**Step 1: Set Up MongoDB Atlas**

1. **Create a MongoDB Atlas account:**
   * If you don't already have an account, visit [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/register) and sign up.
2. **Create a Cluster:**
   * Once logged in, create a new cluster. MongoDB Atlas will provide instructions for setting up your cluster.
   * Choose the free-tier (M0 Sandbox) cluster for learning purposes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Create a Database User:**
   * In the "Database Access" section of the Atlas dashboard, create a new database user with a username and password. Make sure to save the credentials securely.

A screenshot of a computer

Description automatically generated

1. **Allow IP Access:**
   * Go to "Network Access" in your Atlas dashboard and click "Add IP Address."
   * Choose "Allow access from anywhere" (0.0.0.0/0) if you're working locally, or add your specific IP address.

A screenshot of a computer

Description automatically generated

1. **Get the Connection String:**
   * In your cluster dashboard, click "Connect" and choose "Connect your application."
   * Copy the provided connection string, which will look something like this:

A screenshot of a computer

Description automatically generated

Replace <username>, <password>, and myFirstDatabase with your actual database name and credentials.

**Step 2: Node.js Application**

Now, update your Node.js application to connect to MongoDB Atlas.

1. Install the required MongoDB package:

A screenshot of a computer

Description automatically generated

1. Code

const express = require("express");

const { MongoClient } = require("mongodb");

const app = express();

const PORT = 3000;

let db = null;

let client = null;

const connectDB = async () => {

  try {

    client = new MongoClient(

      "mongodb+srv://cayyanraj:2305@full-stack-lab.ptod1.mongodb.net/?retryWrites=true&w=majority&appName=full-stack-lab"

    );

    await client.connect();

    db = client.db("employees"); // Make sure the 'employees' database exists

    console.log("Connected with Atlas");

  } catch (err) {

    console.error(err);

    process.exit(1);

  }

};

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

  res.send(`

    <h1>Employee Management</h1>

    <button onclick="window.location.href='/employees'">Get Employees</button>

    <button onclick="window.location.href='/add-employee'">Add Employee</button>

  `);

});

app.get("/employees", async (req, res) => {

  try {

    const employeesCollection = db.collection("employees");

    const employees = await employeesCollection.find({}).toArray();

    const employeesCount = await employeesCollection.countDocuments();

    if (employeesCount === 0) {

      res.send(`<h2> No employees found`);

    } else {

      res.send(employees);

    }

  } catch (err) {

    res.status(500).send("Error retrieving employees");

  }

});

app.get("/add-employee", (req, res) => {

  res.send(`

    <h2>Add New Employee</h2>

    <form action="/add-employee" method="POST">

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

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

      <label for="position">Position:</label><br>

      <input type="text" id="position" name="position" required><br><br>

      <input type="submit" value="Add Employee">

    </form>

  `);

});

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

app.post("/add-employee", async (req, res) => {

  try {

    const { name, position } = req.body;

    const employeesCollection = db.collection("employees");

    await employeesCollection.insertOne({ name, position });

    res.send("Employee added successfully! <br> <a href='/'>Go back</a>");

  } catch (err) {

    res.status(500).send("Error adding employee");

  }

});

app.listen(PORT, async () => {

  await connectDB();

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

});

**Explanation:**

1. **Importing Modules:**

* **express**: Used to create a web server and handle routes.
* **mongodb**: The MongoDB client used to interact with the MongoDB database.

const express = require("express");

const { MongoClient } = require("mongodb");

1. **App and Server Setup:**

* **app**: The instance of the Express app.
* **PORT**: The port where the server will listen (in this case, 3000).
* **db and client**: Variables to store the MongoDB connection and database instance.

const app = express();

const PORT = 3000;

let db = null;

let client = null;

1. **Connecting to MongoDB:**

* **connectDB()**: This function connects to MongoDB using the connection string (replace credentials accordingly). It connects to the MongoDB Atlas cluster and selects the **employees** database.
* If the connection is successful, it logs "Connected with Atlas" to the console.
* If there's an error, it logs the error and exits the process.

const connectDB = async () => {

try {

client = new MongoClient(

"mongodb+srv://cayyanraj:2305@full-stack-lab.ptod1.mongodb.net/?retryWrites=true&w=majority&appName=full-stack-lab"

);

await client.connect();

db = client.db("employees");

console.log("Connected with Atlas");

} catch (err) {

console.error(err);

process.exit(1);

}

};

1. **Home Route ("/"):**

* This route serves as the homepage and displays two buttons:
  1. **"Get Employees"**: Navigates to the /employees route to view the list of employees.
  2. **"Add Employee"**: Navigates to the /add-employee route to add a new employee.

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

res.send(`

<h1>Employee Management</h1>

<button onclick="window.location.href='/employees'">Get Employees</button>

<button onclick="window.location.href='/add-employee'">Add Employee</button>

`);

});

1. **Route to Get Employees ("/employees"):**

* This route retrieves and displays all employee records from the **employees** collection in MongoDB.
* It checks if there are any employees in the collection, and if there are none, it shows "No employees found." If employees exist, it returns the list in JSON format.

app.get("/employees", async (req, res) => {

try {

const employeesCollection = db.collection("employees");

const employees = await employeesCollection.find({}).toArray();

const employeesCount = await employeesCollection.countDocuments();

if (employeesCount === 0) {

res.send(`<h2> No employees found`);

} else {

res.send(employees);

}

} catch (err) {

res.status(500).send("Error retrieving employees");

}

});

1. **Route to Add Employee ("/add-employee"):**

* This route serves an HTML form where the user can input a new employee's name and position.
* The form submits the data to the same route using the **POST** method.

app.get("/add-employee", (req, res) => {

res.send(`

<h2>Add New Employee</h2>

<form action="/add-employee" method="POST">

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

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

<label for="position">Position:</label><br>

<input type="text" id="position" name="position" required><br><br>

<input type="submit" value="Add Employee">

</form>

`);

});

1. **Middleware to Parse Form Data:**

* The express.urlencoded middleware is used to parse the URL-encoded data (form submissions). It allows the server to handle the form data sent via the POST request.

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

1. **Handling POST Request to Add an Employee ("/add-employee"):**

* This route handles the form submission for adding a new employee.
* The employee's name and position are extracted from the form data (req.body).
* A new employee record is inserted into the **employees** collection in MongoDB.
* On success, it sends a confirmation message, otherwise, it returns an error.

app.post("/add-employee", async (req, res) => {

try {

const { name, position } = req.body;

const employeesCollection = db.collection("employees");

await employeesCollection.insertOne({ name, position });

res.send("Employee added successfully! <br> <a href='/'>Go back</a>");

} catch (err) {

res.status(500).send("Error adding employee");

}

});

1. **Starting the Server:**

* The server starts by calling the **connectDB()** function to establish the database connection.
* Once the connection is successful, the server listens on the specified port (3000).

app.listen(PORT, async () => {

await connectDB();

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

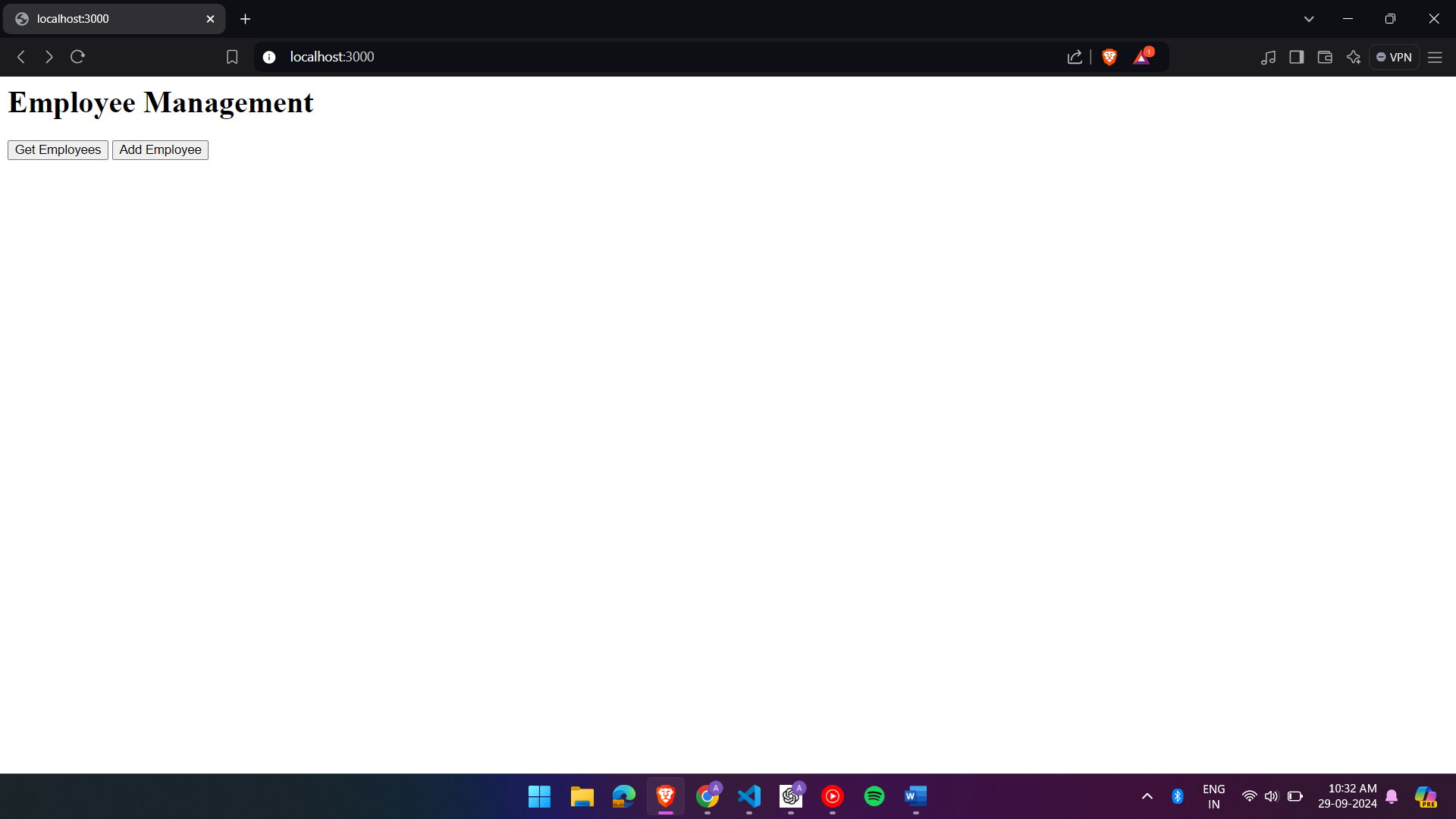
});

**Step 3: Run Your Application**

1. Make sure MongoDB Atlas is set up correctly, and your IP address is whitelisted in MongoDB Atlas.
2. Start your Node.js server by running:

Your app will now connect to MongoDB Atlas instead of a local MongoDB instance. You can visit http://localhost:3000 to test it, and use Postman or curl to interact with the /add-employee and /employees endpoints.

**Output:**

****

**Add Employee Example:**

**A computer screen with a white screen

Description automatically generated**

**A computer screen with a white screen

Description automatically generated**

**Get Employee:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**