PRACTICE 3.3

Student Name: Niharika Singh **UID: 23BCS11624**
Branch: BE-CSE **Date of upload: 25 SEP 2025**
Semester: 5th **Subject Name: Fullstack**

AIM:

Aim 1: CLI Employee Management System Using Node.js and Arrays

Aim 2: REST API for Playing Card Collection Using Express.js

Aim 3: Concurrent Ticket Booking System with Seat Locking and Confirmation

CODE 1 :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Employee Management System</title>
</head>
<body>
  <h1>Employee Management System</h1>

  <div>
    <input id="empName" placeholder="Employee Name">
    <input id="empId" placeholder="Employee ID">
    <button onclick="addEmployee()">Add</button>
  </div>

  <h2>Employees</h2>
  <ul id="employeeList"></ul>

  <div>
    <input id="removeId" placeholder="Employee ID to remove">
```

```
.
  <button onclick="removeEmployee()">Remove</button>
 </div>

 <script>
  async function fetchEmployees() {
   const res = await fetch('/api/employees');
   const data = await res.json();
   const list = document.getElementById("employeeList");
   list.innerHTML = "";
   data.forEach(emp => {
    const li = document.createElement("li");
    li.textContent = `Name: ${emp.name}, ID: ${emp.id}`;
    list.appendChild(li);
   });
  }

  async function addEmployee() {
   const name = document.getElementById("empName").value;
   const id = document.getElementById("empId").value;
   await fetch('/api/employees', {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name, id })
   });
   fetchEmployees();
  }

  async function removeEmployee() {
   const id = document.getElementById("removeId").value;
   await fetch(`/api/employees/${id}`, { method: "DELETE" });
   fetchEmployees();
  }

  // Load employees on start
  fetchEmployees();
 </script>
</body>
</html>

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

app.use(express.json()); app.use(express.static(__dirname)); // serve index.html, styles, etc.

// In-memory employee list let employees = [ { name: "Alice", id: "E101" }, { name: "Bob", id:
"E102" }, { name: "Charlie", id: "E103" } ];

// Routes app.get('/api/employees', (req, res) => { res.json(employees); });

app.post('/api/employees', (req, res) => { const { name, id } = req.body; if (!name || !id) { return
res.status(400).json({ error: "Name and ID required" }); } employees.push({ name, id });
res.json({ message: Employee ${name} (ID: ${id}) added successfully. }); });

app.delete('/api/employees/:id', (req, res) => { const { id } = req.params; const index =
employees.findIndex(emp => emp.id === id); if (index === -1) { return res.status(404).json({
error: "Employee not found" }); } const removed = employees.splice(index, 1); res.json({
message: Employee ${removed[0].name} (ID: ${removed[0].id}) removed successfully. }); });
```

.

// Start server const PORT = 3000; app.listen(PORT, () => { console.log(Server running at http://localhost:${PORT}); });

Employee Management System

Employee Name | Employee ID | Add

Employees

- Name: Alice, ID: E101
- Name: Bob, ID: E102
- Name: Charlie, ID: E103

Employee ID to remove | Remove

CODE 2:

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

app.use(express.json()); app.use(express.static(__dirname)); // serve index.html, styles, etc.

// In-memory employee list let employees = [{ name: "Alice", id: "E101" }, { name: "Bob", id: "E102" }, { name: "Charlie", id: "E103" }];

// Routes app.get('/api/employees', (req, res) => { res.json(employees); });

app.post('/api/employees', (req, res) => { const { name, id } = req.body; if (!name || !id) { return res.status(400).json({ error: "Name and ID required" }); } employees.push({ name, id }); res.json({ message: Employee ${name} (ID: ${id}) added successfully. }); });

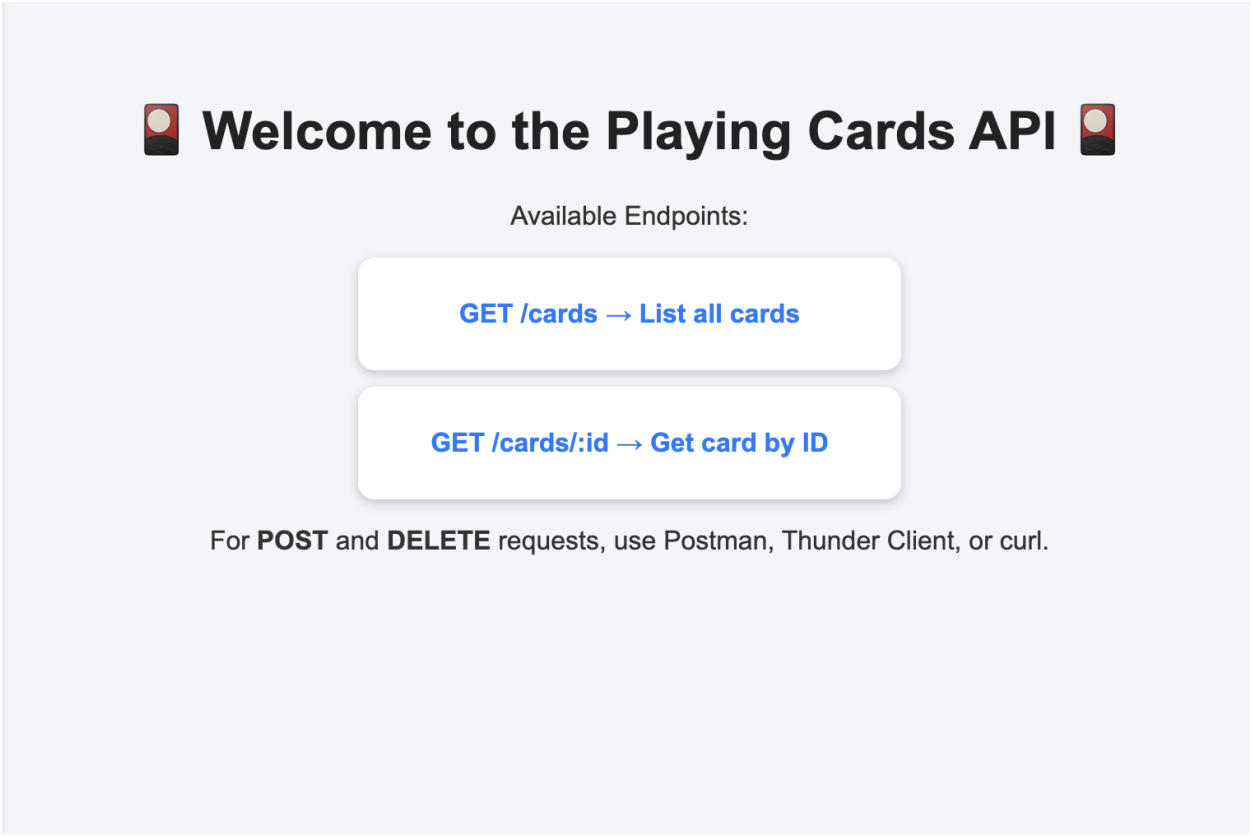
app.delete('/api/employees/:id', (req, res) => { const { id } = req.params; const index = employees.findIndex(emp => emp.id === id); if (index === -1) { return res.status(404).json({ error: "Employee not found" }); } const removed = employees.splice(index, 1); res.json({ message: Employee ${removed[0].name} (ID: ${removed[0].id}) removed successfully. }); });

// Start server const PORT = 3000; app.listen(PORT, () => { console.log(Server running at http://localhost:${PORT}); });

```
.
const express = require("express");
const app = express();
app.use(express.json());

// In-memory cards data
let cards = [
  { id: 1, suit: "Hearts", value: "Ace" },
  { id: 2, suit: "Spades", value: "King" },
  { id: 3, suit: "Diamonds", value: "Queen" },
];

// Root route with styled homepage
app.get("/", (req, res) => {
  res.send(`
    <html>
      <head>
        <title>Playing Cards API</title>
        <style>
          body {
            font-family: Arial, sans-serif;
            background: #f4f4f9;
            color: #333;
            text-align: center;
            padding: 50px;
          }
          h1 {
            color: #222;
          }
          .card {
            background: white;
            padding: 15px;
            margin: 10px auto;
            width: 300px;
            border-radius: 10px;
            box-shadow: 0 2px 6px rgba(0,0,0,0.2);
          }
          a {
            display: block;
            margin: 10px;
            color: #007BFF;
            text-decoration: none;
            font-weight: bold;
          }
          a:hover {
            text-decoration: underline;
          }
        </style>
      </head>
      <body>
        <h1> 🎴 Welcome to the Playing Cards API 🎴 </h1>
        <p>Available Endpoints:</p>
        <div class="card"><a href="/cards">GET /cards → List all cards</a></div>
        <div class="card"><a href="/cards/1">GET /cards/:id → Get card by ID</a></div>
        <p>For <b>POST</b> and <b>DELETE</b> requests, use Postman, Thunder Client, or
curl.</p>
      </body>
```

```
          .
    </html>
  `);
});

// GET all cards
app.get("/cards", (req, res) => {
  res.json(cards);
});

// GET card by ID
app.get("/cards/:id", (req, res) => {
  const card = cards.find(c => c.id === parseInt(req.params.id));
  if (!card) {
    return res.status(404).json({ error: "Card not found" });
  }
  res.json(card);
});

// POST new card
app.post("/cards", (req, res) => {
  const newCard = {
    id: cards.length + 1,
    suit: req.body.suit,
    value: req.body.value
  };
  cards.push(newCard);
  res.status(201).json(newCard);
});

// DELETE a card
app.delete("/cards/:id", (req, res) => {
  const cardIndex = cards.findIndex(c => c.id === parseInt(req.params.id));
  if (cardIndex === -1) {
    return res.status(404).json({ error: "Card not found" });
  }
  const removedCard = cards.splice(cardIndex, 1);
  res.json({
    message: `Card with ID ${req.params.id} removed`,
    card: removedCard[0]
  });
});

// Start server
app.listen(3000, () => {
  console.log("Server running on http://localhost:3000");
});
```

.



CODE 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Seat Booking System 🎟️</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      background: #f4f4f9;
      padding: 20px;
    }
    h1 {
      color: #333;
    }
    #seat-container {
      display: grid;
      grid-template-columns: repeat(5, 60px);
      gap: 10px;
      justify-content: center;
      margin-top: 20px;
    }
    .seat {
      width: 60px;
```

```
        .
    height: 60px;
    line-height: 60px;
    border-radius: 8px;
    font-weight: bold;
    cursor: pointer;
    border: 2px solid #555;
    transition: all 0.3s ease;
   }
   .available { background: #4caf50; color: white; }
   .locked { background: #ff9800; color: white; }
   .booked { background: #f44336; color: white; cursor: not-allowed; }
   button {
    margin-top: 15px;
    padding: 10px 20px;
    border: none;
    background: #333;
    color: white;
    border-radius: 5px;
    cursor: pointer;
   }
   button:hover {
    background: #555;
   }
  </style>
</head>
<body>
  <h1>🎟️ Seat Booking System</h1>
  <p>Click a seat to lock → confirm within <b>1 minute</b> or it resets.</p>

  <div id="seat-container"></div>
  <p id="message"></p>

  <script>
   const seatContainer = document.getElementById("seat-container");
   const message = document.getElementById("message");

   // Fetch seats from backend
   async function loadSeats() {
    const res = await fetch("/seats");
    const seats = await res.json();

    seatContainer.innerHTML = "";
    for (let id in seats) {
     const seat = document.createElement("div");
     seat.className = "seat " + seats[id].status;
     seat.textContent = id;
     seat.onclick = () => handleSeatClick(id, seats[id].status);
     seatContainer.appendChild(seat);
    }
   }

   // Handle seat actions
   async function handleSeatClick(id, status) {
    if (status === "available") {
     const res = await fetch(`/lock/${id}`, { method: "POST" });
     const data = await res.json();
```

```
        .
        message.textContent = data.message;
      } else if (status === "locked") {
        const res = await fetch(`/confirm/${id}`, { method: "POST" });
        const data = await res.json();
        message.textContent = data.message;
      } else {
        message.textContent = `Seat ${id} is already booked.`;
      }
      loadSeats(); // refresh seats
    }

    // Refresh seats every 2s
    setInterval(loadSeats, 2000);
    loadSeats();
  </script>
</body>
</html>

const express = require("express"); const path = require("path");

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

app.use(express.json());

// ------------------ In-Memory Seats ------------------ let seats = {}; for (let i = 1; i <= 10; i++) {
seats[i] = { status: "available", lockTimer: null }; }

// ------------------ Routes ------------------

// Root page → serve index.html app.get("/", (req, res) => { res.sendFile(path.join(__dirname,
"index.html")); });

// Get all seats app.get("/seats", (req, res) => { const seatData = {}; for (let id in seats) {
seatData[id] = { status: seats[id].status }; } res.json(seatData); });

// Lock a seat app.post("/lock/:id", (req, res) => { const id = req.params.id; if (!seats[id]) { return
res.status(404).json({ message: "Seat not found" }); }

if (seats[id].status === "available") { seats[id].status = "locked";

// Auto unlock after 1 minute if not confirmed
seats[id].lockTimer = setTimeout(() => {
  if (seats[id].status === "locked") {
    seats[id].status = "available";
    seats[id].lockTimer = null;
  }
}, 60000);

return res.json({ message: `Seat ${id} locked successfully. Confirm within 1 minute.` });


} else { return res.status(400).json({ message: Seat ${id} is not available for locking. }); } });

// Confirm a booking app.post("/confirm/:id", (req, res) => { const id = req.params.id; if
(!seats[id]) { return res.status(404).json({ message: "Seat not found" }); }

if (seats[id].status === "locked") { seats[id].status = "booked";
```

.

```
// Clear lock timer
if (seats[id].lockTimer) {
  clearTimeout(seats[id].lockTimer);
  seats[id].lockTimer = null;
}

return res.json({ message: `Seat ${id} booked successfully!` });


} else { return res.status(400).json({ message: Seat is not locked and cannot be booked }); } });

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