>
        Message: ${message} <br>
        Numbers: ${numbers} <br>
        Multiply( ${a}, ${b}): ${multiply(10, 5)} <br>
        Divide( ${a}, ${b}): ${divide(10, 5)}
      `;
    }
  </script>
</body>
</html>
```

```
</script>
</body>
</html>
```



Explanation of Logic:

- **Variables** (**let**, **const**) → store data (**a**, **b**, **sum**).
- **Operators** → **+**, **>**, **?** : used for arithmetic and conditions.
- **Condition (ternary operator)** → checks if **sum > 10**.
- **Loop** (**for**) → prints numbers 1 to 5.
- **Functions** →
 - **multiply()** → regular function syntax.
 - **divide()** → **arrow function** (shorter way to write functions).
- **DOM Manipulation** → **document.getElementById("output").innerHTML** displays results in the browser.

EXP 3

Use and demonstrations of classes and inheritance and fetch function

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Classes, Inheritance & Fetch Demo</title>
</head>
<body>
  <h2>JavaScript Classes, Inheritance, and Fetch Example</h2>
  <button onclick="showData()">Show Data</button>
  <p id="output"></p>

  <script>
    // ✓ Class: Parent class
    class Person {
      constructor(name, age) {
        this.name = name;
        this.age = age;
      }

      greet() {
        return `Hello, I am ${this.name} and I am ${this.age} years old.`;
      }
    }

    // ✓ Inheritance: Child class extending Person
    class Student extends Person {
      constructor(name, age, course) {
        super(name, age); // call parent constructor
        this.course = course;
      }

      study() {
        return `${this.name} is studying ${this.course}.`;
      }
    }

    // ✓ Fetch Function Example
    async function showData() {
      // Create object from child class
      const student = new Student("Vaishnavi", 21, "Web Development");

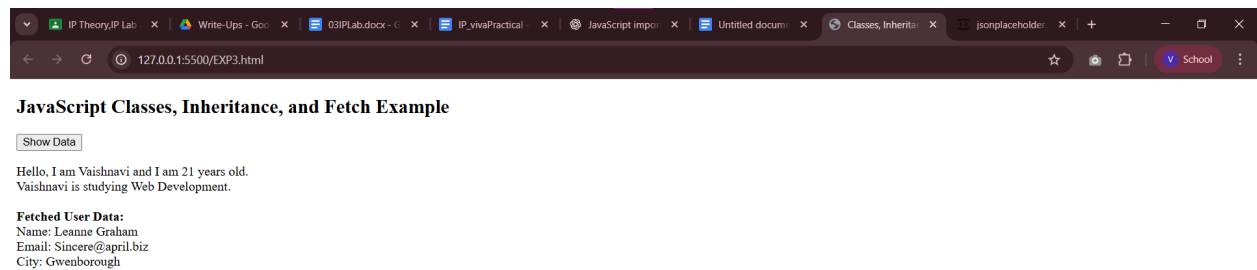
      // Display class data
      let output = `${student.greet()}<br>${student.study()}<br><br>`;
    }
  </script>

```

```
// Fetching sample data from API
const response = await fetch("https://jsonplaceholder.typicode.com/users/1");
const data = await response.json();

output += `<strong>Fetched User Data:</strong><br>
          Name: ${data.name}<br>
          Email: ${data.email}<br>
          City: ${data.address.city}`;

// Show result on webpage
document.getElementById("output").innerHTML = output;
}
</script>
</body>
</html>
```



JavaScript Classes, Inheritance, and Fetch Example

[Show Data](#)

Hello, I am Vaishnavi and I am 21 years old.
Vaishnavi is studying Web Development.

Fetched User Data:
Name: Leanne Graham
Email: Sincere@april.biz
City: Gwenborough

```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Guenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

Explanation of Code Logic:

1. **Class Person** → has **name**, **age**, and a method **greet()**.
2. **Class Student** → extends **Person** using **extends** keyword and adds a new property **course**.
 - Uses **super()** to call parent constructor.
3. **Object Creation** → **new Student("Vaishnavi", 21, "Web Development")**.
4. **Fetch API** → retrieves user data from a sample online API (**jsonplaceholder.typicode.com**).
5. **DOM Manipulation** → displays both class-based output and fetched API data on the page.

EXP 4

Use and demonstrations of React Fundamental(State and Props)

Make sure **Node.js** and **React** are installed.

In command prompt, Run

```
npx create-react-app exp4
cd exp4
npm start
(create app.jsx, main.jsx, and index.html if not already)
```

OR

```
npm create vite@latest exp4
To run
cd exp4
npm run dev
(delete unnecessary files)
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>npm create vite@latest exp4
> npx
> create-vite exp4

|
| o Select a framework:
|   React
|
| o Select a variant:
|   JavaScript
|
| o Use rolldown-vite (Experimental)?:
|   No
|
| o Install with npm and start now?
|   Yes
|
| o Scaffolding project in C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp4...
|
| o Installing dependencies with npm...
added 153 packages, and audited 154 packages in 14s

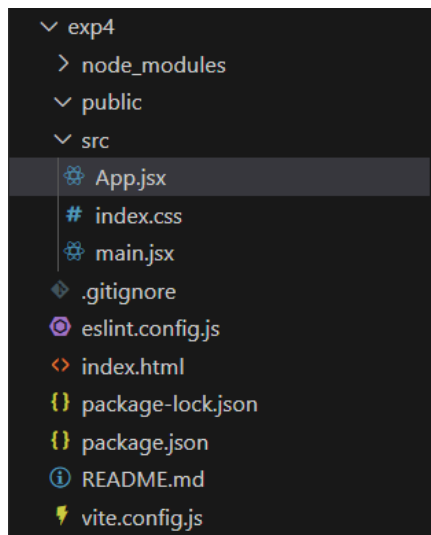
32 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
|
| o Starting dev server...
> exp4@0.0.0 dev
> vite

VITE v7.1.12 ready in 427 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h + enter to show help
```


Folder structure



Replace the code inside `src/App.jsx`

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

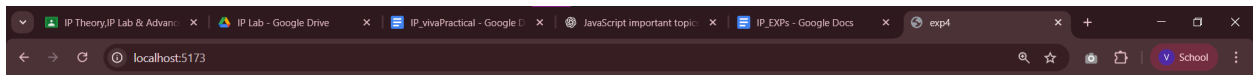
function Greeting(props) {
  // Props received from parent
  return <h3>Hello, {props.name} 🙌</h3>;
}

export default function App() {
  // useState() → manages state inside component
  const [count, setCount] = useState(0);

  // Function to update state
  const increaseCount = () => {
    setCount(count + 1);
  };

  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      {/* Passing prop to child component */}
      <Greeting name="Vaishnavi" />

      {/* Displaying and updating state */}
      <h2>Counter: {count}</h2>
      <button onClick={increaseCount}>Increase</button>
    </div>
  );
}
```



Hello, Vaishnavi 🍌

Counter: 0

Increase

Explanation of Code Logic

Concept	Description
Props	Used to send data from parent to child component. Here, <code>name="Vaishnavi"</code> is passed to the <code>Greeting</code> component.
State	A variable that React tracks . When you update it using <code>setCount</code> , React re-renders the component automatically.
useState() Hook	Creates and manages the state (<code>count</code> here).
Event Handling	The button's <code>onClick</code> event triggers <code>increaseCount()</code> to change the state.

EXP 5

Use and demonstrations of React Router and Single page applications

Make sure **Node.js** and **React** are installed.

In command prompt, Run

```
npx create-react-app exp4
cd exp4
npm start
(create app.jsx, main.jsx, and index.html if not already)
```

OR

```
npm create vite@latest exp4
To run
cd exp4
npm run dev
(delete unnecessary files)
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>npm create vite@latest exp4
> npx
> create-vite exp4

o Select a framework:
  React

o Select a variant:
  JavaScript

o Use rolldown-vite (Experimental)?:
  No

o Install with npm and start now?
  Yes

o Scaffolding project in C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp4...

o Installing dependencies with npm...

added 153 packages, and audited 154 packages in 14s

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

found 0 vulnerabilities

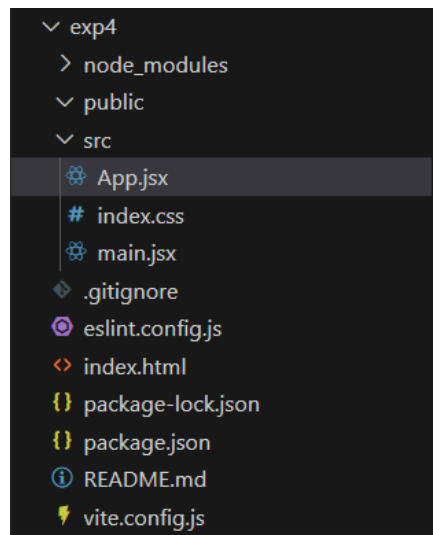
o Starting dev server...

> exp4@0.0.0 dev
> vite

VITE v7.1.12 ready in 427 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h + enter to show help
```

Folder structure



Install React Router:

```
npm install react-router-dom
```

Replace the code inside `src/App.jsx`

```
// App.jsx
import React from "react";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

function Home() {
  return <h2>🏠 Welcome to the Home Page</h2>;
}

function About() {
  return <h2>📄 This is the About Page</h2>;
}

function Contact() {
  return <h2>📧 Contact us at: example@mail.com</h2>;
}

export default function App() {
  return (
    <Router>
      <div style={{ textAlign: "center", marginTop: "40px" }}>
        <h1>React Router Demo (SPA)</h1>

        {/* Navigation Links */}
        <nav>
          <Link to="/" style={{ margin: "10px" }}>Home</Link>
          <Link to="/about" style={{ margin: "10px" }}>About</Link>
        </nav>
      </div>
    </Router>
  );
}
```

```

    <Link to="/contact" style={{ margin: "10px" }}>Contact</Link>
  </nav>


  { /* Route Definitions */ }
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/about" element={<About />} />
    <Route path="/contact" element={<Contact />} />
  </Routes>
</div>
</Router>
);
}

```



React Router Demo (SPA)

[Home](#) [About](#) [Contact](#)

 Contact us at: example@mail.com

Explanation of Code Logic

Concept	Description
BrowserRouter (as Router)	Wraps the entire app to enable routing functionality.
Routes	Holds all route definitions.
Route	Defines which component to render for a given path (URL).
Link	Used instead of <code><a></code> tags — changes route without reloading the page.
SPA Behavior	When you click a link, React changes the displayed component instantly, without refreshing the browser — this makes it a Single Page Application .

EXP 6

Use and demonstrations of Advance React:Hooks(useEffect,useRef)

Make sure **Node.js** and **React** are installed.

In command prompt, Run

```
npx create-react-app exp4
cd exp4
npm start
(create app.jsx, main.jsx, and index.html if not already)
```

OR

```
npm create vite@latest exp4
To run
cd exp4
npm run dev
(delete unnecessary files)
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>npm create vite@latest exp4
> npx
> create-vite exp4

|
| o Select a framework:
|   React
|
| o Select a variant:
|   JavaScript
|
| o Use rolldown-vite (Experimental)?:
|   No
|
| o Install with npm and start now?
|   Yes
|
| o Scaffolding project in C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp4...
|
| o Installing dependencies with npm...
|
added 153 packages, and audited 154 packages in 14s

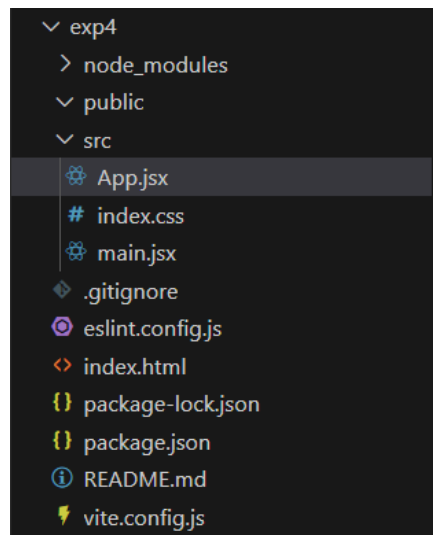
32 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
|
| o Starting dev server...
|
> exp4@0.0.0 dev
> vite

VITE v7.1.12 ready in 427 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h + enter to show help
```

Folder structure



Replace the code inside `src/App.jsx`

```
// App.js
import React, { useState, useEffect, useRef } from "react";

export default function App() {
  const [count, setCount] = useState(0); // useState → to store counter value
  const inputRef = useRef(null);        // useRef → to directly access DOM element

  // useEffect → runs after component renders
  useEffect(() => {
    document.title = `You clicked ${count} times`; // updates browser tab title
  }, [count]); // runs whenever 'count' changes

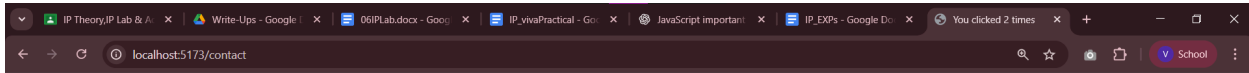
  const focusInput = () => {
    inputRef.current.focus(); // directly focuses the input box using ref
  };

  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      <h1>React Hooks Demo</h1>
      <h2>Count: {count}</h2>

      <button onClick={() => setCount(count + 1)}>Click Me</button>

      <div style={{ marginTop: "30px" }}>
        <input ref={inputRef} type="text" placeholder="Click button to focus me" />
        <br />
        <button onClick={focusInput} style={{ marginTop: "10px" }}>
          Focus Input
        </button>
      </div>
    </div>
  );
}
```

```
        </button>
      </div>
    </div>
  );
}
```



React Hooks Demo

Count: 2

Click Me

Click button to focus me

Focus Input

Explanation of Code Logic

Hook	Purpose	Example in Code
<code>useState</code>	To store and update dynamic data in a component.	<code>const [count, setCount] = useState(0)</code>
<code>useEffect</code>	Runs side effects — code that happens after render (like updating title, fetching data, etc.).	Changes the browser title every time count changes.
<code>useRef</code>	Gives direct access to a DOM element or stores mutable values.	Focuses the input box using <code>inputRef.current.focus()</code>

EXP 7

Setting up Node.js Environment setup

Node + express

1. Create Project Folder

```
mkdir node-server-demo
```

```
cd node-server-demo
```

2. Initialize Node Project

```
npm init -y
```

3. Install Express

```
npm install express
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>mkdir exp7

C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>cd exp7

C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp7>npm init -y
Wrote to C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp7\package.json:

{
  "name": "exp7",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

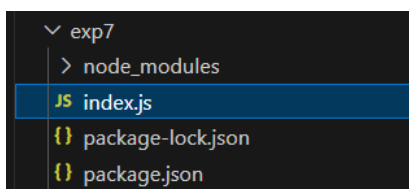
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp7>npm install express

added 68 packages, and audited 69 packages in 5s

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

found 0 vulnerabilities
```

4. Create `index.js` File



```
// Import Express module
const express = require('express');

// Create an Express application
const app = express();

// Define a port number
const PORT = 3000;

// Define a simple route
app.get('/', (req, res) => {
  res.send('Hello from Node.js Express Server!');
});

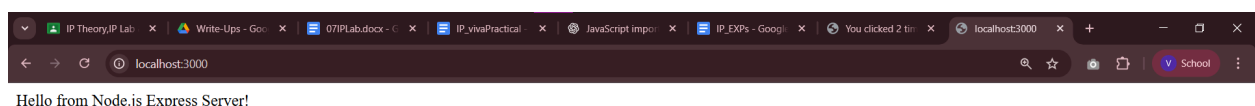
// Start the server
app.listen(PORT, () => {
  console.log(`Server is running at http://localhost:${PORT}`);
});
```

5. Run the Server

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp7>node index.js
Server is running at http://localhost:3000
```

6. Test It

- Open a browser and visit → <http://localhost:3000>



Explanation of Code Logic

Code Part	Purpose
<code>require('express')</code>	Imports the Express framework.
<code>app.get('/', ...)</code>	Defines a route for the home page.
<code>res.send()</code>	Sends a response to the client.
<code>app.listen(PORT)</code>	Starts the server and listens for requests on the given port.

EXP 8

Simple Web Server using nodejs

Only node

1. Create Project Folder

```
mkdir node-server-demo
```

```
cd node-server-demo
```

2. Initialize Node Project

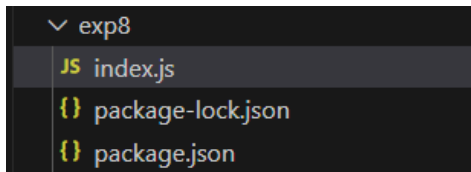
```
npm init -y
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>mkdir exp8
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>cd exp8
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp8>npm init -y
Wrote to C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp8\package.json:

{
  "name": "exp8",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp8>npm i
up to date, audited 1 package in 1s
found 0 vulnerabilities
```

4. Create `index.js` File



```
// Import built-in http module
const http = require('http');

// Create the server
const server = http.createServer((req, res) => {
  res.end('<h1>Hello, World!</h1><p>This is a simple Node.js web server.</p>');
});
```

```
});

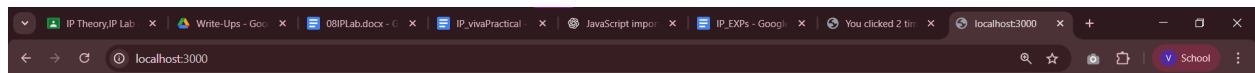
// Start listening on the defined port
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

5. Run the Server

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp8>node index.js
Server running at http://127.0.0.1:3000/
```

6. Test It

- Open a browser and visit → <http://localhost:3000>



Hello, World!

This is a simple Node.js web server.

Explanation of Code Logic

Line	Description
<code>require('http')</code>	Imports Node's built-in HTTP module.
<code>http.createServer()</code>	Creates a new web server.
<code>(req, res)</code>	Request and Response objects.
<code>res.end()</code>	Sends the final response to the browser.
<code>server.listen()</code>	Starts the server on the given port.

EXP 9

Demo of Promise using Async,Await

Create file promiseDemo.js

```
// Function that returns a Promise
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const success = true; // change to false to test rejection
      if (success) {
        resolve("✅ Data fetched successfully!");
      } else {
        reject("❌ Failed to fetch data!");
      }
    }, 2000);
  });
}

// Async function using await
async function getData() {
  console.log("Fetching data... please wait...");

  try {
    const result = await fetchData(); // waits for Promise to resolve
    console.log(result);
  } catch (error) {
    console.error(error);
  }
}

// Run the async function
getData();
```

To run

In cmd, run `node promiseDemo.js`

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>node exp9.js
Fetching data... please wait...
✅ Data fetched successfully!
```

(If you set `success = false`, you'll see `❌ Failed to fetch data!.`)

Explanation of Code Logic

Concept	Description
Promise	Represents a value that will be available later (either success or failure).
resolve()	Called when the operation succeeds.
reject()	Called when the operation fails.
async	Declares a function that can use <code>await</code> .
await	Pauses execution until the Promise resolves or rejects.
try...catch	Handles success and error cases neatly.

EXP 10

Express app using template engine:ejs for login page

Step ① — Create a folder

```
mkdir express-ejs-login  
cd express-ejs-login
```

Step ② — Initialize Node project

```
npm init -y
```

Step ③ — Install dependencies

```
npm install express ejs body-parser
```

```
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>mkdir exp10  
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical>cd exp10  
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp10>npm init -y  
Wrote to C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp10\package.json:  
  
{  
  "name": "exp10",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}  
  
C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp10>npm install express ejs body-parser  
added 76 packages, and audited 77 packages in 7s  
  
16 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

✿ 2. Project Structure

```
exp10  
├── node_modules  
├── views  
│   ├── login.ejs  
│   └── success.ejs  
├── app.js  
├── package-lock.json  
└── package.json
```

📄 3. Code for app.js

```
// Import modules
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

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

// Set EJS as template engine
app.set('view engine', 'ejs');

// Route: Home/Login Page
app.get('/', (req, res) => {
  res.render('login'); // Renders login.ejs
});

// Route: Handle Form Submission
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  // Simple validation logic
  if (username === 'admin' && password === '1234') {
    res.render('success', { user: username });
  } else {
    res.send('<h2>Invalid Credentials. Please try again!</h2>');
  }
});

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

🧠 4. Code for views/login.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login Page</title>
  </head>
  <body style="font-family: Arial; text-align: center; margin-top: 100px;">
    <h2>Login Form</h2>
    <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>

```

✓ 5. Code for views/success.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <title>Login Success</title>
  </head>
  <body style="font-family: Arial; text-align: center; margin-top: 100px;">
    <h2>Welcome, <%= user %> 🎉</h2>
    <p>You have logged in successfully.</p>
  </body>
</html>

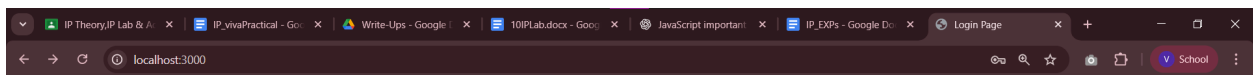
```

▶ 6. Run the App

```

C:\Users\dell\Documents\TEIT_SEM-V\IP\IP_Practical\exp10>node app.js
Server running at http://localhost:3000

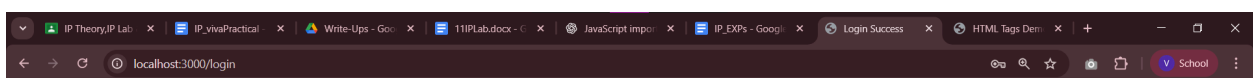
```



Login Form

Username:

Password:



Welcome, admin 🎉

You have logged in successfully.



Invalid Credentials. Please try again!

Explanation

Concept	Description
EJS	Template engine used to embed JS inside HTML (<code><%= %></code>).
body-parser	Parses form data from POST requests.
res.render()	Renders EJS files from the <code>views</code> folder.
app.post()	Handles form submissions.

EXP 11

Create List, Frames, Forms, Multimedia using html tags

Create demo.html file

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>HTML Tags Demo</title>
</head>
<body>

  <h3>1. Lists</h3>
  <ul>
    <li>Unordered item A</li>
    <li>Unordered item B</li>
  </ul>
  <ol>
    <li>Ordered item 1</li>
    <li>Ordered item 2</li>
  </ol>
  <dl>
    <dt>Term</dt><dd>Definition</dd>
  </dl>

  <h3>2. Frame (iframe)</h3>
  <!-- embeds another page -->
  <iframe src="https://example.com" width="300" height="120" title="example
frame"></iframe>

  <h3>3. Simple Form</h3>
  <form action="#" method="post">
    <label>Name: <input type="text" name="name" required></label><br>
    <label>Email: <input type="email" name="email"></label><br>
    <label>Gender:
      <select name="gender">
        <option value="">--</option><option>Male</option><option>Female</option>
      </select>
    </label><br>
    <label><input type="checkbox" name="agree" required> I agree</label><br>
    <button type="submit">Submit</button>
  </form>

  <h3>4. Multimedia</h3>
  <!-- audio (controls shown) -->
  <audio controls>
    <source src="sample-audio.mp3" type="audio/mpeg">
```

```

    Your browser does not support audio.
</audio>
<br>
<!-- video (controls shown) -->
<video width="320" height="180" controls>
  <source src="sample-video.mp4" type="video/mp4">
  Your browser does not support video.
</video>
</body>
</html>

```

HTML Tags Demo

127.0.0.1:5500/exp11.html

V School

1. Lists

- Unordered item A
- Unordered item B

- Ordered item 1
- Ordered item 2

Term	Definition

2. Frame (iframe)

Example Domain

This domain is for use in

3. Simple Form

Name:

Email:

Gender:

☐ I agree

4. Multimedia

0:00 / 0:00

0:00

Explanation:

`ul`, `ol`, `dl` — unordered, ordered, description lists.

`iframe` replaces old `<frame>` and embeds external pages.

`<form>` with `input`, `select`, `checkbox`, and `button` shows basic form elements.

`<audio>` and `<video>` need actual media files (`sample-audio.mp3`, `sample-video.mp4`) in the same folder or valid URLs.

EXP 12

Use of Breadcrumb in bootstrap

Create breadcreumbDemo.html

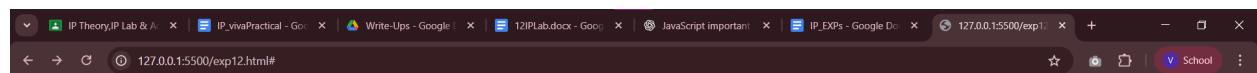
```
<!DOCTYPE html>
<html>
<head>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"

integrity="sha384-EVSTQN3/azprG1Anm3QDgpgJLIIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOMLASjC"
crossorigin="anonymous">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"

integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</head>

<body>
  <h1>This is Iterators1 page</h1>
  <nav style="--bs-breadcrumb-divider: '>';" aria-label="breadcrumb">
    <ol class="breadcrumb">
      <li class="breadcrumb-item"><a href="page11.html">Page11</a></li>
      <li class="breadcrumb-item"><a href="promiseEx1.html">PromiseEx1</a></li>
      <li class="breadcrumb-item active" aria-current="page">Iterators1</li>
    </ol>
  </nav>
</body>

</html>
```



This is Iterators1 page

[Page11](#) > [PromiseEx1](#) > Iterators1

Explanation

Part	Description
<code><nav aria-label="breadcrumb"></code>	Defines a navigation region for screen readers (accessibility).
<code><ol class="breadcrumb"></code>	Ordered list styled as breadcrumb navigation.
<code><li class="breadcrumb-item"></code>	Each navigation item (links or current page).
<code>active + aria-current="page"</code>	Indicates the current page in the breadcrumb trail.