perplexity

pls continue

Certainly! Continuing with detailed beginner-friendly instructions for experiments 11–15, including setup, code, and explanations.

11. Create a Job Listing Site (Basic Node.js + Express)

Step 1: Setup Project

```
mkdir job-listing-site
cd job-listing-site
npm init -y
npm install express body-parser
```

Step 2: Create `app.js`

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

let jobs = [
  { title: "Frontend Developer", company: "Tech Corp", location: "New York" },
  { title: "Backend Developer", company: "Innovate Ltd", location: "Boston" }
];

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

app.get('/', (req, res) => {
  let jobList = jobs.map(job =>
    `<li><strong>${job.title}</strong> at ${job.company} (${job.location})</li>`
  ).join('');
  res.send(`<h1>Job Listings</h1><ul>${jobList}</ul>
    <h2>Add a Job</h2>
    <form method="post" action="/add">
      <input name="title" placeholder="Job Title" required/>
      <input name="company" placeholder="Company" required/>
      <input name="location" placeholder="Location" required/>
      <button type="submit">Add Job</button>
    </form>`);
});

app.post('/add', (req, res) => {
  jobs.push({ title: req.body.title, company: req.body.company, location: req.body.locati
  res.redirect('/');
});
```

```
app.listen(3000, () => {
  console.log('Job listing site running on http://localhost:3000');
});
```

Run with:

```
node app.js
```

12. Build a Product Inventory System in Node.js + MySQL

Step 1: Setup Project

```
mkdir product-inventory
cd product-inventory
npm init -y
npm install express body-parser mysql2
```

Make sure MySQL is installed on your system and running.

Step 2: Create Database and Table

Log into MySQL and run:

```
CREATE DATABASE inventory_db;
USE inventory_db;
CREATE TABLE products (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255),
  quantity INT,
  price DECIMAL(10,2)
);
```

Step 3: Create `app.js`

```
const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql2');
const app = express();

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

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',       // Use your MySQL username
  password: '',       // Use your MySQL password
  database: 'inventory_db'
});

connection.connect(err => {
```

```
  if (err) throw err;
  console.log('Connected to MySQL database');
});

app.get('/', (req, res) => {
  connection.query('SELECT * FROM products', (err, results) => {
    if (err) throw err;
    let productList = results.map(p =>
      `<li>${p.name} - Qty: ${p.quantity} - $${p.price}</li>`).join('');
    res.send(`<h1>Product Inventory</h1><ul>${productList}</ul>
      <h2>Add Product</h2>
      <form method="post" action="/add">
        <input name="name" placeholder="Name" required/>
        <input name="quantity" placeholder="Quantity" type="number" required/>
        <input name="price" placeholder="Price" type="number" step="0.01" required/>
        <button type="submit">Add Product</button>
      </form>`);
  });
});

app.post('/add', (req, res) => {
  const { name, quantity, price } = req.body;
  const sql = 'INSERT INTO products (name, quantity, price) VALUES (?, ?, ?)';
  connection.query(sql, [name, quantity, price], (err) => {
    if (err) throw err;
    res.redirect('/');
  });
});

app.listen(3000, () => {
  console.log('Inventory system running on http://localhost:3000');
});
```

Run with:

```
node app.js
```

13. Build a Chat Application Using React and Socket.io

Step 1: Setup Backend (Node.js + Socket.io)

Create a folder `chat-app`:

```
mkdir chat-app
cd chat-app
npm init -y
npm install express socket.io
```

Create `server.js`:

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

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

io.on('connection', (socket) => {
  console.log('a user connected');
  socket.on('chat message', msg => {
    io.emit('chat message', msg);
  });
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });
});

server.listen(3000, () => {
  console.log('Chat server running on http://localhost:3000');
});
```

Create `index.html` in the same folder:

```
<!DOCTYPE html>
<html>
  <head><title>Chat</title></head>
  <body>
    <ul id="messages"></ul>
    <form id="form">
      <input id="input" autocomplete="off" /><button>Send</button>
    </form>
    <script src="/socket.io/socket.io.js"></script>
    <script>
      var socket = io();
      var form = document.getElementById('form');
      var input = document.getElementById('input');
      var messages = document.getElementById('messages');

      form.addEventListener('submit', function(e) {
        e.preventDefault();
        if (input.value) {
          socket.emit('chat message', input.value);
          input.value = '';
        }
      });

      socket.on('chat message', function(msg) {
        var item = document.createElement('li');
        item.textContent = msg;
```

```
        messages.appendChild(item);
        window.scrollTo(0, document.body.scrollHeight);
      });
    </script>
  </body>
</html>
```

Run with:

```
node server.js
```

14. Create a Student Result Management App (Basic Node.js)

Step 1: Setup Project

```
mkdir student-results
cd student-results
npm init -y
npm install express body-parser
```

Step 2: Create `app.js`

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

let students = [];

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

app.get('/', (req, res) => {
  let studentList = students.map(s =>
    `<li>${s.name} - Grade: ${s.grade}</li>`).join('');
  res.send(`<h1>Student Results</h1><ul>${studentList}</ul>
    <form method="post" action="/add">
      <input name="name" placeholder="Student Name" required/>
      <input name="grade" placeholder="Grade" required/>
      <button type="submit">Add Result</button>
    </form>`);
});

app.post('/add', (req, res) => {
  students.push({ name: req.body.name, grade: req.body.grade });
  res.redirect('/');
});

app.listen(3000, () => {
  console.log('Student result app running on http://localhost:3000');
});
```

15. Design Node E-learning System (Basic Demo)

Step 1: Setup Project

```
mkdir elearning-system
cd elearning-system
npm init -y
npm install express body-parser
```

Step 2: Create `app.js`

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

let courses = [
  { id: 1, title: 'JavaScript Basics', description: 'Learn the basics of JS' },
  { id: 2, title: 'Node.js Introduction', description: 'Start with Node.js' }
];

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

app.get('/', (req, res) => {
  let courseList = courses.map(c =>
    `<h3>${c.title}</h3><p>${c.description}</p>`).join('');
  res.send(`<h1>E-learning Courses</h1>${courseList}
    <h2>Add Course</h2>
    <form method="post" action="/add">
      <input name="title" placeholder="Course Title" required/>
      <textarea name="description" placeholder="Description" required></textarea>
      <button type="submit">Add Course</button>
    </form>`);
});

app.post('/add', (req, res) => {
  let newId = courses.length + 1;
  courses.push({ id: newId, title: req.body.title, description: req.body.description });
  res.redirect('/');
});

app.listen(3000, () => {
  console.log('E-learning system running on http://localhost:3000');
});
```

You can run each Node.js app by navigating to its folder and running `node app.js` (or `node server.js` if named so). These examples provide simple structures for full beginner understanding.

If you want me to help with any additional features, client-side code, or more advanced versions, just ask!