

PROGRAM:

Exercise 6.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Customer CRUD App</title>

  <link rel="stylesheet" href="ex_6_style.css">

</head>

<body>

  <h2>Customer Management System</h2>

  <form id="customerForm">

    <input type="text" id="name" placeholder="Enter Name" required>

    <input type="text" id="city" placeholder="Enter City" required>

    <input type="text" id="mobile" placeholder="Enter Mobile No" required>

    <button type="submit">Add Customer</button>

  </form>

  <table>

    <thead>

      <tr>

        <th>Name</th>

        <th>City</th>

        <th>Mobile No</th>

        <th>Actions</th>

      </tr>

    </thead>

    <tbody id="customersList"></tbody>

  </table>

</body>
```

```

const API_URL = 'http://localhost:5000/customers';

async function fetchCustomers() {
  const res = await fetch(API_URL);
  const customers = await res.json();

  document.getElementById('customersList').innerHTML = customers.map(customer
=> `
    <tr>
      <td><input type="text" value="${customer.name}" id="name-
${customer.id}"></td>
      <td><input type="text" value="${customer.city}" id="city-
${customer.id}"></td>
      <td><input type="text" value="${customer.mobile}" id="mobile-
${customer.id}"></td>
      <td>
        <button onclick="updateCustomer('${customer.id}')">Update</button>
        <button onclick="deleteCustomer('${customer.id}')">Delete</button>
      </td>
    </tr>
  `).join("");
}

async function addCustomer(event) {
  event.preventDefault();

  const name = document.getElementById('name').value;
  const city = document.getElementById('city').value;
  const mobile = document.getElementById('mobile').value;

  await fetch(API_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ name, city, mobile })
  });

  document.getElementById('customerForm').reset();

  fetchCustomers();
}

```

```

    }

    async function updateCustomer(id) {
        const name = document.getElementById(`name-${id}`).value;
        const city = document.getElementById(`city-${id}`).value;
        const mobile = document.getElementById(`mobile-${id}`).value;
        await fetch(`${API_URL}/${id}`, {
            method: 'PUT',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ name, city, mobile })
        });
        fetchCustomers();
    }

    async function deleteCustomer(id) {
        await fetch(`${API_URL}/${id}`, { method: 'DELETE' });
        fetchCustomers();
    }

    document.getElementById('customerForm').addEventListener('submit', addCustomer);
    fetchCustomers();
</script>
</body>
</html>

```

Exercise 6 style.css

```

body {
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #f4f4f4;
}

form {
    margin: 20px auto;
    width: 50%;

```

```
padding: 15px;
background: white;
border-radius: 8px;
}
input {
padding: 10px;
margin: 5px;
width: 80%;
border: 1px solid #ccc;
}
table {
width: 60%;
margin: 20px auto;
border-collapse: collapse;
}
th, td {
padding: 10px;
border: 1px solid #ddd;
text-align: center;
}
button {
padding: 8px 12px;
border: none;
color: white;
border-radius: 5px;
}
button:nth-child(1) { background-color: #28a745; }
button:nth-child(2) { background-color: #dc3545; }
```

Ex_6_server.js

```
require('dotenv').config();
```

```

const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const app = express();
app.use(express.json());
app.use(cors());
const db = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME
});
db.connect(err => {
  if (err) {
    console.error("✗ MySQL Connection Error:", err.message);
    process.exit(1);
  }
  console.log('✓ MySQL Connected');
});
app.post('/customers', (req, res) => {
  const { name, city, mobile } = req.body;
  const sql = 'INSERT INTO customers (name, city, mobile) VALUES (?, ?, ?)';
  db.query(sql, [name, city, mobile], (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json({ id: result.insertId, name, city, mobile });
  });
});
app.get('/customers', (req, res) => {
  const sql = 'SELECT * FROM customers';
  db.query(sql, (err, results) => {

```

```
        if (err) return res.status(500).json({ error: err.message });
        res.json(results);
    });
});

app.put('/customers/:id', (req, res) => {
    const { name, city, mobile } = req.body;
    const sql = 'UPDATE customers SET name = ?, city = ?, mobile = ? WHERE id = ?';
    db.query(sql, [name, city, mobile, req.params.id], (err) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json({ message: 'Customer updated successfully' });
    });
});

app.delete('/customers/:id', (req, res) => {
    const sql = 'DELETE FROM customers WHERE id = ?';
    db.query(sql, [req.params.id], (err) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json({ message: 'Customer deleted successfully' });
    });
});

app.listen(5000, () => console.log('🚀 Server running on port 5000'));
```

Output:

Customer Management System

Enter Name

Enter City

Enter Mobile No

Add Customer

Name	City	Mobile No	Actions
subi	madurai	12345465	<button>Update</button> <button>Delete</button>
sibi	chennai	9876456	<button>Update</button> <button>Delete</button>
sunil	vandavasi	56786865	<button>Update</button> <button>Delete</button>

PROGRAM:

Exercise 7.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta
      name="description"
      content="Web site created using create-react-app"
    />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>
```

App.css

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}
```



```

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```

App.js

```

import React, { useState } from 'react'; // Import useState from React
import logo from './logo.svg';

```

```
import './App.css';

function App() {

  const [count, setCount] = useState(0);

  const handleClick = () => {

    setCount(count + 1); // Increase the count by 1 each time the button is clicked

  };

  return (

    <div className="App">

      <header className="App-header">

        <img src={logo} className="App-logo" alt="logo" />

        <p>

          Edit <code>src/App.js</code> and save to reload.

        </p>

        <a

          className="App-link"

          href="https://reactjs.org"

          target="_blank"

          rel="noopener noreferrer"

        >

          Learn React

        </a>

        { /* Button to trigger count update */}

        <button onClick={handleClick}>Click Me!</button>

        { /* Display the click count */}

        <p>Click count: {count}</p>

      </header>

    </div>

  );

}

export default App;
```

Output:

PROGRAM:

Exercise 8:

App.js

```
import React, { useState } from 'react';
import './App.css';
function App() {
  const [todos, setTodos] = useState([]);
  const [input, setInput] = useState("");
  const [editIndex, setEditIndex] = useState(null);
  const [editText, setEditText] = useState("");
  const handleInputChange = (e) => {
    setInput(e.target.value);
  };
  const addTodo = () => {
    if (input.trim() !== "") {
      setTodos([...todos, { text: input, completed: false }]);
      setInput("");
    }
  };
  const toggleComplete = (index) => {
    const updatedTodos = todos.map((todo, i) =>
      i === index ? { ...todo, completed: !todo.completed } : todo
    );
    setTodos(updatedTodos);
  };
  const deleteTodo = (index) => {
    const updatedTodos = todos.filter((_, i) => i !== index);
    setTodos(updatedTodos);
  };
  const editTodo = (index) => {
```

```

setEditIndex(index);

  setEditText(todos[index].text);
};

const saveEdit = () => {
  if (editText.trim() !== "") {
    const updatedTodos = todos.map((todo, i) =>
      i === editIndex ? { ...todo, text: editText } : todo
    );
    setTodos(updatedTodos);
    setEditIndex(null);
    setEditText("");
  }
};

return (
  <div className="App">
    <h1>To-Do App</h1>
    <div className="todo-input">
      <input
        type="text"
        value={input}
        onChange={handleInputChange}
        placeholder="Enter a new task"
      />
      <button className="add-btn" onClick={addTodo}>Add</button>
    </div>
    { /* Edit todo */ }
    { editIndex !== null && (
      <div className="edit-todo">
        <input
          type="text"

```

```

    value={editText}
    onChange={(e) => setEditText(e.target.value)}
    placeholder="Edit your task"
  />
  <button className="save-btn" onClick={saveEdit}>Save</button>
</div>
)}
<table className="todo-table">
  <thead>
    <tr>
      <th>Task</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    {todos.map((todo, index) => (
      <tr key={index}>
        <td className={todo.completed ? 'completed' : ''}>
          <span onClick={() => toggleComplete(index)}>{todo.text}</span>
        </td>
        <td>
          <button className="edit-btn" onClick={() => editTodo(index)}>Edit</button>
          <button className="delete-btn" onClick={() =>
deleteTodo(index)}>Delete</button>
        </td>
      </tr>
    ))}
  </tbody>
</table>
</div>
);

```

```
}  
export default App;
```

App.css

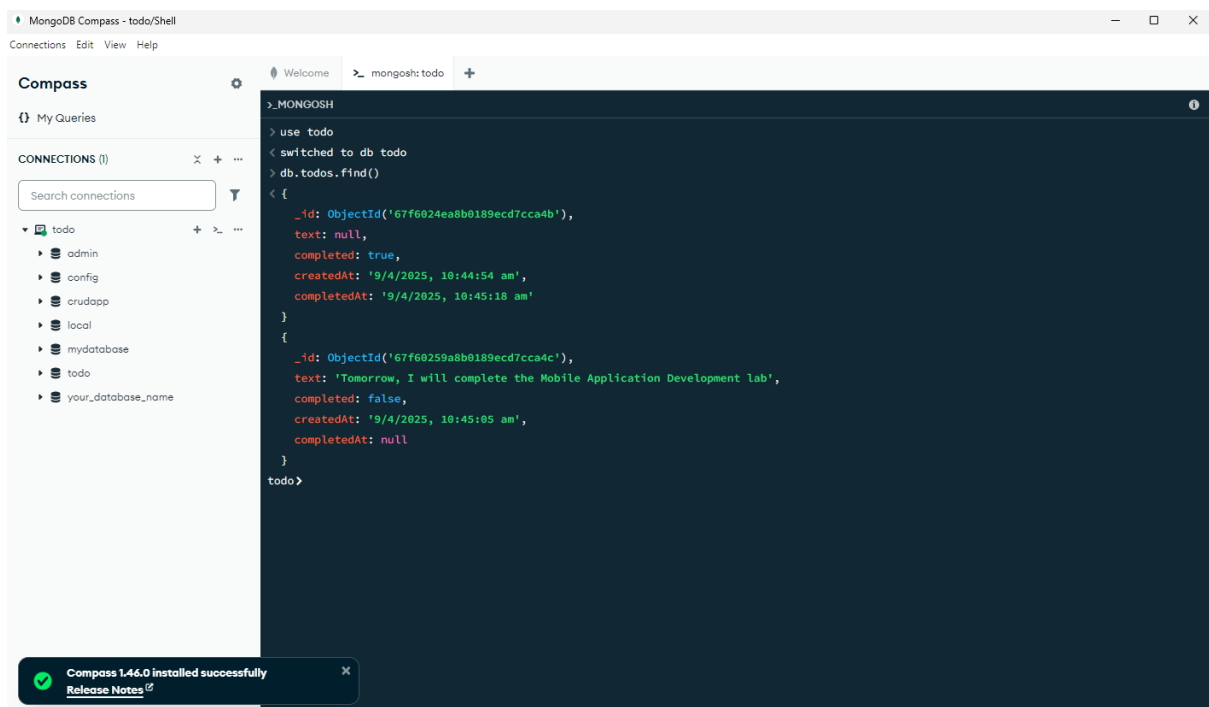
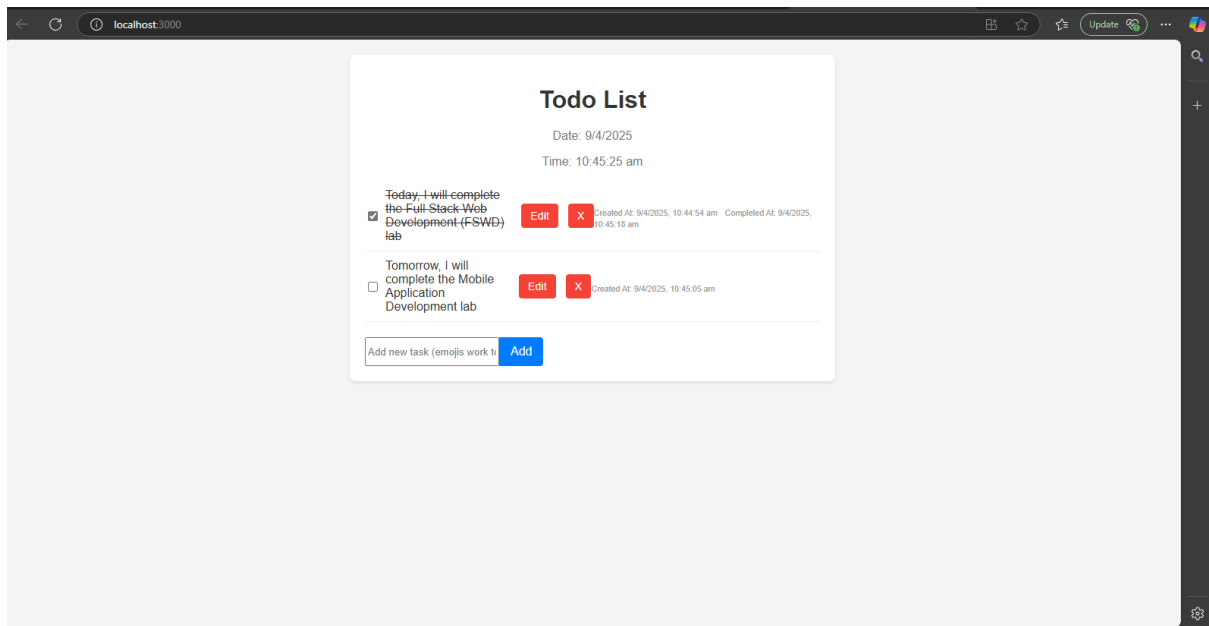
```
.App {  
  text-align: center;  
  margin-top: 20px;  
  font-family: Arial, sans-serif;  
}  
.todo-input {  
  margin-bottom: 20px;  
}  
input {  
  padding: 10px;  
  font-size: 16px;  
  width: 250px;  
}  
button {  
  padding: 10px;  
  font-size: 16px;  
  cursor: pointer;  
  margin-left: 10px;  
  border: none;  
  border-radius: 4px;  
}  
.add-btn {  
  background-color: green;  
  color: white;  
}  
.add-btn:hover {
```

```
    background-color: darkgreen;
}
.delete-btn {
    background-color: red;
    color: white;
}
.delete-btn:hover {
    background-color: darkred;
}
.edit-btn {
    background-color: orange;
    color: white;
}
.edit-btn:hover {
    background-color: darkorange;
}
.todo-table {
    width: 80%;
    margin: 0 auto;
    border-collapse: collapse;
}
th, td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ddd;
}
th {
    background-color: #f2f2f2;
}
.completed {
```



```
text-decoration: line-through;
color: gray;
}
.edit-todo {
  margin-top: 20px;
}
.edit-todo input {
  padding: 10px;
  font-size: 16px;
  width: 250px;
}
.save-btn {
  background-color: blue;
  color: white;
  cursor: pointer;
  padding: 10px;
  margin-left: 10px;
}
.save-btn:hover {
  background-color: darkblue;
}
```

Output:



PROGRAM

Exercise 9

models/user.js

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

const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

module.exports = mongoose.model('User', UserSchema);
```

routes/user.js

```
import express from 'express';
import { getAllUsers, login, logout, signUp } from '../controllers/user.js';
import { checkRole, checkToken } from '../middlewares/middlewares.js';
const router = express.Router();

router.post("/signUp", signUp);
router.post("/login", login);
router.post("/logout", checkToken, logout);
router.get('/getAllUsers', checkToken, checkRole(['admin', 'manager']), getAllUsers);

export default router;
```

Server.js

```
const express = require('express');
const connectDB = require('./config/db');
const cookieParser = require('cookie-parser');
const authRoutes = require('./routes/auth');
require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 2222;

app.use(express.json());
app.use(cookieParser());

app.use('/api/auth', authRoutes);

connectDB();
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

controller/user.js

```

import bcrypt from 'bcrypt';
import User from "../models/user.js";
import { CreateToken } from '../middlewares/middlewares.js';
import jwt from 'jsonwebtoken';

export const signUp = async (req, res) => {

  const { name, mobile, email, password, role } = req.body;
  if (!name || !mobile || !email || !password) {
    return res.status(422).json({ message: "All feilds should be filled" });
  }
  try {
    let existingUser;
    try {
      existingUser = await User.findOne({ $or: [{ email: email }, { mobile:
mobile }] });
    } catch (err) {
      console.error(err);
    }

    if (existingUser) {
      if (existingUser.email == email) {
        return res.status(409).json({ message: "A User is already signUp with
this email" });
      }
      else if (existingUser.mobile == mobile) {
        return res.status(409).json({ message: "A User is already signUp with
this mobile" });
      }
    }

    const salt = await bcrypt.genSalt(6);
    const hashedpassword = await bcrypt.hash(password, salt);

    const user = new User({
      name,
      mobile,
      email,
      password: hashedpassword,
      role: role,
    });

    await user.save();
    return res.status(201).json({ message: "Account Creation is success, Login
to your account", User: user });
  }
}

```

```

    } catch (err) {
      console.error(err)
      return res.status(400).json({ message: "Error in saving user in DB" });
    }
  }

export const login = async (req, res) => {

  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(422).json({ message: "All feilds should be filled" })
  }

  let loggedUser;

  try {
    loggedUser = await User.findOne({ email: email });

    if (!loggedUser) {
      return res.status(404).json({ message: "Email is not found, Check it and try again" })
    }
    const isPasswordCorrect = bcrypt.compareSync(password, loggedUser.password);
    if (!isPasswordCorrect) {
      return res.status(400).json({ message: "Invalid password, Check it and try again" })
    }
    const token = CreateToken(loggedUser._id);
    res.cookie(String(loggedUser._id), token, {
      path: "/",
      expires: new Date(Date.now() + 1000 * 59),
      httpOnly: true,      sameSite: "lax"
    })

    return res.status(200).json({ message: "Successfully logged in", User: loggedUser })
  } catch (err) {
    console.log(err)
  }
}

export const logout = (req, res) => {
  const cookies = req.headers.cookie
  const previousToken = cookies.split("=")[1];

```

```

    if (!previousToken) {
      return res.status(400).json({ message: "Couldn't find token" });
    }
    jwtwebtoken.verify(String(previousToken), process.env.JWTAUTHSECRET, (err,
user) => {
      if (err) {
        console.log(err);
        return res.status(403).json({ message: "Authentication failed" });      }
      res.clearCookie(`${user.id}`);
      req.cookies[`${user.id}`] = "";
      return res.status(200).json({ message: "Successfully Logged Out" });
    });
  });
};

export const getAllUsers = async (req, res) => {
  try {
    const allusers = await User.find();
    if (!allusers) {
      return res.status(404).json({ message: "There are not any users" });
    }
    else {
      res.status(200).json({ allusers })
    }
  } catch (error) {
    console.log(error);
    return res.status(500).json({ message: "Error in getting the Users" })
  }
}

```

Output

The screenshot shows the Postman application interface. The left sidebar displays a history of requests, with the current request being a POST to `http://localhost:2222/auth/signup`. The main panel shows the request details, including the URL, method (POST), and headers. The body is a JSON object with the following fields: `name` (Sibikarthik), `mobile` (7894941230), `email` (sibi@gmail.com), and `password` (12345). The response is displayed in the bottom panel, showing a status of 201 Created, a time of 17 ms, and a size of 516 B. The response body is a JSON object with the following fields: `message` (Account Creation is success. Login to your account), `User` (Sibikarthik), `mobile` (7894941230), `email` (sibi@gmail.com), `password` (\$2b\$06\$Mu8O5gRuUHipkxKD6i2ZiOvrSusX2jh72Zct9ZjHiV33xaMwFCZVC), and `role` (admin).

The screenshot shows the Postman application interface. The left sidebar displays a history of requests, with the current request being a POST to `http://localhost:2222/auth/login`. The main panel shows the request details, including the URL, method (POST), and headers. The body is a JSON object with the following fields: `email` (sibi@gmail.com) and `password` (12345). The response is displayed in the bottom panel, showing a status of 200 OK, a time of 20 ms, and a size of 763 B. The response body is a JSON object with the following fields: `message` (Successfully logged in), `User` (Sibikarthik), `mobile` (7894941230), `email` (sibi@gmail.com), `password` (\$2b\$06\$Mu8O5gRuUHipkxKD6i2ZiOvrSusX2jh72Zct9ZjHiV33xaMwFCZVC), and `role` (admin).

PROGRAM

Exercise 10

Step 1: Prepare your computer for Virtualization:

- **Enable Processor Virtualization:** Ensure Virtualization is enabled on your computer. See the Virtualization Error (VT-d/VT-x or AMD-V) for troubleshooting support.
- **Review File Sync Services** for tools like OneDrive, Nextcloud, DropBox Sync, iCloud, etc. If you are using a data synchronization service, make sure it DOES NOT (or at least not frequently) synchronize the folder in which your hypervisor imports and installs the Virtual Machines.
- **File sync services** can cause a dramatic fall-off in performance for your entire system as these services try to synchronize these massive files that are getting updated constantly while you are using the Virtual Machines.
- **Sufficient Disk Space:** Virtual Machines require a significant amount of Disk space (10 GB or more each is typical). Ensure you have sufficient space on your computer.
- **Admin Privileges:** Installing a hypervisor on a host in most cases requires admin privileges.

Step 2: Install Hypervisor (Virtualization Tool): Installing a hypervisor on your host is usually quite simple. In most cases, the install program will ask only a couple of questions, such as where to install the hypervisor software.

Step 3: Import a Virtual Machine:

- The first step is to download the Virtual Machine for your course from our Course Virtual Machines page. This will download an .ova file. The .ova file is actually a compressed (zipped) tarball of a Virtual Machine exported from Virtual Box.
- Once the Virtual Machine has been imported, it will normally show up in the guest list within your hypervisor tool.

Step 4: Start the Virtual Machine: To start up a Virtual Machine guest in most hypervisors, you simply click on the desired guest and click the Start button (often double-clicking the guest icon will work as well).

Step 5: Using the Virtual Machine: MC4266-FSWD LAB 34

- **Sharing files between the guest and host:** To learn about different ways of sharing files, check out this guide.
- **Run a command with sudo (root) privileges:** Open a terminal and type any command with sudo in front to run that command as root.
- **Example:** `sudo apt-get install vim` – will install the vim text editor package on an Ubuntu Linux Virtual Machine.
- **Find the IP address of your guest:** Open a terminal and type `ifconfig | more` – The `| more` (pronounced “pipe more”) will “pipe” the output of the `ifconfig` command to the `more` command, which will show the results one page at a time, so it doesn’t scroll by before you see it all.

- If you have a Host-Only Network IP address, you will see an IP of 192.168.56.101 (or something similar). Check the Trouble-Shooting section below for more information about the Host-Only Network.

Step 6: Shut down the Virtual Machine: When you are done using a guest Virtual Machine, regardless of hypervisor, you need to shut it down properly. This can be done in three ways:

1. Press the shutdown button found on the desktop, taskbar, or task menu of the guest operating system.
2. Open a terminal and type the command: `sudo shutdown -h now`
3. In the guest window, click Machine (menu) -> ACPI Shut down – This will simulate the power button being pressed

PROGRAM

Exercise 11

Server.js

```
const http = require('http'); // Import the 'http' module for creating a server.

const hostname = '0.0.0.0'; // Set the hostname to listen on all IP addresses.

const port = 8080; // Define the port for the server to listen on.

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

  if (req.method === 'GET' && req.url === '/ping') { // Check if the request method is GET
    and the URL is '/ping'

    res.statusCode = 200; // Set status code to 200 (OK)

    res.setHeader('Content-Type', 'application/json'); // Set response type as JSON

    res.end(JSON.stringify({ message: 'pong' })); // Send a 'pong' response
  } else {

    res.statusCode = 404; // For all other routes or methods, return a 404

    res.end('Not Found');

  }

});

server.listen(port, hostname, () => { // Start the server on the specified hostname and port.

  console.log(`Server running at http://${hostname}:${port}`);

});
```

Dockerfile

```
# Use official Node.js image from Docker Hub

FROM node:16

# Set the working directory inside the container

WORKDIR /usr/src/app

# (Optional) Copy package.json and package-lock.json if dependencies are needed

# COPY package*.json ./
```

RUN npm install

Copy the server.js file to the working directory inside the container
COPY server.js .

Expose port 8080 for the app to be accessible outside the container
EXPOSE 8080

Run the Node.js server when the container starts
CMD ["node", "server.js"]

Output

Containers

Images

Volumes

Builds



Docker Hub


Docker Scout

Extensions

Containers / ping-server

ping-server

<  ce5b5e6beec3  [nodejs-ping-server:latest](#)

[8080:8080](#) 

Logs


Inspect

Bind mounts

Exec

Files

Stats

Server running at <http://0.0.0.0:8080/> 

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[+] Building 17.5s (8/8) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 294B                               0.0s
=> [internal] load metadata for docker.io/library/node:16         2.8s
=> [internal] load .dockerignore                                  0.1s
=> => transferring context: 2B                                     0.0s
=> [1/3] FROM docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e424 13.0s
=> => resolve docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246 0.1s
=> => sha256:ee7d78be1eb92caf6ae84fc3af736b23eca018d5dedc967ae5bdee6d7082483b 450B / 450B 0.3s
=> => sha256:ca266fd6192108b67fb57b74753a8c4ca5d8bd458baae3d4d4f7ce9f42dedcc1d 2.27MB / 2.27MB 0.9s
=> => sha256:0e421f66aff42bb069dffc26af6d132194b22a1082b08c5ef7cd69c627783c04 34.79MB / 34.79MB 3.2s
=> => sha256:a3b95bbaa61ce24cefd89e7c74d6fbd7713b2bcae93af47063d06bd7e02172 4.20kB / 4.20kB 0.7s
=> => sha256:513d779256048c961239af5f500589330546b072775217272e19ffae1635e98e 191.90MB / 191.90MB 7.4s
=> => sha256:ffd9397e94b74abcb54e514f1430e00f04328d1f895eadbd482f08cc02444e5 51.89MB / 51.89MB 3.6s
=> => sha256:7e9bf114588c05b2df612b083b96582f3b8dbf51647aa6138a50d09d42df2454 17.58MB / 17.58MB 1.5s
=> => sha256:311da6c465ea1576925360eba391bcd32dece9ba95960a0bc9ffcb25fe712017 50.50MB / 50.50MB 2.8s
=> => extracting sha256:311da6c465ea1576925360eba391bcd32dece9ba95960a0bc9ffcb25fe712017 0.9s
=> => extracting sha256:7e9bf114588c05b2df612b083b96582f3b8dbf51647aa6138a50d09d42df2454 0.3s
=> => extracting sha256:ffd9397e94b74abcb54e514f1430e00f04328d1f895eadbd482f08cc02444e5 1.0s
=> => extracting sha256:513d779256048c961239af5f500589330546b072775217272e19ffae1635e98e 2.3s
=> => extracting sha256:a3b95bbaa61ce24cefd89e7c74d6fbd7713b2bcae93af47063d06bd7e02172 0.0s
=> => extracting sha256:0e421f66aff42bb069dffc26af6d132194b22a1082b08c5ef7cd69c627783c04 2.0s
=> => extracting sha256:ca266fd6192108b67fb57b74753a8c4ca5d8bd458baae3d4d4f7ce9f42dedcc1d 0.3s
=> => extracting sha256:ee7d78be1eb92caf6ae84fc3af736b23eca018d5dedc967ae5bdee6d7082483b 0.2s
=> [internal] load build context                                0.1s
=> => transferring context: 545B                                0.0s
=> [2/3] WORKDIR /usr/src/app                                0.7s
=> [3/3] COPY server.js .                                    0.1s
=> => exporting layers                                           0.5s
=> => exporting manifest sha256:f5023de4a6907d23a1c272a6fa1124ca36b01c52f8c6b10820d162d382f2829c 0.0s
=> => exporting config sha256:3140a400f6464cf9d4308d8734a1bda0f7d1a0080c7e5f9a8feb371cc321f0ae 0.0s => => exporting attestation manifest sha256:24ea65fcb15402bad73924c9bbc3ca0bbf637a72e075042017a93972 0.0s => => exporting manifest list sha256:eba9071afeb0da23471e0a6f1f04cde32dfb4e10c54fd684c736add50dfac41 0.0s => => n
aming to docker.io/library/nodejs-ping-server:latest          0.0s
=> => unpacking to docker.io/library/nodejs-ping-server:latest 0.1s
PS D:\65_FSMO_LAB\nodejs-ping-server> docker run -d -p 8080:8080 --name ping-server nodejs-ping-server
>>
ce5b5e6beec340226e04ca9ebf1f2cb48f8c9485ac927596e86de1f2020b9698
PS D:\65_FSMO_LAB\nodejs-ping-server> curl http://localhost:8080/ping
>>
```

142224621057

