------------------------> 1.js

```
// Write a NodeJS script to take 2 elements 1 & 1000 using file system
module & find
// Kaprekar numbers between them. A Kaprekar number is a number whose
square when
// divided into two parts and such that sum of parts is equal to the
original number and
// none of the parts has value 0.

const fs = require('fs');

// Function to check if a number is a Kaprekar number
function isKaprekarNumber(num)
{
  const square = num * num;
  const squareStr = square.toString();
  const length = squareStr.length;

  for (let i = 1; i < length; i++)
  {
    const leftPart = parseInt(squareStr.slice(0, i));
    const rightPart = parseInt(squareStr.slice(i));

    if (leftPart + rightPart === num && leftPart !== 0 && rightPart !==
0)
    {
      return true;
    }
  }
  return false;
}

// Find Kaprekar numbers between 1 and 1000
const kaprekarNumbers = [];
for (let num = 1; num <= 1000; num++)
{
  if (isKaprekarNumber(num))
  {
    kaprekarNumbers.push(num);
  }
}

// Write Kaprekar numbers to a file
const outputPath = 'kaprekarNumbers.txt';
fs.writeFileSync(outputPath, kaprekarNumbers.join('\n'));

console.log(`Kaprekar numbers between 1 and 1000 have been written to
${outputPath}`);
```

---------------------------------> 2.js

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

// Function to copy file content asynchronously
function copyFileAsync(sourcePath, destinationPath, callback)
{
  fs.readFile(sourcePath, 'utf8', (err, data) =>
```

```
  {
    if (err)
    {
      callback(err);
    }

    fs.writeFile(destinationPath, data, 'utf8', (err) =>
    {
      if (err)
      {
        callback(err);
      }
      callback(null); // Success
    });
  });
}

const sourceFilePath = 'source.txt';
const destinationFilePath = 'destination.txt';

// Copy the file content
copyFileAsync(sourceFilePath, destinationFilePath, (err) =>
{
  if (err)
  {
    console.error('Error:', err);
  } else
  {
    console.log('File copied successfully.');
  }
});
```

>> 3
Que

------------------------------------> 3.js

```
const exp = require("express")
const cookieParser = require("cookie-parser")
const app = exp()

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

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

app.post("/signup",(req,res)=>
{
    const { name ,contactNumbet, email, address, gender, dob} = req.body;

    res.cookie("registered",JSON.stringify({ name ,contactNumbet, email,
address, gender, dob}), {maxAge:15000});

    res.send(`Registration is Successful!! <a href="/details">View
Details<a>`);
```

```
})

app.get("/details", (req,res)=>
{
    const user = req.cookies.registered ?
JSON.parse(req.cookies.registered) :null;

    if (user)
    {
        res.send(`
        <h2>User Information</h2>
        <p>Name: ${user.name}</p>
        <p>Contact Number: ${user.contactNumber}</p>
        <p>Email: ${user.email}</p>
        <p>Address: ${user.address}</p>
        <p>Gender: ${user.gender}</p>
        <p>DOB: ${user.dob}</p>
        <a href="/logout">Logout</a>`)
    }
    else
    {
        res.send(`NO USER FOUND!!  <a href="/">Signup</a>`)
    }
})

app.get("/logout",(req,res)=>
{
    res.clearCookie("registered")
    res.redirect("/")
});

app.listen(3000,()=>
{
    console.log("SERVER STARTED")
})


-----------------------------> Signup.html

<html>
  <head>
    <title>User Signup</title>
  </head>
  <body>
    <h2>User Signup</h2>
    <form action="/signup" method="post">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required /><br />

      <label for="contactNumber">Contact Number:</label>
      <input
        type="text"
        id="contactNumber"
        name="contactNumber"
        required
      /><br />

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required /><br />
```

```
      <label for="address">Address:</label>
      <textarea id="address" name="address" rows="4"
required></textarea><br />

      <label>Gender:</label>
      <input type="radio" name="gender" value="Male" required />Male
      <input type="radio" name="gender" value="Female" required />Female
      <input type="radio" name="gender" value="Others" required
/>Others<br />

      <label for="dob">DOB:</label>
      <input type="date" id="dob" name="dob" required /><br />

      <input type="submit" value="Submit" />
    </form>
  </body>
</html>
```

>> 4 Que

------------------------------>>> 4.js

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const app = express();


// Set Pug as the view engine
app.set('view engine', 'pug');
app.set('views', __dirname);

// Use bodyParser middleware to parse form data
app.use(bodyParser.urlencoded({ extended: false }));

// Serve the student form page
app.get('/', (req, res) =>
{
  res.render('form')
});

// Handle form submission and display data
app.post('/data', (req, res) =>
{
  const studentData = {
    rollNo: req.body.rollNo,
    name: req.body.name,
    division: req.body.division,
    email: req.body.email,
    subject: req.body.subject,
  };

  res.render('data', { studentData });
});

app.listen(3000,()=>
{
```

```
  console.log("SERVER STARTED")
})
```

----------> form.pug

```
doctype html
html
  head
    title Student Form
  body
    h1 Student Form
    form(method="post" action="/data")
      label(for="rollNo") Roll No:
      input(type="number" id="rollNo" name="rollNo" required)
      br
      label(for="name") Name:
      input(type="text" id="name" name="name" required)
      br
      label(for="division") Division:
      input(type="text" id="division" name="division" required)
      br
      label(for="email") Email:
      input(type="email" id="email" name="email" required)
      br
      label(for="subject") Subject:
      input(type="radio" id="fsd" name="subject" value="FSD-2" required)
      label(for="fsd") FSD-2
      input(type="radio" id="coa" name="subject" value="COA" required)
      label(for="coa") COA
      input(type="radio" id="python" name="subject" value="PYTHON-2"
required)
      label(for="python") PYTHON-2
      input(type="radio" id="dm" name="subject" value="DM" required)
      label(for="dm") DM
      input(type="radio" id="toc" name="subject" value="TOC" required)
      label(for="toc") TOC
      br
      input(type="submit" value="Submit")
```

----------> data.pug

```
doctype html
html
  head
    title Student Data
  body
    h1 Student Data
    ul
      li Roll No: #{studentData.rollNo}
      li Name: #{studentData.name}
      li Division: #{studentData.division}
      li Email: #{studentData.email}
      li Subject: #{studentData.subject}
    a(href="/") Back to Form
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 5 Que

--------------------> 5.js

```
// Write a program to upload a text file upto 1MB size only using express
JS. Perform
// necessary validation for file format and size.

// Allows only pdf file to be uploaaded and save in 'specific'. If file
other than pdf then error msg
// only pdf allowed. Also finl o/p through pug file


var exp=require("express");
var app=exp()
const multer=require("multer")

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

var storage=multer.diskStorage({
    destination:"specific",
    filename:function(req,file,cb)
    {
        cb(null,file.originalname)
    }
});

const filter=(req,file,cb)=>
{
    if(file.mimetype=="application/pdf")
    {
        cb(null,true);
    }
    else
    {
        cb(null,false);
        return cb("ONLY PDF FILE ALLOWED")
    }
}
var max=1024*1024

var upload=multer({storage:storage,
limits:{fieldSize:max},fileFilter:filter});

app.post("/uploadfile",upload.single("MyPic"),(req,res)=>
{
    const file=req.file;
    if(file)
    {
        res.set("Content-type","text/html")
        res.send("<h1>File <span style='color:red'>"+ file.originalname +
```

```
        "</span> has been uploaded in <span style='color:red'>" +
file.destination + "</span> folder")
    }
});

app.listen(8001,()=>
{
    console.log("SERVER STARTED")
});


-------------------------------------> multer.html


<html>
    <body>
        <form action="/uploadfile" method="post" enctype="multipart/form-
data">
            <input type="file" name="MyPic" accept=".pdf,.jpg">
            <input type="submit" value="upload"/>
        </form>
    </body>
</html>



>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 6 Que


--------------> 6.js



// Write a script to meet following requirements:
// (1) Create an index.html file & open it on localhost.
// (2) After clicking submit button, it should jump to "savesession"
page. Store username
// in session.
// (3) After saving session, redirect to "fetchsession" page & read
session value. Put a
// logout link button here.
// (4) Jump on "deletesession" page on clicking "logout" link.
// (5) Destroy the session on this page & redirect to index.html page.

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

// Configure session middleware
app.use(session({
  secret: 'my-secret-key',
  resave: false,
  saveUninitialized: true
}));

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

// Display the index.html page
app.get('/', (req, res) =>
{
```

```
  res.sendFile(__dirname + "/index.html");
});

// Save the username in session and redirect to "fetchsession" page
app.post('/savesession', (req, res) => {
  const username = req.body.username;
  req.session.username = username;
  res.redirect('/fetchsession');
});

// Display the session value and a logout link
app.get('/fetchsession', (req, res) =>
{
  const username = req.session.username;

  if (username)
  {
    res.send(`
      <h1>Session Data</h1>
      <p>Username: ${username}</p>
      <a href="/deletesession">Logout</a>
    `);
  }
  else
  {
    res.send('Session data not found. <a href="/">Go back</a>');
  }
});

// Delete the session and redirect to the index.html page
app.get('/deletesession', (req, res) =>
{
  req.session.destroy();
  res.redirect('/');
});

app.listen(3000, () => {
  console.log("SERVER STARTED");
});


----------------------> index.html


<!DOCTYPE html>
<html>
<head>
    <title>Session Demo</title>
</head>
<body>
    <h1>Session Demo</h1>
    <form action="/savesession" method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

>>> 7 Que

------------------------> 7.js

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

const server = http.createServer((req, res) => {
  // Check if the request URL is for the root path ("/")
  if (req.url === '/')
  {
    // Read the HTML file
    fs.readFile(path.join(__dirname, 'simple.html'), 'utf8', (err, data)
=> {
      if (err)
      {
        // Handle file read error
        res.writeHead(500, { 'Content-Type': 'text/plain' });
        res.end('Internal Server Error');
      }
      else
      {
        // Set the HTTP response headers and send the HTML content
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
      }
    });
  }
  else
  {
    // Handle other URLs (e.g., 404 Not Found)
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
  }
});

const port = 3000;

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

-------------------------> simple.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Simple HTML Page</title>
</head>
<body>
 <h1>Hello, World!</h1>
 <p>This is a simple HTML page.</p>
```

```
</body>
</html>



>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 8 Que


----------------> 8.js



// You are tasked with creating a basic Express.js application for a
small online store. The
// application should have the following features:
// 1) Create a server using Express.js that listens on port 3000. Set up
a route to display a
// welcome message on the homepage ("/") when a user visits it.
// 2)Create a route to display a list of products ("/products"). You
should define an array of
// product objects with properties like "name," "description," and
"price."
// 3)Create a route handler that renders this list of products in an HTML
format.
// Implement a dynamic route for product details ("/products/:id").
// 4)Handle 404 errors by displaying a custom error page when a user
tries to access a nonexistent route.
// 5)Your task is to write the Express.js code to achieve the above
functionality.
// 6)Please make sure to include all the necessary dependencies and set
up the Express
// application correctly.
// Note: You can use any template engine (e.g. Pug) of your choice to
render HTML
// pages, or you can send plain HTML as a response. Ensure that you have
installed the
// required packages and set up the project structure accordingly. Write
the Express.js code
// to implement the features mentioned above.

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

// Sample array of product objects
const products = [
  { id: 1, name: 'Product 1', description: 'Description 1', price: 10.99
},
  { id: 2, name: 'Product 2', description: 'Description 2', price: 19.99
},
  { id: 3, name: 'Product 3', description: 'Description 3', price: 7.99
},
];

// Set up Pug as the view engine
app.set('view engine', 'pug');
app.set('views', __dirname);

// Route to display a welcome message on the homepage
app.get('/', (req, res) => {
```

```
  res.send('<h1>Welcome to the Online Store</h1>');
});

// Route to display the list of products
app.get('/products', (req, res) =>
{
  res.render('product', { products });
});

// Dynamic route for product details
app.get('/products/:id', (req, res) =>
{
  const productId = parseInt(req.params.id);
  const product = products.find((p) => p.id === productId);

  if (product)
  {
    res.send(`
      <h1>${product.name}</h1>
      <p>Description: ${product.description}</p>
      <p>Price: $${product.price.toFixed(2)}</p>
      <a href="/products">Back to Products</a>
    `);
  }
  else
  {
    res.status(404).render('error');
  }
});

// Custom error handler for 404 Not Found

app.use((req, res) =>
{
  res.status(404).render('error');
});

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


--------------------> product.pug


doctype html
html
  head
    title Online Store - Products
  body
    h1 Products
    ul
      each product in products
        li
          a(href="/products/#{product.id}") #{product.name}
    a(href="/") Back to Home
```

----------------------> error.pug

```
doctype html
html
  head
    title Online Store - Error
  body
    h1 Error
    p Page not found.
    a(href="/") Back to Home
```

>> 9 que

--------------------> 9.js

```
// You are developing a simple Express.js application for a weather
forecast service. The
// application should have the following features:
// 1) Create an Express.js server that listens on port 3000.
// 2) Set up a route ("/") that displays a basic welcome message when a
user visits the root URL.
// 3) Create a route ("/weather") that accepts a query parameter
"location" and returns the
// weather forecast for the specified location. You can assume that the
weather data is
// available as an object with properties like "location," "temperature,"
and "description."
// For example, if the user accesses "/weather?location=NewYork," the
server should
// respond with the weather information for New York.
// 4)Handle cases where the user doesn't provide a location query
parameter or provides an
// unknown location by displaying an appropriate error message.
// 5)Your task is to write the Express.js code to implement these
features. Make sure to
// include all the necessary dependencies and set up the Express
application correctly.
// 6)Note: You can use any template engine (e.g. Pug) of your choice to
render HTML
// responses, or you can send plain JSON responses.

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

// Sample weather data object (you can replace this with actual weather
data)
const weatherData =
{
  NewYork: { location: 'New York', temperature: 22, description: 'Partly
Cloudy' },
  LosAngeles: { location: 'Los Angeles', temperature: 28, description:
'Sunny' },
  Chicago: { location: 'Chicago', temperature: 18, description: 'Cloudy'
},
```

```
};

// Set up Pug as the view engine
app.set('view engine', 'pug');
app.set('views', __dirname);

// Route to display a welcome message on the homepage
app.get('/', (req, res) =>
{
  res.send('<h1>Welcome to the Weather Forecast Service</h1>');
});

// Route to display the weather forecast for a specified location
app.get('/weather', (req, res) =>
{
  const location = req.query.location;

  if (location && weatherData[location])
  {
    const weather = weatherData[location];
    res.render('weather', { weather });
  }
  else
  {
    res.status(404).render('error', { error: 'Location not found or
weather data not available.' });
  }
});

// Custom error handler for 404 Not Found
app.use((req, res) =>
{
  res.status(404).render('error', { error: 'Page not found.' });
});

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


------------------------------> weather.pug


doctype html
html
  head
    title Weather Forecast
  body
    h1 Weather Forecast
    if weather
      p Location: #{weather.location}
      p Temperature: #{weather.temperature}°C
      p Description: #{weather.description}
    else
      p Weather data not available for this location.
    a(href="/") Back to Home
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 10 Que

---------------> 10.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Button Click Example</title>
</head>
<body>
    <button id="myButton">Click Me</button>
    <script src="10.js"></script>
</body>
</html>
```

-----------------> 10.js

```
// You are building a simple web page that includes a button with the ID
"myButton.
// " You want to display an alert message when the button is clicked.
Write the node js
// code to achieve this using event handling.
// Instructions: Create an HTML file with a button element that has the
ID "myButton."
// Write the JavaScript code to add an event listener to the button
element.
//  When the button is clicked, display an alert message that says,
"Button Clicked!"

// Add an event listener to the button element with the ID "myButton"

const myButton = document.getElementById("myButton");

myButton.addEventListener("click", function()
{
  alert("Button Clicked!");
});
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 11 Que

```
// Write a script to Initialize two variables and increment both the
variables each time and
// display the addition of both the variables at interval of 1 second.

let variable1 = 0;
let variable2 = 0;

const interval = setInterval(() => {
  variable1++;
  variable2++;
```

```
  const addition = variable1 + variable2;
  console.log(`Variable1: ${variable1}, Variable2: ${variable2},
Addition: ${addition}`);
}, 1000); // Execute the function every 1000 milliseconds (1 second)
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 12 Que

---------------> 12.js

```
// Write express js script to load student form using pug file which
contains following
// fields Name(text),Email(email),Course(radio : CE, IT, CSE).Once form
submitted then
// data must be displayed on '/student' page using pug file.Means data
should be submitted
// from express application to PUG file.

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

app.set('view engine', 'pug');
app.set('views', __dirname);
app.use(express.urlencoded({ extended: true }));

app.get('/', (req, res) =>
{
  res.render('StudentForm');
});

app.post('/student', (req, res) =>
{
  const studentData =
  {
    name: req.body.name,
    email: req.body.email,
    course: req.body.course
  };
  res.render('Student', { student: studentData });
});

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

----------------> studentForm.pug

```
doctype html
```

```
html
  head
    title Student Form
  body
    h1 Student Form
    form(method="post" action="/student")
      label(for="name") Name:
      input(type="text" id="name" name="name" required)
      br
      label(for="email") Email:
      input(type="email" id="email" name="email" required)
      br
      label Courses:
      br
      input(type="radio" id="ce" name="course" value="CE" required)
      label(for="ce") CE
      br
      input(type="radio" id="it" name="course" value="IT" required)
      label(for="it") IT
      br
      input(type="radio" id="cse" name="course" value="CSE" required)
      label(for="cse") CSE
      br
      input(type="submit" value="Submit")
```

-----------------------> student.pug

```
doctype html
html
  head
    title Student Data
  body
    h1 Student Data
    if student
      p Name: #{student.name}
      p Email: #{student.email}
      p Course: #{student.course}
    else
      p No student data found.
    a(href="/") Back to Form
```

>>>. 13 Que

```
// Write node js script to handle events as asked below.1) Check the
radius of circle is
// negative or not. If negative then display message “Radius must
bepositive” else
// calculate the area of circle. 2) Check side of square is negative or
not. If negative then
// display message “Side must be positive” else calculate the perimeter
of square.
```

```
const EventEmitter = require('events');

class ShapeCalculator extends EventEmitter
{
  calculateCircleArea(radius)
  {
    if (radius < 0)
    {
      this.emit('negativeRadius', 'Radius must be positive');
    }
    else
    {
      const area = Math.PI * Math.pow(radius, 2);
      this.emit('circleArea', area);
    }
  }

  calculateSquarePerimeter(side)
  {
    if (side < 0)
    {
      this.emit('negativeSide', 'Side must be positive');
    }
    else
    {
      const perimeter = 4 * side;
      this.emit('squarePerimeter', perimeter);
    }
  }
}

const calculator = new ShapeCalculator();

calculator.on('circleArea', (area) =>
{
  console.log(`Area of the circle: ${area.toFixed(2)}`);
});

calculator.on('negativeRadius', (message) =>
{
  console.error(message);
});

calculator.on('squarePerimeter', (perimeter) =>
{
  console.log(`Perimeter of the square: ${perimeter}`);
});

calculator.on('negativeSide', (message) =>
{
  console.error(message);
});

// Example usage:
calculator.calculateCircleArea(5); // Calculate circle area with radius 5
calculator.calculateCircleArea(-3); // Display error message for negative
radius
```

```
calculator.calculateSquarePerimeter(4); // Calculate square perimeter
with side 4
calculator.calculateSquarePerimeter(-2); // Display error message for
negative side
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>. 14 Que

```
// Write a node.js script to create two listeners for a common event and
call their respective
// callbacks. Print number of events associated with an emitter. Remove
one of the
// listeners & call remaining listeners again. Print number of remaining
listeners also

const EventEmitter = require('events');

// Create an instance of EventEmitter
const emitter = new EventEmitter();

// Listener 1 for the common event 'myEvent'
const listener1 = () => {
  console.log('Listener 1 called');
};

// Listener 2 for the common event 'myEvent'
const listener2 = () => {
  console.log('Listener 2 called');
};

// Add both listeners to the 'myEvent' event
emitter.on('myEvent', listener1);
emitter.on('myEvent', listener2);

// Print the number of listeners for the 'myEvent' event
const listenersCount = emitter.listenerCount('myEvent');
console.log(`Number of listeners for 'myEvent': ${listenersCount}`);

// Emit the 'myEvent' event, which will trigger both listeners
emitter.emit('myEvent');

// Remove one of the listeners (listener1)
emitter.removeListener('myEvent', listener1);

// Print the number of remaining listeners for the 'myEvent' event
const remainingListenersCount = emitter.listenerCount('myEvent');
console.log(`Number of remaining listeners for 'myEvent':
${remainingListenersCount}`);

// Emit the 'myEvent' event again, which will trigger only listener2
emitter.emit('myEvent');
```

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 15.js

```
// Write a code snippet to configure the multer middleware to store
uploaded files in a
// specific directory called "uploads"

const express = require('express');
const multer = require('multer');
const path = require('path');

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

// Set up Multer storage configuration
const storage = multer.diskStorage(
{
  destination: (req, file, cb) =>
  {
    cb(null, 'uploads/'); // Specify the directory where uploaded files
will be stored
  },
  filename: (req, file, cb) =>
  {
    const ext = path.extname(file.originalname);
    cb(null, Date.now() + ext); // Use a timestamp as the filename to
avoid conflicts
  },
});

// Initialize Multer with the configured storage
const upload = multer({ storage: storage });

// Define a route to handle file uploads
app.post('/upload', upload.single('file'), (req, res) =>
{
  // File has been uploaded and stored in the "uploads" directory
  res.send('File uploaded successfully.');
});


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

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>16.js

```
// Process a form using post method. Form has fields like username,
password, confirm
// password, gender, submit and reset buttons. After entering all fields,
If password and
// confirm password matches, then form should be processed and all
relevant and
```

```
// selected fields’ values should be printed. Otherwise, by printing
warning message
// in red color, it should terminate. No need to write file having form
elements.

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

// Middleware to parse form data
app.use(express.urlencoded({ extended: true }));

app.get('/', (req, res) =>
{
  // Display the HTML form
  res.send(`
    <html>
    <head>
      <title>Form Example</title>
      <style>
        .error {
          color: red;
        }
      </style>
    </head>
    <body>
      <h1>Form Example</h1>
      <form method="post" action="/process-form">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password"
required><br>

        <label for="confirmPassword">Confirm Password:</label>
        <input type="password" id="confirmPassword"
name="confirmPassword" required><br>

        <label for="gender">Gender:</label>
        <select id="gender" name="gender" required>
          <option value="male">Male</option>
          <option value="female">Female</option>
          <option value="other">Other</option>
        </select><br>

        <input type="submit" value="Submit">
        <input type="reset" value="Reset">
      </form>
    </body>
    </html>
  `);
});

app.post('/process-form', (req, res) =>
{
  const { username, password, confirmPassword, gender } = req.body;

  if (password === confirmPassword)
  {
```

```
    // Passwords match, process the form
    res.send(`
      <h2>Form Submitted Successfully</h2>
      <p>Username: ${username}</p>
      <p>Password: ${password}</p>
      <p>Gender: ${gender}</p>
    `);
  }
  else
  {
    // Passwords don't match, display a warning message in red
    res.send('<p class="error">Password and Confirm Password do not
match.</p>');
  }
});

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