Lab Practice: Building a Simple Web Server with Node.js and SQLite

Objective:

- Set up a basic Node.js server using Express.js
- Create routes to display data and add new entries
- Connect and query a SQLite database

Part 1: Set Up the Environment

1. Create a project folder:

   ```
   mkdir nodejs-backend-lab

   cd nodejs-backend-lab
   ```

2. Initialize a new Node.js project

   ```
   npm init -y
   ```

3. Install dependencies:

   - Express.js for the server
   - sqlite3 for database access

     ```
     npm install express sqlite3
     ```

Part 2: Set Up the Database

4. Create a script to initialize the database:

   - Create a file named setup_db.js with the following code:

     ```
     const sqlite3 = require('sqlite3').verbose();

     const db = new sqlite3.Database('sample.db');

     db.serialize(() => {
       // Create table
       db.run(`CREATE TABLE IF NOT EXISTS users (
         id INTEGER PRIMARY KEY AUTOINCREMENT,
         name TEXT NOT NULL
       )`);

       // Insert sample data
       db.run(`INSERT INTO users (name) VALUES ('Alice')`);
       db.run(`INSERT INTO users (name) VALUES ('Bob')`);
       db.run(`INSERT INTO users (name) VALUES ('Charlie')`);
     ```

```
  console.log("Database setup complete.");
});

db.close();
```

5. Run the setup script:

```
node setup_db.js
```

- **Confirm the sample.db file is created.**

Part 3: Create the Web Server

6. Create app.js in your project folder:

```
const express = require('express');
const sqlite3 = require('sqlite3').verbose();

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

// Middleware to parse JSON and URL-encoded data
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Function to get a database connection
function getDB() {
  return new sqlite3.Database('sample.db');
}

// Route: Home
app.get('/', (req, res) => {
  res.send('<h1>Welcome to the Node.js Web Server!</h1>');
});

// Route: List users
app.get('/users', (req, res) => {
  const db = getDB();
  db.all(`SELECT id, name FROM users`, [], (err, rows) => {
    if (err) {
      res.status(500).send("Error retrieving users");
      return;
    }
    let html = '<h2>User List</h2><ul>';
    rows.forEach(user => {
      html += `<li>ID: ${user.id}, Name: ${user.name}</li>`;
    });
    html += '</ul>';
    res.send(html);
    db.close();
  });
});

// Route: Add user via query param
app.get('/add_user', (req, res) => {
  const name = req.query.name;
  if (!name) {
    res.status(400).send("Please provide a 'name' query parameter");
    return;
  }
  const db = getDB();
  db.run(`INSERT INTO users (name) VALUES (?)`, [name], function(err) {
    if (err) {
      res.status(500).send("Error adding user");
      return;
    }
    res.send(`User '${name}' added with ID ${this.lastID}`);
    db.close();
  });
});

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

Part 4: Running the Server and Testing

7. Start the server:

```
node app.js
```

8. Test your application:

   - Visit http://localhost:3000/ to see the welcome page.
   - Visit http://localhost:3000/users to see the list of users.
   - Add a new user: http://localhost:3000/add_user?name=David

- Refresh the /users page to see the updated list.