**1.) Create a node.js application to display the implementation of events.**

const http = require('http');

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

eventEmitter.on('serverStarted', (port) => {

  console.log(`Server started on port ${port}`);

});

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

  res.writeHead(200, { 'Content-Type': 'text/plain' });

  res.end('Hello, world!');

});

const port = 3000;

server.listen(port, () => {

  eventEmitter.emit('serverStarted', port);

});

**2.) Create a node.js application to implement listeners.**

const EventEmitter = require('events');

const myEmitter = new EventEmitter();

const listener1 = () => {

  console.log('Listener 1 called');

};

const listener2 = () => {

  console.log('Listener 2 called');

};

myEmitter.on('my-event', listener1);

myEmitter.on('my-event', listener2);

myEmitter.emit('my-event');

**3) Create a node.js application that should open and read from a filename “demo.txt”, if file is empty then it should print on console that “File is empty” otherwise content should be printed on the console.**

const http = require('http');

const fs = require('fs');

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

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

    if (err) {

      console.error(err);

      res.writeHead(500, { 'Content-Type': 'text/plain' });

      res.end('Internal server error');

      return;

    }

    if (data.trim().length === 0) {

      console.log('File is empty');

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end('File is empty');

    } else {

      console.log(`File contents: ${data}`);

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end(data);

    }

  });

});

server.listen(3000 , () => {

  console.log("Server started on port " + 3000);

});

But demo.txt file banani padegi.

**4.) Create an application in node.js for basic unit converter using Events Basic unit converter (Fahernheit to Celsius) C=(F-32)\*5/9.**

const EventEmitter = require('events');

const myEmitter = new EventEmitter();

const convertFtoC = (fahrenheit) => {

const celsius = (fahrenheit - 32) \* 5 / 9;

console.log(`${fahrenheit}°F = ${celsius.toFixed(2)}°C`);

};

myEmitter.on('convert', convertFtoC);

myEmitter.emit('convert', 75);

const http = require('http');

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

eventEmitter.on('convert', (fahrenheit, res) => {

  const celsius = (fahrenheit - 32) \* 5 / 9;

  res.writeHead(200, { 'Content-Type': 'text/html' });

  res.write(`<h1>${fahrenheit}°F = ${celsius}°C</h1>`);

  res.end();

});

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

  if (req.method === 'GET' && req.url === '/') {

    res.writeHead(200, { 'Content-Type': 'text/html' });

    res.write(`

      <form method="POST">

        <label for="fahrenheit">Enter temperature in Fahrenheit:</label>

        <input type="number" id="fahrenheit" name="fahrenheit" required>

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

      </form>

    `);

    res.end();

  } else if (req.method === 'POST' && req.url === '/') {

    let body = '';

    req.on('data', (chunk) => {

      body += chunk.toString();

    });

    req.on('end', () => {

      const fahrenheit = parseFloat(body.split('=')[1]);

      if (!isNaN(fahrenheit)) {

        eventEmitter.emit('convert', fahrenheit, res);

      } else {

        res.writeHead(400, { 'Content-Type': 'text/html' });

        res.write('<h1>Bad request</h1>');

        res.end();

      }

    });

  } else {

    res.writeHead(404, { 'Content-Type': 'text/html' });

    res.write('<h1>Not found</h1>');

    res.end();

  }

});

const port = 3000;

server.listen(port, () => {

  console.log(`Server started on port ${port}`);

});

**5.) Create a node.js application to make http request to open a feedback form which contains following fields –Name, contact number as text fields, Feedback as text area and a submit button. On submission “Thank you for your feedback” message should be displayed in a popup box.**

const http = require('http');

const querystring = require('querystring');

const { exec } = require('child\_process');

const feedbackForm = `

  <html>

    <head>

      <title>Feedback Form</title>

    </head>

    <body>

      <form method="post" action="/submit">

        <div>

          <label for="name">Name:</label>

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

        </div>

        <div>

          <label for="contact">Contact Number:</label>

          <input type="text" id="contact" name="contact">

        </div>

        <div>

          <label for="feedback">Feedback:</label>

          <textarea id="feedback" name="feedback"></textarea>

        </div>

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

      </form>

      <script>

        const form = document.querySelector('form');

        form.addEventListener('submit', () => {

          window.close();

        });

      </script>

    </body>

  </html>

`;

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

  if (req.method === 'GET' && req.url === '/') {

    res.setHeader('Content-Type', 'text/html');

    res.write(feedbackForm);

    res.end();

  }

  else if (req.method === 'POST' && req.url === '/submit') {

    let body = '';

    req.on('data', (chunk) => {

      body += chunk.toString();

    });

    req.on('end', () => {

      const formData = querystring.parse(body);

      const name = formData.name;

      const contact = formData.contact;

      const feedback = formData.feedback;

      res.setHeader('Content-Type', 'text/html');

      res.write(`<script>alert('Thank you for your feedback');</script>`);

      res.end();

    });

  }

});

server.listen(3000, () => {

  console.log('Server listening on port 3000');

});

exec('open http://localhost:3000');

**6.) Create a node.js application which opens a file ‘demo.txt’ and copies the content into another file ‘copy.txt’. Also delete the old file after copying.**

const fs = require('fs');

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

if (err) {

console.error(err);

return;

}

fs.writeFile('copy.txt', data, (err) => {

if (err) {

console.error(err);

return;

}

fs.unlink('demo.txt', (err) => {

if (err) {

console.error(err);

return;

}

console.log('File copied and deleted successfully');

});

});

});

const http = require('http');

const fs = require('fs');

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

  if (req.method === 'GET' && req.url === '/') {

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

      if (err) {

        console.error(err);

        res.writeHead(500, { 'Content-Type': 'text/plain' });

        res.end('Internal server error');

        return;

      }

      fs.writeFile('copy.txt', data, (err) => {

        if (err) {

          console.error(err);

          res.writeHead(500, { 'Content-Type': 'text/plain' });

          res.end('Internal server error');

          return;

        }

        fs.unlink('demo.txt', (err) => {

          if (err) {

            console.error(err);

            res.writeHead(500, { 'Content-Type': 'text/plain' });

            res.end('Internal server error');

            return;

          }

          res.writeHead(200, { 'Content-Type': 'text/plain' });

          res.end('File copied and deleted successfully');

        });

      });

    });

  } else {

    res.writeHead(404, { 'Content-Type': 'text/plain' });

    res.end('Not found');

  }

});

server.listen(3000 , () => {

  console.log("Server started on port " + 3000);

});

**7.) Create a web server application with the http module for the following scenario:**

**A) Create a server nodejs application to verify whether the number is prime or not.**

**B) Accept a number from the input text field of the client page and provide the output values as a response with the click event on button.**

const http = require('http');

const url = require('url');

const querystring = require('querystring');

function isPrime(num) {

  if (num <= 1) return false;

  if (num === 2) return true;

  if (num % 2 === 0) return false;

  for (let i = 3; i <= Math.sqrt(num); i += 2) {

    if (num % i === 0) return false;

  }

  return true;

}

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

  const { pathname, query } = url.parse(req.url);

  const { number } = querystring.parse(query);

  if (pathname === '/') {

    res.writeHead(200, { 'Content-Type': 'text/html' });

    res.write(`

      <html>

        <head>

          <title>Prime Checker</title>

        </head>

        <body>

          <h1>Prime Checker</h1>

          <form action="/check" method="get">

            <label for="number">Enter a number:</label>

            <input type="number" id="number" name="number">

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

          </form>

        </body>

      </html>

    `);

    res.end();

  } else if (pathname === '/check') {

    const num = parseInt(number);

    if (isNaN(num)) {

      res.writeHead(400, { 'Content-Type': 'text/plain' });

      res.write('Invalid input');

      res.end();

    } else {

      const prime = isPrime(num);

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.write(prime ? `${num} is a prime number` : `${num} is not a prime number`);

      res.end();

    }

  } else {

    res.writeHead(404, { 'Content-Type': 'text/plain' });

    res.write('Not found');

    res.end();

  }

});

server.listen(3000,() => {

  console.log("Server running at port " + 3000);

});

**8.) Create a web server application with http module for the following scenario-**

**A) Create a test.txt file and add personal information (name,rollno) in the server system.**

**B) Access the file and display the content as a response to the client (user)request in the browser.**

const http = require('http');

const fs = require('fs');

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

  if (req.method === 'GET' && req.url === '/') {

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

      if (err) {

        res.writeHead(500, { 'Content-Type': 'text/plain' });

        res.write('Internal Server Error');

        res.end();

      } else {

        res.writeHead(200, { 'Content-Type': 'text/plain' });

        res.write(data);

        res.end();

      }

    });

  }

  else {

    res.writeHead(404, { 'Content-Type': 'text/plain' });

    res.write('404 Not Found');

    res.end();

  }

});

server.listen(3000,() => {

    console.log("Server running at port " + 3000);

});

**9.) Implement an express aplication to perform the following operations**

**a. Add two text textboxes in the client page to accept the current year and date of birth and calculate the age of the user.**

**b. Add a button in the client page and provide the output values as response with click event.**

const express = require('express');

const app = express();

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

  res.send(`

    <html>

      <head>

        <title>Age Calculator</title>

      </head>

      <body>

        <form>

          <label for="year">Current year:</label>

          <input type="number" id="year" name="year"><br>

          <label for="dob">Date of birth:</label>

          <input type="date" id="dob" name="dob"><br>

          <button type="button" onclick="calculateAge()">Calculate Age</button><br>

          <label for="age">Age:</label>

          <input type="number" id="age" name="age" readonly>

          <button type="button" onclick="submitForm()">Submit</button>

        </form>

        <script>

          function calculateAge() {

            const year = parseInt(document.getElementById('year').value);

            const dob = new Date(document.getElementById('dob').value);

            const age = year - dob.getFullYear();

            document.getElementById('age').value = age;

          }

          function submitForm() {

            const year = document.getElementById('year').value;

            const dob = document.getElementById('dob').value;

            const age = document.getElementById('age').value;

            const xhr = new XMLHttpRequest();

            xhr.open('POST', '/submit');

            xhr.setRequestHeader('Content-Type', 'application/json');

            xhr.onload = () => alert(xhr.responseText);

            xhr.send(JSON.stringify({ year, dob, age }));

          }

        </script>

      </body>

    </html>

  `);

});

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

  const year = req.body.year;

  const dob = req.body.dob;

  const age = req.body.age;

  console.log(`Year: ${year}, Date of birth: ${dob}, Age: ${age}`);

  res.send(`Year: ${year}, Date of birth: ${dob}, Age: ${age}`);

});

app.listen(3000, () => {

  console.log('Server listening on port 3000');

});

**10.) Create a readable stream using fs module and demonstrate its working.**

const fs = require('fs');

const stream = fs.createReadStream('demo.txt', { encoding: 'utf8' });

stream.on('data', (chunk) => {

  console.log(`Received ${chunk.length} bytes of data.`);

});

stream.on('end', () => {

  console.log('Finished reading the file.');

});

stream.on('error', (err) => {

  console.error(`An error occurred: ${err}`);

});

**11.) Create a writable stream using fs module and demonstrate its working.**

const fs = require('fs');

const stream = fs.createWriteStream('demo.txt', { flags: 'a' });

stream.write('Hello, world!\n');

stream.end();

stream.on('finish', () => {

  console.log('Data has been written to the file.');

});

stream.on('error', (err) => {

  console.error(`An error occurred: ${err}`);

});

12.) Implement an express application.

a) Accept a number from the input text field of a user web page and perform the basic arithmetic operations, increment (++) decrement (-), and square, on the number inside a middleware function of server node.js application.

b) Display the output values in the user web page as a response to the click event from the button.

const express = require('express');

const app = express();

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

app.use(express.json());

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

  res.send(`

    <form action="/" method="POST">

      <input type="number" name="number" />

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

    </form>

  `);

});

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

  const number = parseInt(req.body.number);

  const increment = number + 1;

  const decrement = number - 1;

  const square = number \* number;

  res.send(`

    <p>Increment: ${increment}</p>

    <p>Decrement: ${decrement}</p>

    <p>Square: ${square}</p>

  `);

});

app.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});

13.) implementation in express.

a) Create a text file and add student information (Reg No., Name, Grade) in the server system.

b) Accept a file name from the input text field of a user web page and transfer the requested tile using

sendFile() function from the server as a response to the button click event from the user web page.

const express = require('express');

const fs = require('fs');

const app = express();

const port = 3000;

const studentsFile = 'students.txt';

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

app.use(express.json());

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

  res.send(`

    <form action="/" method="POST">

      <label for="filename">Enter file name:</label>

      <input type="text" name="filename" id="filename" />

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

    </form>

  `);

});

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

  const filename = req.body.filename;

  if (filename == "/") {

    res.status(400).send('Filename is required');

    return;

  }

  const filePath = `${\_\_dirname}/${filename}`;

  if (!fs.existsSync(filePath)) {

    res.status(404).send('File not found');

    return;

  }

  res.sendFile(filePath);

});

app.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});

14.) Create an express application.

a) Set and clear the cookies for a client using the cookies-parser module in the server application.

b) Display the cookies information on the client web page with a click on the show button and clear the cookies information from the system with a click on the reset button on the user web page.

const express = require('express');

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

const app = express();

app.use(cookieParser());

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

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

  const cookieValue = req.cookies['cookie\_name'];

  let message = 'Cookie not set';

  if (cookieValue) {

    message = `Cookie value: ${cookieValue}`;

  }

  res.send(`

    <html>

      <body>

        <h1>Cookie Example</h1>

        <p>${message}</p>

        <form action="/set-cookie" method="post">

          <label for="cookie-value">Enter cookie value:</label>

          <input type="text" name="cookie-value" id="cookie-value">

          <input type="submit" value="Set Cookie">

        </form>

        <form action="/clear-cookie" method="post">

          <input type="submit" value="Clear Cookie">

        </form>

      </body>

    </html>

  `);

});

app.post('/set-cookie', (req, res) => {

  const cookieValue = req.body['cookie-value'];

  res.cookie('cookie\_name', cookieValue, { maxAge: 900000, httpOnly: true });

  res.redirect('/');

});

app.post('/clear-cookie', (req, res) => {

  res.cookie('cookie\_name', '', { maxAge: 0, httpOnly: true });

   res.redirect('/');

});

app.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});

15.) Create an express application with the following requirements

a) Accept Student Name, Reg. no.. Roll. No., Mobile No. and Mail Id from the input text fields of a client page and perform a chain of validations on the data using the express-validator module in the server application.

b) Check all the fields are not empty, minimum and maximum lengths of data.

c) Add a submit button on the client web page to submit the data and display the warning messages if required.

const express = require('express');

const { body, validationResult } = require('express-validator');

const app = express();

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

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

  res.send(`

    <form action="/" method="POST">

      <label for="name">Name:</label>

      <input type="text" name="name" id="name" />

      <br>

      <label for="regno">Reg. No.:</label>

      <input type="text" name="regno" id="regno" />

      <br>

      <label for="rollno">Roll No.:</label>

      <input type="text" name="rollno" id="rollno" />

      <br>

      <label for="mobile">Mobile No.:</label>

      <input type="text" name="mobile" id="mobile" />

      <br>

      <label for="email">Mail id:</label>

      <input type="email" name="email" id="email" />

      <br>

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

    </form>

  `);

});

app.post('/', [

  body('name').notEmpty().withMessage('Name is required'),

  body('regno').notEmpty().withMessage('Reg. No. is required'),

  body('rollno').notEmpty().withMessage('Roll No. is required'),

  body('mobile').notEmpty().withMessage('Mobile No. is required'),

  body('mobile').isMobilePhone('en-IN').withMessage('Mobile No. is invalid'),

  body('email').notEmpty().withMessage('Mail id is required'),

  body('email').isEmail().withMessage('Mail id is invalid'),

], (req, res) => {

  const errors = validationResult(req);

  if (!errors.isEmpty()) {

    return res.status(400).json({ errors: errors.array() });

  }

  const { name, regno, rollno, mobile, email } = req.body;

  console.log(`Name: ${name}`);

  console.log(`Reg. No.: ${regno}`);

  console.log(`Roll No.: ${rollno}`);

  console.log(`Mobile No.: ${mobile}`);

  console.log(`Mail id: ${email}`);

  res.send('Student information submitted successfully!');

});

app.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});

16.) Create a web server application with the http module for the following scenario.

a) Create a server node.js application using http module to verify whether a number is prime or not.

b) Accept a number from the input text field of the client page and provide the output values as a response with the click event on a button.

const http = require('http');

const url = require('url');

const querystring = require('querystring');

const fs = require('fs');

function isPrime(num) {

  if (num <= 1) return false;

  if (num === 2) return true;

  if (num % 2 === 0) return false;

  for (let i = 3; i <= Math.sqrt(num); i += 2) {

    if (num % i === 0) return false;

  }

  return true;

}

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

  const { pathname, query } = url.parse(req.url);

  const { number } = querystring.parse(query);

  if (pathname === '/') {

    res.writeHead(200, { 'Content-Type': 'text/html' });

    res.write(`

      <html>

        <head>

          <title>Prime Checker</title>

        </head>

        <body>

          <h1>Prime Checker</h1>

          <form action="/check" method="get">

            <label for="number">Enter a number:</label>

            <input type="number" id="number" name="number">

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

          </form>

        </body>

      </html>

    `);

    res.end();

  } else if (pathname === '/check') {

    const num = parseInt(number);

    if (isNaN(num)) {

      res.writeHead(400, { 'Content-Type': 'text/plain' });

      res.write('Invalid input');

      res.end();

    } else {

      const prime = isPrime(num);

      const result = prime ? `${num} is a prime number` : `${num} is not a prime number`;

      const data = `${new Date().toLocaleString()}: ${result}\n`;

      fs.appendFile('result.txt', data, err => {

        if (err) console.error(err);

      });

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.write(result);

      res.end();

    }

  } else {

    res.writeHead(404, { 'Content-Type': 'text/plain' });

    res.write('Not found');

    res.end();

  }

});

server.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});

17.) Create a web server application with http module for the following scenario.

a) Create a server node.js application using http module to find the nth Fibonacci number

b) Accept a number from the input text field of the client page and provide the output values as a response with the click event on a button.

const http = require('http');

const fs = require('fs');

const url = require('url');

const querystring = require('querystring');

function fibonacci(n) {

  if (n <= 0) return 0;

  if (n === 1) return 1;

  return fibonacci(n - 1) + fibonacci(n - 2);

}

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

  const { pathname, query } = url.parse(req.url);

  const { n } = querystring.parse(query);

  if (pathname === '/') {

    res.writeHead(200, { 'Content-Type': 'text/html' });

    res.write(`

      <html>

        <head>

          <title>Fibonacci Calculator</title>

        </head>

        <body>

          <h1>Fibonacci Calculator</h1>

          <form action="/calculate" method="get">

            <label for="n">Enter the value of n:</label>

            <input type="number" id="n" name="n">

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

          </form>

        </body>

      </html>

    `);

    res.end();

  } else if (pathname === '/calculate') {

    const num = parseInt(n);

    if (isNaN(num) || num < 0) {

      res.writeHead(400, { 'Content-Type': 'text/plain' });

      res.write('Invalid input');

      res.end();

    } else {

      fs.writeFile('input.txt', num.toString(), (err) => {

        if (err) console.log(err);

        console.log('Input value saved to input.txt');

      });

      const fib = fibonacci(num);

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.write(`The ${num}${num % 10 === 1 && num % 100 !== 11 ? 'st' : num % 10 === 2 && num % 100 !== 12 ? 'nd' : num % 10 === 3 && num % 100 !== 13 ? 'rd' : 'th'} Fibonacci number is: ${fib}`);

      res.end();

    }

  }else {

    res.writeHead(404, { 'Content-Type': 'text/plain' });

    res.write('Not found');

    res.end();

  }

});

server.listen(3000 , () => {

  console.log("Server running at port " + 3000);

});