

---

---

# MERN Stack

# CRUD Operations

---

---

---

**M**

MongoDB - Document Database

**E**

ExpressJS – Server Side Web FW, Runs in NodeJS, has URL Routing & Handles HttpRequests and Responses

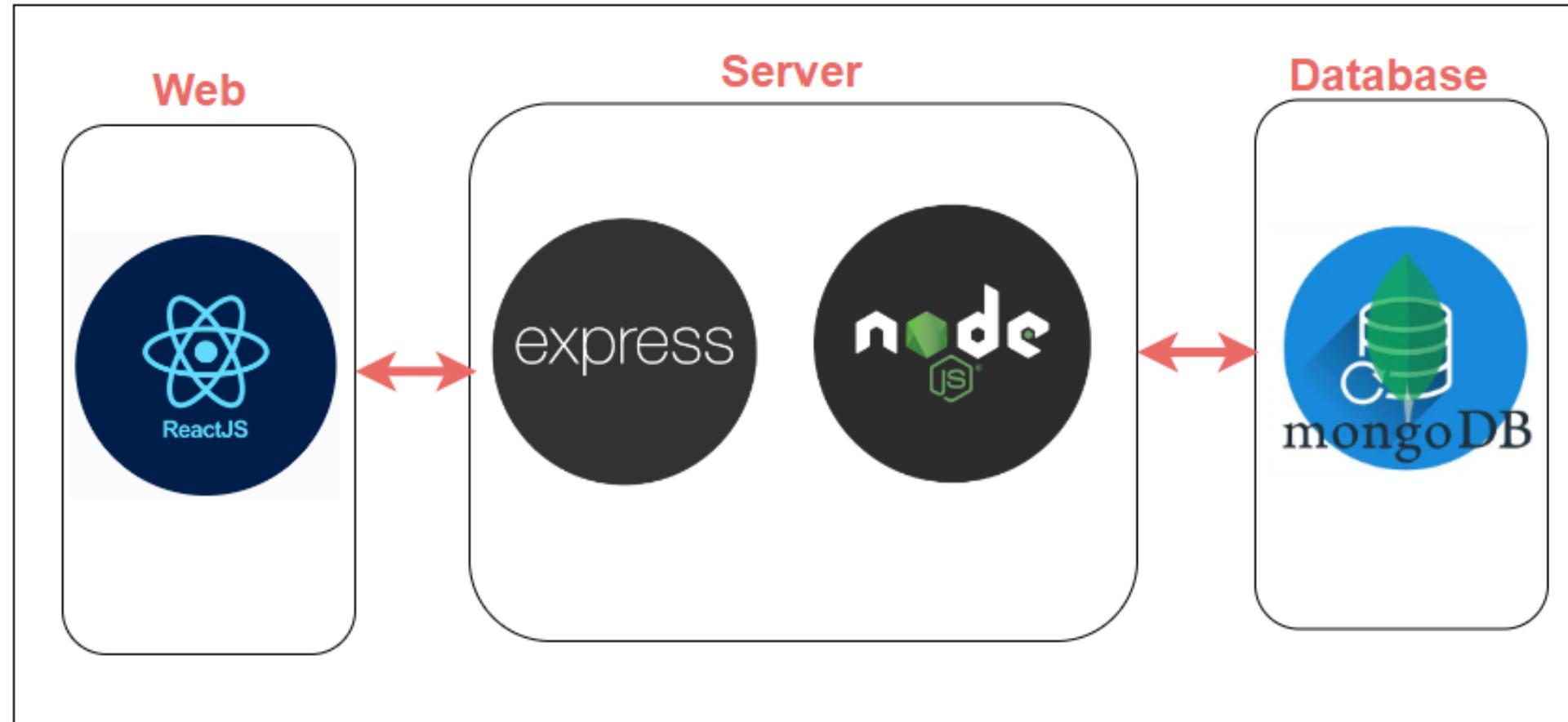
**R**

ReactJS – Client side JavaScript Framework

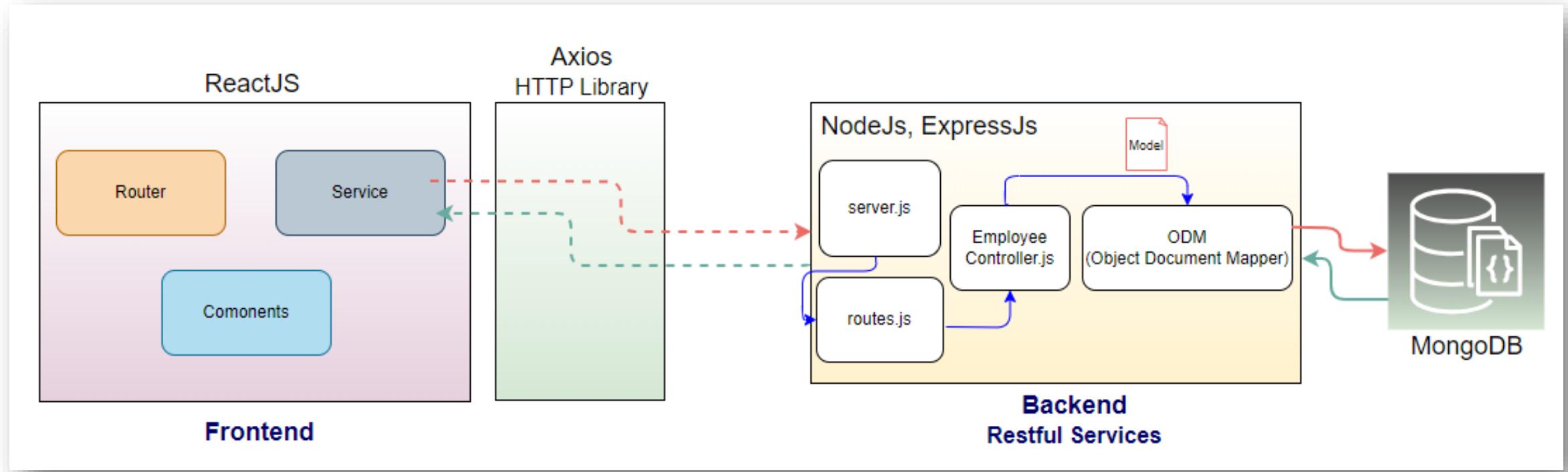
**N**

NodeJS – JavaScript Web Server

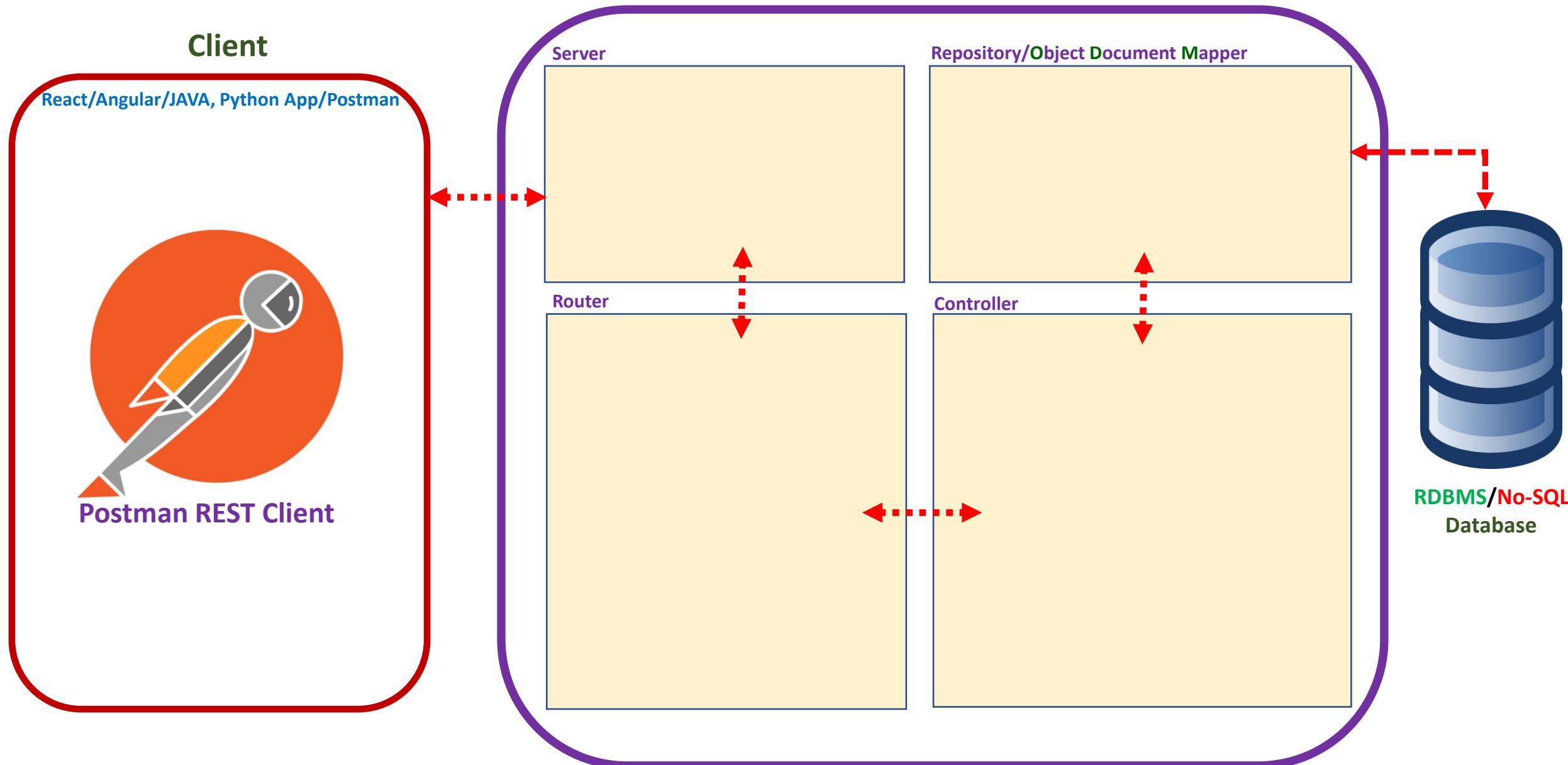
# How MERN Stack Works



# Architecture - Building Restful Services Using NodeJS, ExpressJS



## Backend



# **STEPS – Implementing REST API in NodeJS, ExpressJS**

---

I. Project setup - NodeJS & ExpressJS

2. Design the Contract:API End-Points, Request, Response, Error Codes

3. API Implementation

- Create server.js By Configuring all routes with project level path
- Create routes.js with Routers (end-point & Handler Method Mapping)
- Create Controller with Handler methods which Maps to End-Points
- Create a Model Class with Schema
- Create a db.js file with Database configuration & Return Database Connection

4. Test API Using any REST Clients - Postman

# Designing the Contract

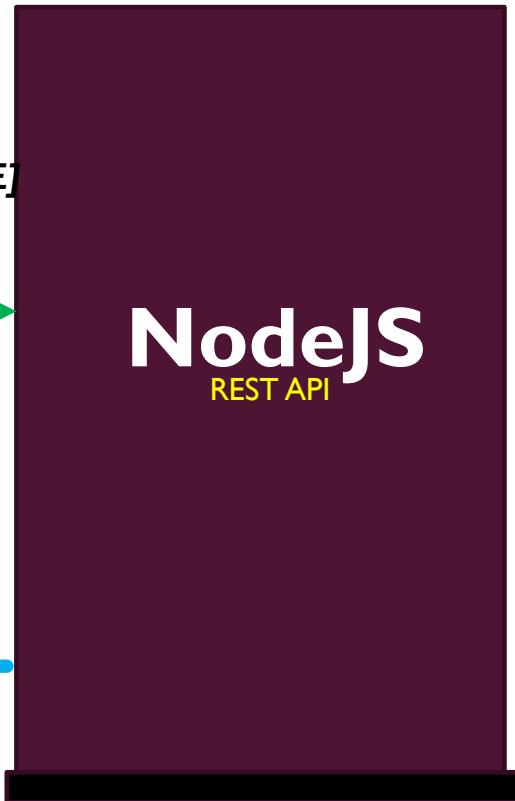
## End-Points

<b>Create</b> - <a href="http://127.0.0.1:4000/api/employee">http://127.0.0.1:4000/api/employee</a>	[HTTP POST]
<b>Get All</b> - <a href="http://127.0.0.1:4000/api/employee">http://127.0.0.1:4000/api/employee</a>	[HTTP GET]
<b>Get</b> - <a href="http://127.0.0.1:4000/api/employee/1001">http://127.0.0.1:4000/api/employee/1001</a>	[HTTP GET]
<b>Update</b> - <a href="http://127.0.0.1:4000/api/employee/1001">http://127.0.0.1:4000/api/employee/1001</a>	[HTTP PUT]
<b>Delete</b> - <a href="http://127.0.0.1:4000/api/employee/1001">http://127.0.0.1:4000/api/employee/1001</a>	[HTTP DELETE]



**Body: JSON/XML**

```
{  
  "empid": 1001,  
  "name": "Ali",  
  "age": 21,  
  "salary": 21000.0,  
  "address": "Bengaluru"  
}
```



**Response Body: JSON/XML**

**Status: OK / NOT FOUND**

```
{  
  "empid": 1001,  
  "name": "Ali",  
  "age": 21,  
  "salary": 21000,  
  "address": "Bengaluru"  
}
```

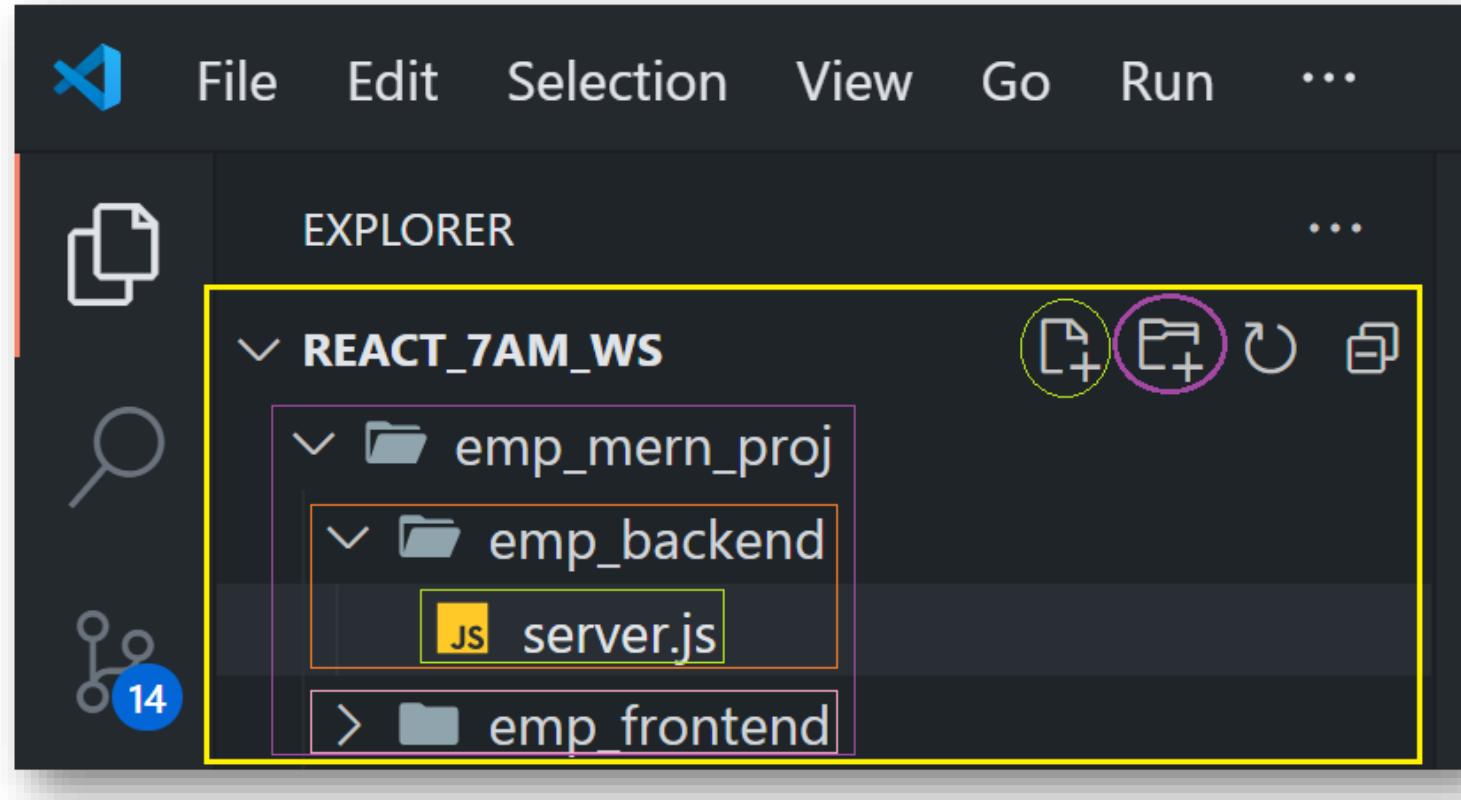
---

---

# Backend Project Setup



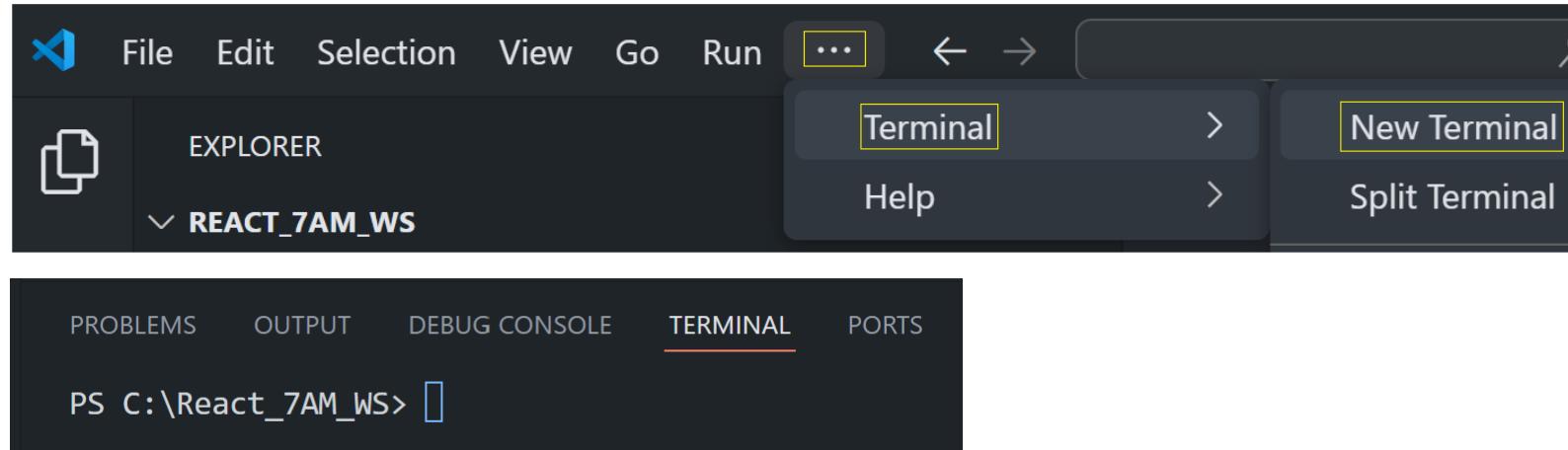
# Project folders Creation



1. Create **emp\_mern\_proj** Folder inside Workspace folder
2. Create **emp\_frontend** folder inside **emp\_mern\_proj** folder
3. Create **emp\_backend** folder inside **emp\_mern\_proj** folder
4. Create **server.js** file inside **emp\_backend** folder

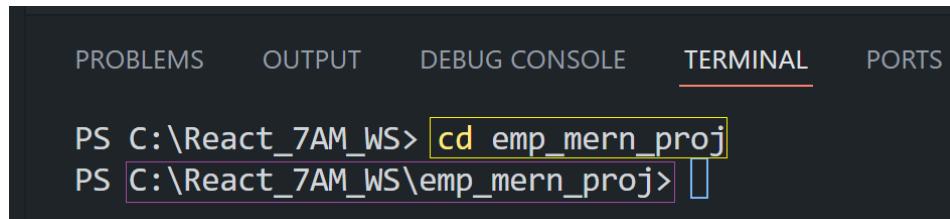
# npm init command - Initialize backed service

- Go to the terminal



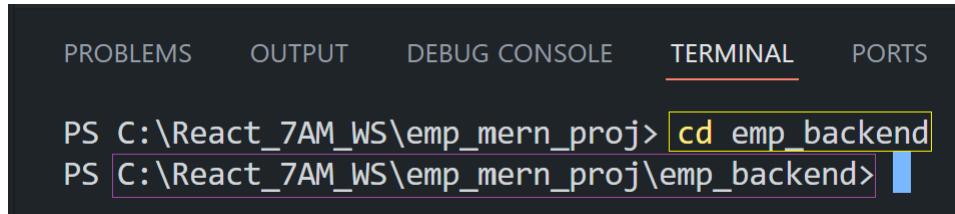
The screenshot shows the VS Code interface with the 'Terminal' tab selected in the bottom navigation bar. The terminal window displays a PowerShell prompt: 'PS C:\React\_7AM\_WS>'. The top menu bar shows 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', '...', 'Terminal', 'New Terminal', 'Help', and 'Split Terminal'. The left sidebar shows 'EXPLORER' and a folder named 'REACT\_7AM\_WS'.

- From workspace folder, Go to project folder using **cd emp\_mern\_proj**



The screenshot shows the VS Code interface with the 'Terminal' tab selected. The terminal window displays the command 'cd emp\_mern\_proj' being typed, followed by the path 'C:\React\_7AM\_WS\emp\_mern\_proj'. The top navigation bar shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (selected), and 'PORTS'.

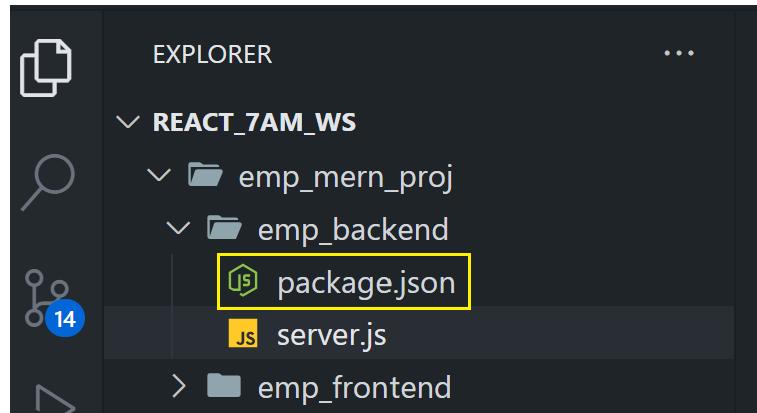
- From emp\_mern\_proj folder, Go to backend folder using **cd emp\_backend**



The screenshot shows the VS Code interface with the 'Terminal' tab selected. The terminal window displays the command 'cd emp\_backend' being typed, followed by the path 'C:\React\_7AM\_WS\emp\_mern\_proj\emp\_backend'. The top navigation bar shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (selected), and 'PORTS'.

# npm init command - Initialize backed service ...

- From terminal, run npm init command from **emp\_backend** folder



- package.json** new file will be created in **emp\_backend** folder

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (emp_backend) emp_backend_service
version: (1.0.0)
description:
entry point: (server.js)
test command:
git repository:
keywords:
author:
license: (ISC)
Press Enter
About to write to C:\React_7AM_WS\emp_mern_proj\emp_backend\package.json:
{
  "name": "emp_backend_service",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) Press Enter
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

# nodemon setup - Automatic Server Restart

- Install nodemon

A screenshot of a terminal window titled "TERMINAL". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal shows the command "npm install -g nodemon" being run in a Windows environment (PS prompt). The output indicates 29 packages were added in 2 seconds, and 4 packages are looking for funding. A yellow box highlights the command "npm install -g nodemon".

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm install -g nodemon
added 29 packages in 2s
4 packages are looking for funding
  run `npm fund` for details
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

- Checking nodemon Version

A screenshot of a terminal window showing the command "nodemon -v" being run. The output shows the version 3.1.1. A yellow box highlights the command "nodemon -v".

```
C:\Users\shany>nodemon -v
3.1.1

C:\Users\shany>
```

# nodemon setup - Automatic Server Restart...

- Go to package.json
- Update scripts “**start**” & add “**dev**”

The screenshot shows the Visual Studio Code interface with the following details:

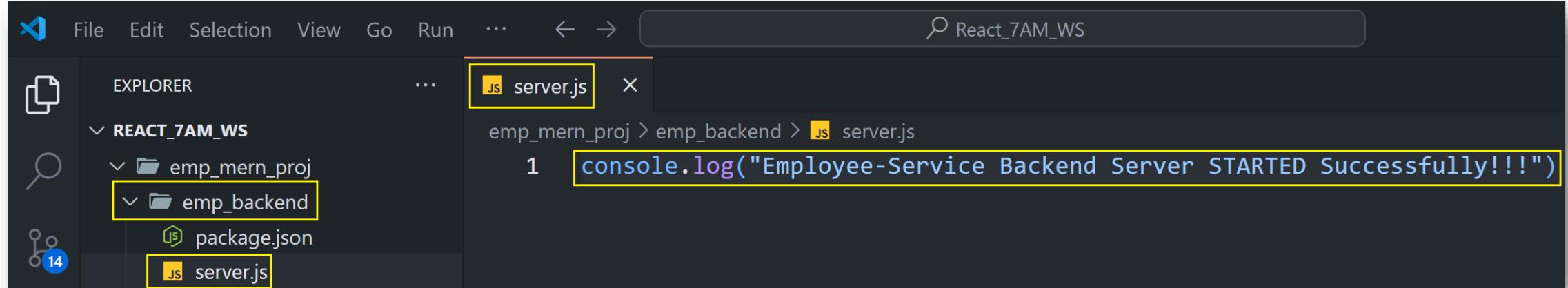
- File Explorer (Left):** Shows a project structure under "REACT\_7AM\_WS". It includes "emp\_mern\_proj", "emp\_backend" (which contains "package.json" and "server.js"), and "emp\_frontend". A notification badge "14" is visible on the folder icon.
- Search Bar (Top):** Contains the text "React\_7AM\_WS".
- Code Editor (Right):** Displays the contents of "package.json". The file path is shown as "emp\_mern\_proj > emp\_backend > package.json". The code is as follows:

```
1 {  
2   "name": "emp_backend_service",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "server.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",  
8     "start": "nodemon server.js",  
9     "dev": "nodemon server.js"  
10    },  
11    "author": "",  
12    "license": "ISC"  
13  }  
14
```

The "start" and "dev" scripts in the "scripts" object are highlighted with orange boxes.

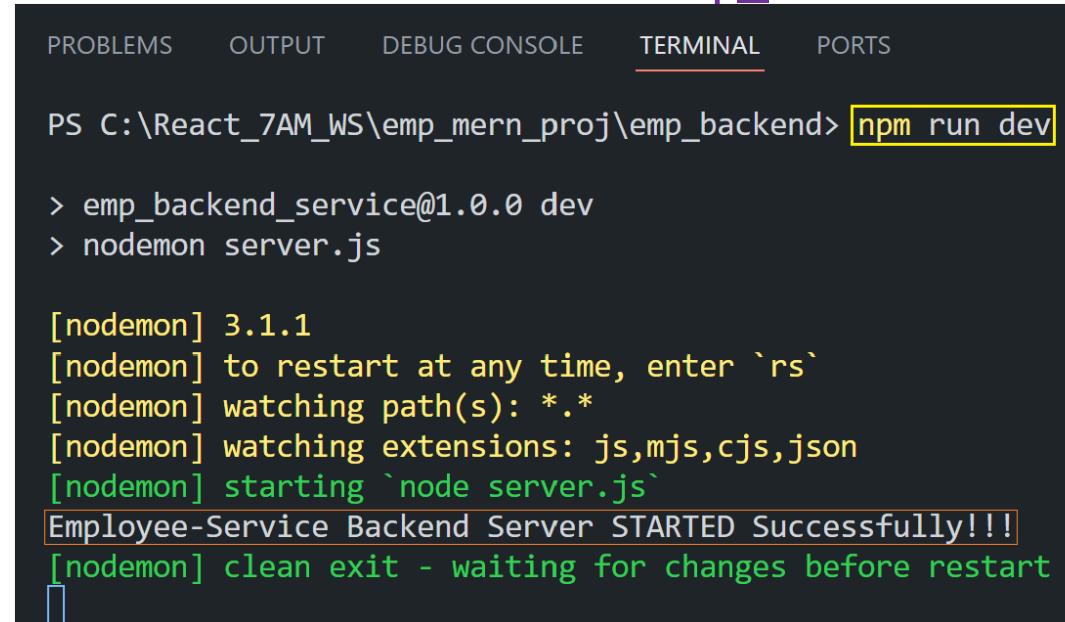
# Testing Backend Server

- Add Log statement in emp\_backend/server.js



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure under 'REACT\_7AM\_WS': 'emp\_mern\_proj' > 'emp\_backend' > 'server.js'. The 'server.js' file is open in the main editor area, showing the code: `console.log("Employee-Service Backend Server STARTED Successfully!!!")`. The code is highlighted with a yellow box.

- Start the backed server from emp\_backend folder using **npm run dev** command



The screenshot shows the VS Code terminal tab. The command `npm run dev` is entered and executed. The output shows the process starting with 'dev' script, using 'nodemon' to monitor 'server.js', and finally outputting the log message: 'Employee-Service Backend Server STARTED Successfully!!!'. The command and the log message are highlighted with a yellow box.

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm run dev

> emp_backend_service@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Employee-Service Backend Server STARTED Successfully!!!
[nodemon] clean exit - waiting for changes before restart
```

# Install required software

## ▪ Install express

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm i express
added 64 packages, and audited 65 packages in 3s
12 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

## ▪ Install cors – for connecting Frontend, Backend

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm i cors
added 2 packages, and audited 87 packages in 925ms
13 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

## ▪ Install mongoose

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm i mongoose
added 20 packages, and audited 85 packages in 3s
13 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

## ▪ Install dotenv – for reading files

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\React_7AM_WS\emp_mern_proj\emp_backend> npm i dotenv
added 1 package, and audited 88 packages in 799ms
14 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_backend>
```

# Changes in package.json & server.js

- Add “type”: “module” field in **package.json** for using import statements in other files

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows a project structure under **REACT\_7AM\_WS**. It includes **emp\_mern\_proj**, **emp\_backend** (which contains **node\_modules**, **package-lock.json**, and **package.json**), and **emp\_frontend**.
- package.json** file content (highlighted with a yellow box):

```
1  {
2    "name": "emp_backend_service",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "type": "module",
```

A tooltip for the **"type": "module"** line is displayed: "Add this line Used for using import in other files".

- Do the following changes in **server.js**

The screenshot shows the VS Code interface with the following details:

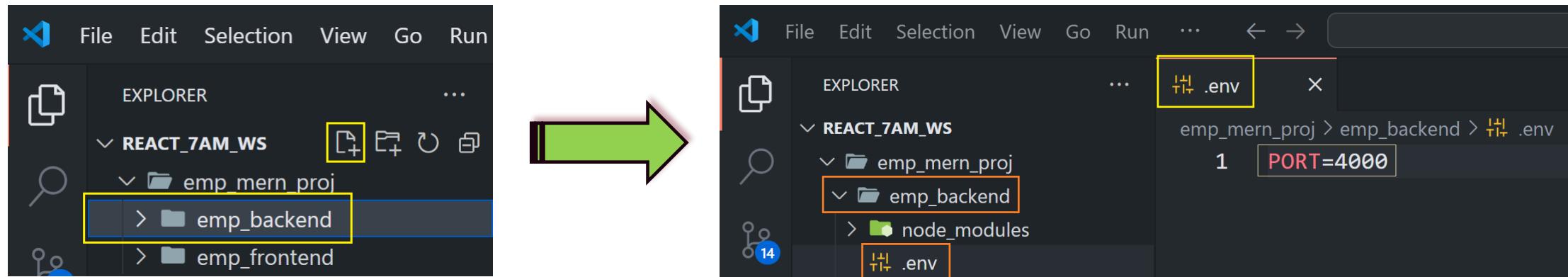
- EXPLORER** view: Shows the same project structure as the previous screenshot.
- server.js** file content (highlighted with a yellow box):

```
1  import express from 'express'
2
3  const app = express() // NOTE: Assigning all features of express to app variable
4
5  app.listen(4000, () => {
6    console.log("Employee-Service Backend Server STARTED Successfully!!!")
7 })
```

A note in the code editor states: **NOTE: Assigning all features of express to app variable**.

# Create .env file in emp\_backend folder AND define PORT Number

- Defining PORT Value in .env file



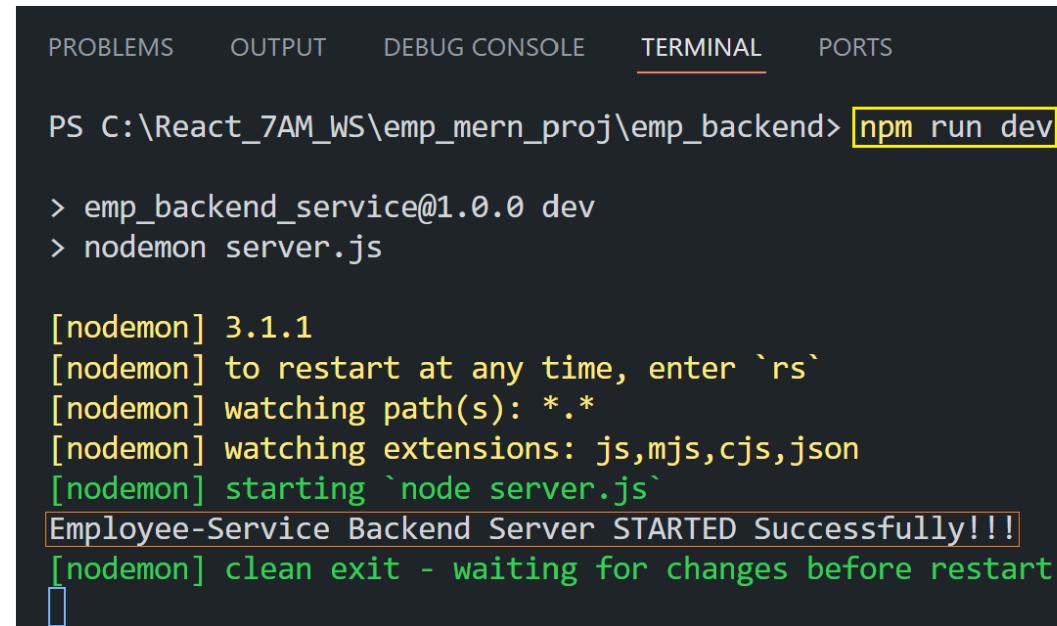
- Reading PORT Value from .env file in server.js

The image shows the VS Code interface with the server.js file open. The code reads the PORT value from the environment variable:

```
import express from 'express'
const app = express()
app.listen(process.env.PORT, () => {
  console.log("Employee-Service Backend Server STARTED Successfully!!!")})
```

# Start Backend Server

- Start the backed server from emp\_backend folder using **npm run dev** command



The screenshot shows a terminal window with the following interface elements:

- Top navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.
- Command line input: PS C:\React\_7AM\_WS\emp\_mern\_proj\emp\_backend> [highlighted]
- Output log:
  - > emp\_backend\_service@1.0.0 dev
  - > nodemon server.js
  - [nodemon] 3.1.1
  - [nodemon] to restart at any time, enter `rs`
  - [nodemon] watching path(s): `.*`
  - [nodemon] watching extensions: js,mjs,cjs,json
  - [nodemon] starting `node server.js`
  - [highlighted] Employee-Service Backend Server STARTED Successfully!!!
  - [nodemon] clean exit - waiting for changes before restart

---

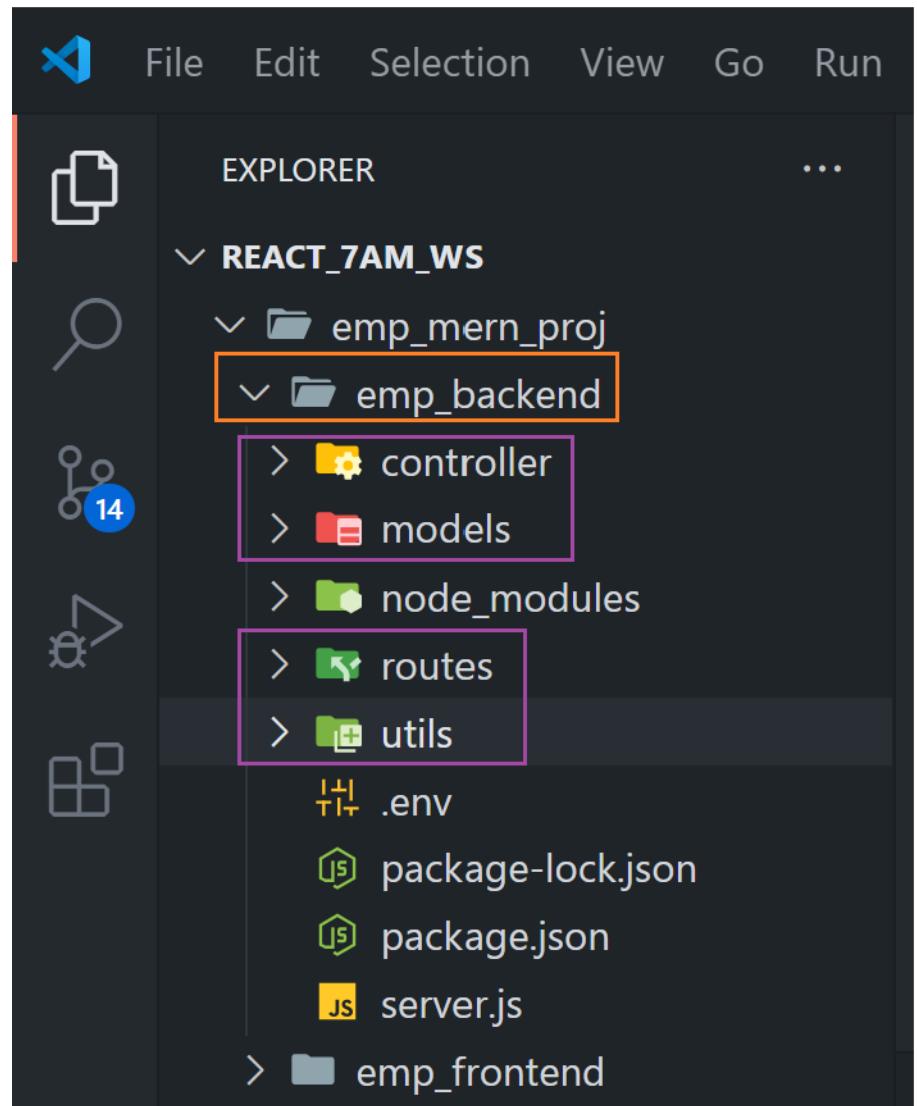
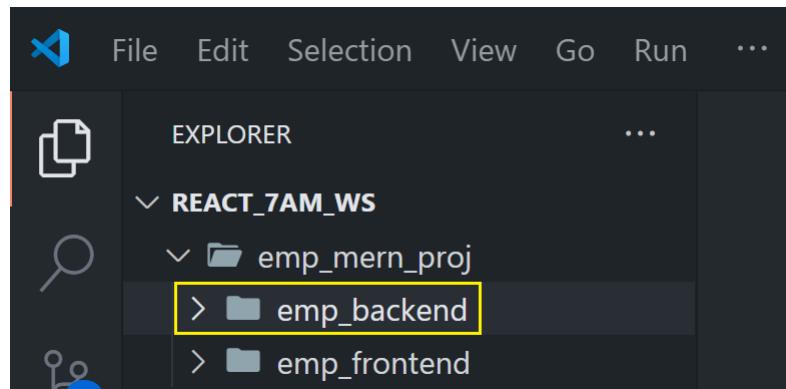
---

---

# Backend

# Package Creation

# Create routes, models, controller, utils folders inside emp\_backend folder



---

# **Configure/Setup**

# **MongoDB Database**

# Start MongoDB Server

```
C:\Users\shannv>mongod
{"t": {"$date": "2024-05-25T22:14:31.502+05:30"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "thread1", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "outgoing": {"minWireVersion": 6, "maxWireVersion": 21}, "isInternalClient": true}}}
{"t": {"$date": "2024-05-25T22:14:32.876+05:30"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "thread1", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2024-05-25T22:14:32.880+05:30"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "thread1", "msg": "Implicit TCP FastOpen in use."}
{"t": {"$date": "2024-05-25T22:14:32.915+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "namespace": "config.tenantMigrationDonors"}}
{"t": {"$date": "2024-05-25T22:14:32.915+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "namespace": "config.tenantMigrationRecipients"}}
{"t": {"$date": "2024-05-25T22:14:32.916+05:30"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "thread1", "msg": "Multi threading initialized"}
{"t": {"$date": "2024-05-25T22:14:32.916+05:30"}, "s": "I", "c": "TENANT_M", "id": 7091600, "ctx": "thread1", "msg": "Starting TenantMigrationAccessBlockerRegistry"}
{"t": {"$date": "2024-05-25T22:14:32.917+05:30"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 31548, "port": 27017, "dbPath": "C:/data/db/", "architecture": "64-bit", "host": "LAPTOP-17C2SB1B"}}
{"t": {"$date": "2024-05-25T22:14:32.917+05:30"}, "s": "I", "c": "CONTROL", "id": 23398, "ctx": "initandlisten", "msg": "Target operating system minimum version", "attr": {"targetMinOS": "Windows 7/Windows Server 2008 R2"}}
{"t": {"$date": "2024-05-25T22:14:32.918+05:30"}, "s": "I", "c": "CONTROL", "id": 23403, "ctx": "initandlisten", "msg": "Build Info", "attr": {"buildInfo": {"version": "7.0.9", "gitVersion": "3ff3a3925c36ed277cf5eafca5495f2e3728dd67", "modules": [], "allocator": "tcmalloc", "environment": {"distmod": "windows", "distarch": "x86_64", "target_arch": "x86_64"}}}}
{"t": {"$date": "2024-05-25T22:14:34.450+05:30"}, "s": "I", "c": "STORAGE", "id": 22262, "ctx": "initandlisten", "msg": "Timestamp monitor starting"}
{"t": {"$date": "2024-05-25T22:14:34.454+05:30"}, "s": "I", "c": "NETWORK", "id": 23015, "ctx": "listener", "msg": "Listening on", "attr": {"address": "127.0.0.1"}}
{"t": {"$date": "2024-05-25T22:14:34.454+05:30"}, "s": "I", "c": "NETWORK", "id": 23016, "ctx": "listener", "msg": "Waiting for connections", "attr": {"port": 27017, "ssl": "off"}}
{"t": {"$date": "2024-05-25T22:14:34.454+05:30"}, "s": "I", "c": "CONTROL", "id": 8423403, "ctx": "initandlisten", "msg": "mongod startup complete", "attr": {"Summary of time elapsed": {"Startup from clean shutdown?": true, "Statistics": {"Transport layer setup": "0 ms", "Run initial syncer crash recovery": "0 ms", "Create storage engine lock file in the data directory": "0 ms", "Get metadata describing storage engine": "0 ms", "Validate options in metadata against configuration": "0 ms"}}}
```

# MongoDB Connection String

DB\_RUL = **mongodb://127.0.0.1:27017/astidb**

```
PS C:\Users\shanz> mongosh
Current Mongosh Log ID: 665219c848e302bdbf46b798
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB:      7.0.9
Using Mongosh:      2.2.5
mongosh 2.2.6 is available for download: https://www.mongodb.com/try/download/shell
For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2024-05-18T12:05:26.386+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test>
```

# Configure Database connection

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the project structure. The db.js file is selected in the Explorer. The main editor area shows the code for db.js:

```
1 import mongoose from "mongoose"
2
3 const db_connection = async() => {
4     try{
5         await mongoose.connect(process.env.DB_URL)
6
7         console.log("MongoDB Connected Successfully!!!")
8
9     } catch(error) {
10        console.log(error)
11    }
12
13
14 export default db_connection
```

- ✓ Changes in .env
- ✓ Create new db.js
- ✓ Changes in server.js
- ✓ Start the server

The terminal tab in VS Code shows the output of the server start command:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> emp_backend_service@1.0.0 dev
> nodemon server.js
Start the server
npm run dev

[nodemon] 3.1.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Employee-Service Backend Server STARTED Successfully!!!
MongoDB Connected Successfully!!!
```

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the project structure. The .env file is selected in the Explorer. The main editor area shows the contents of the .env file:

```
1 PORT=4000
2 DB_URL=mongodb://127.0.0.1:27017/astidb
```

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the project structure. The server.js file is selected in the Explorer. The main editor area shows the code for server.js:

```
1 import express from 'express'
2 import db_connection from './utils/db.js'
3 import dotenv from 'dotenv'
4 dotenv.config()
5
6 const app = express()
7
8 // Establish MongoDB Database Connection
9 db_connection()
10
11 app.listen(process.env.PORT, () => {
12     console.log("Employee-Service Backend Server STARTED Successfully!!!")
13 })
```

# Create Model, Schema

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure:
  - REACT\_...** (highlighted with a purple box)
  - emp\_mern\_proj**
  - emp\_backend**
  - controller** (highlighted with a purple box)
    - EmployeeCon...** (highlighted with a purple box)
  - models** (highlighted with a purple box)
    - Employee.js** (highlighted with a purple box)
  - node\_modules**
  - routes**
  - utils**
  - .env**
  - package-lock.j...**
  - package.json**
  - server.js**
- Editor (Right):** The **Employee.js** file is open, showing the following code:

```
import mongoose from "mongoose"

const employeeSchema = new mongoose.Schema({
    empid: String,
    name: String,
    age: Number,
    salary: Number,
    address: String
}, {timestamps: true})

const EmployeeModel = mongoose.model('employee', employeeSchema)

export default EmployeeModel
```
- Search Bar (Top Right):** Contains the text **React\_7AM\_WS**.

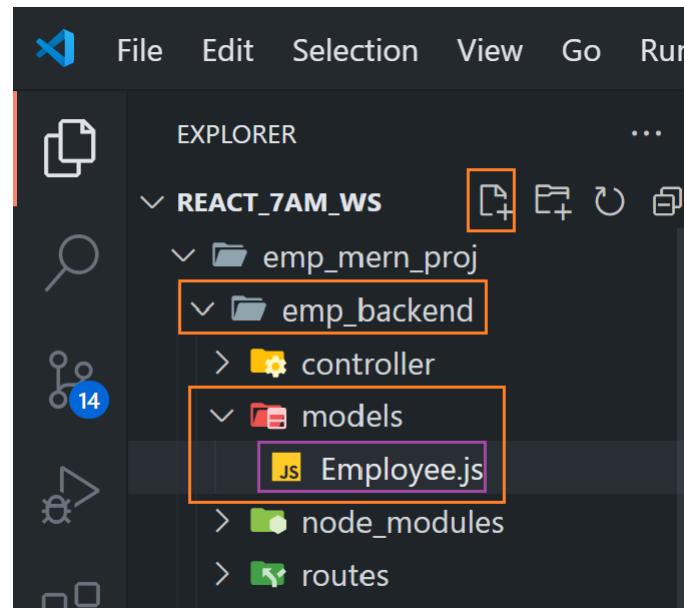
---

---

# **Let us Create A Employee Model**



# Create Employee Schema



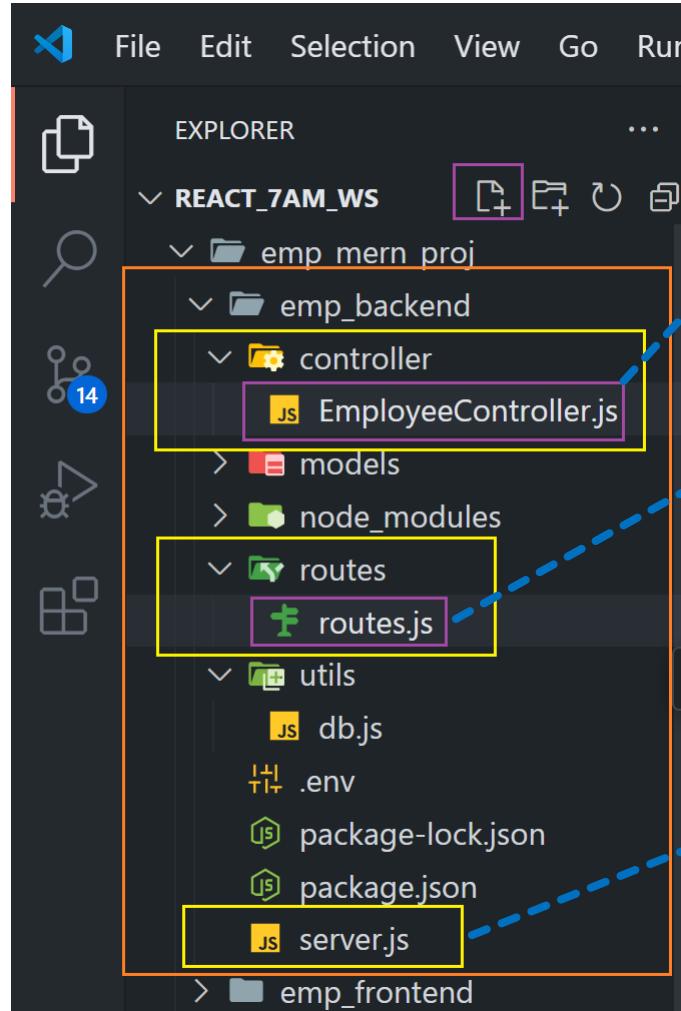
The code editor displays the **Employee.js** file content:

```
JS Employee.js ×  
emp_mern_proj > emp_backend > models > JS Employee.js > ...  
1 import mongoose from "mongoose"  
2  
3 const employeeSchema = new mongoose.Schema({  
4     empid: String,  
5     name: String,  
6     age: Number,  
7     salary: Number,  
8     address: String  
9 }, {timestamps: true})  
10  
11 const EmployeeModel = mongoose.model('employee', employeeSchema)  
12  
13 export default EmployeeModel
```

The code is annotated with three numbered circles on the right side:

1. Create Employee Schema (points to line 3)
2. Create Employee Model (points to line 11)
3. Export Employee Model (points to line 13)

# Create POST Handler Method & Route for CREATE Employee



```
EmployeeController.js
const CreateEmployee = (request, response) => {
  response.send("Create Employee Handler Method Got Called!!!")
}

export {CreateEmployee}
```

```
routes.js
import express from 'express'
import { CreateEmployee } from '../controller/EmployeeController.js'

const routers = express.Router()

//Router for Create Employee
routers.post('/employee', CreateEmployee)

export default routers
```

```
server.js
import express from 'express'
import db_connection from './utils/db.js'
import dotenv from 'dotenv'
import routers from './routes/routes.js'

dotenv.config()

const app = express()

// Establish MongoDB Database Connection
db_connection()

// Informing server about all routers
app.use('/api', routers)

app.listen(process.env.PORT, () => {
  console.log("Employee-Service Backend Server STARTED Successfully!!!")
})
```

Start the  
**SERVER**

# Testing POST Service End-Point using POSTMAN

The screenshot shows the Postman application interface. At the top, there is a search bar with the URL `POST http://127.0.0.1:4000/api/employee`. Below the search bar, the method is set to `POST` and the URL is `http://127.0.0.1:4000/api/employee`. To the right of the URL is a `Send` button. The main area has tabs for `Params`, `Auth`, `Headers (7)`, `Body`, `Pre-req.`, `Tests`, and `Settings`. The `Body` tab is selected, showing options for `Pretty`, `Raw`, `Preview`, `Visualize`, and `HTML`. The response section displays the message `1 Create Employee Handler Method Got Called!!!`. Above the response, status information is shown: `200 OK 11 ms 272 B` and a `Save Response` dropdown. On the far right, there are `Print` and `Copy` icons.

---

# **CREATE**

# **Employee API**



# Create Employee – Full Implementation – EmployeeController.js

```
js EmployeeController.js X
emp_mern_proj > emp_backend > controller > js EmployeeController.js > ...
1 import EmployeeModel from "../models/Employee.js"
2
3 const CreateEmployee = async(request, response) => {
4     //response.send("Create Employee Handler Method Got Called!!!")
5
6     try {
7         const {empid, name, age, salary, address} = request.body
8
9         const empModel = new EmployeeModel({empid, name, age, salary, address})
10        await empModel.save()
11        response.status(201).json({
12            success: true,
13            message: "Employee created successfully", empModel
14        })
15    } catch(error) {
16        console.log(error)
17        response.status(500).json({
18            success: false,
19            message: "Employee NOT Created due to Internal ERROR", empModel
20        })
21    }
22}
23
24 export {CreateEmployee}
```

# Create Employee – Full Implementation – server.js

```
JS server.js X  
emp_mern_proj > emp_backend > JS server.js > ...  
1 import express from 'express'  
2 import db_connection from './utils/db.js'  
3 import dotenv from 'dotenv'  
4 import routers from './routes/routes.js'  
5 import cors from 'cors'  
6  
7 dotenv.config()  
8  
9 const app = express()  
10  
11 // Establish MongoDB Database Connection  
12 db_connection()  
13  
14 // Use Middleware  
15 app.use(express.json())  
16 app.use(cors())  
17  
18 // Informing server about all routers  
19 app.use('/api', routers)  
20  
21 app.listen(process.env.PORT, () => {  
22   console.log("Employee-Service Backend Server STARTED Successfully!!!")  
23 })
```

# Testing POST/Create Service End-Point using POSTMAN

The screenshot shows the POSTMAN interface with the following details:

- Request URL:** `http://127.0.0.1:4000/api/employee`
- Method:** POST
- Body (JSON):**

```
1 {"empid": "1001",
2  "name": "Md Ali",
3  "age": 25,
4  "salary": 25000,
5  "address": "Bengaluru"}  
6  
7
```
- Response Status:** 201 Created
- Response Headers:** 59 ms, 541 B
- Response Body (Pretty JSON):**

```
1 {"success": true,
2  "message": "Employee created successfully",
3  "empModel": {
4    "empid": "1001",
5    "name": "Md Ali",
6    "age": 25,
7    "salary": 25000,
8    "address": "Bengaluru",
9    "_id": "6652c10d0bb00293ff2f392a",
10   "createdAt": "2024-05-26T04:56:45.493Z",
11   "updatedAt": "2024-05-26T04:56:45.493Z",
12   "__v": 0
13 }
14 }
15 }
```

```
astidb> show collections
employee
employees
astidb> db.employees.find()
[
  {
    _id: ObjectId('6652c10d0bb00293ff2f392a'),
    empid: '1001',
    name: 'Md Ali',
    age: 25,
    salary: 25000,
    address: 'Bengaluru',
    createdAt: ISODate('2024-05-26T04:56:45.493Z'),
    updatedAt: ISODate('2024-05-26T04:56:45.493Z'),
    __v: 0
  }
]
astidb>
```

---

# **GET ALL**

# **Employee API**



# Get All Employees – Implementation – EmployeeController.js

```
26 //Handler method for GET ALL Employees
27 const GetAllEmployees = async(request, response) => {
28     try {
29         const employees = await EmployeeModel.find()
30
31         if(!employees) {
32             return response.status(404).json({
33                 success: false,
34                 message: "Employee Data NOT Found in Database"
35             })
36         }
37
38         // ELSE If Employees are found then return those employees
39         return response.status(200).json({
40             success: true,
41             message: "Got List of Employees successfully",
42             employees
43         })
44
45     } catch(error) {
46         console.log(error)
47         return response.status(500).json({
48             success: false,
49             message: "Unable to Get Employees due to Internal ERROR"
50         })
51     }
52 }
53
54 export {CreateEmployee, GetAllEmployees}
```

# Get All Employees – Implementation – routes.js

```
routes.js  X

emp_mern_proj > emp_backend > routes > routes.js > [o] default

1 import express from 'express'
2 import { CreateEmployee, GetAllEmployees } from '../controller/EmployeeController.js'
3
4 const routers = express.Router()
5
6 //Router for Create Employee
7 routers.post('/employee', CreateEmployee)
8
9 //Router for Get All Employees
10 routers.get('/employee', GetAllEmployees)
11
12 export default routers
```

# Testing GET All End-Point using POSTMAN

The screenshot shows the Postman interface with a successful API call and a MongoDB query result.

**Postman Request:**

- Method: GET
- URL: `http://127.0.0.1:4000/api/employee`
- Body tab selected
- Response status: 201 Created
- Response time: 8 ms
- Response size: 747 B

**Postman Response (Pretty JSON):**

```
1:   "success": true,
2:   "message": "Got List of Employees successfully",
3:   "employees": [
4:     {
5:       "_id": "6652c10d0bb00293ff2f392a",
6:       "empid": "1001",
7:       "name": "Md Ali",
8:       "age": 25,
9:       "salary": 25000,
10:      "address": "Bengaluru",
11:      "createdAt": "2024-05-26T04:56:45.493Z",
12:      "updatedAt": "2024-05-26T04:56:45.493Z",
13:      "__v": 0
14:    },
15:    {
16:      "_id": "6652eaf6f39e431e0f78dde8",
17:      "empid": "1002",
18:      "name": "Malli",
19:      "age": 30,
20:      "salary": 30000,
21:      "address": "Chennai-28",
22:      "createdAt": "2024-05-26T07:55:34.061Z",
23:      "updatedAt": "2024-05-26T07:55:34.061Z",
24:      "__v": 0
25:    }
26:  ]
27: }
```

**MongoDB Query Result:**

```
astidb> db.employees.find()
[
  {
    _id: ObjectId('6652c10d0bb00293ff2f392a'),
    empid: '1001',
    name: 'Md Ali',
    age: 25,
    salary: 25000,
    address: 'Bengaluru',
    createdAt: ISODate('2024-05-26T04:56:45.493Z'),
    updatedAt: ISODate('2024-05-26T04:56:45.493Z'),
    __v: 0
  },
  {
    _id: ObjectId('6652eaf6f39e431e0f78dde8'),
    empid: '1002',
    name: 'Malli',
    age: 30,
    salary: 30000,
    address: 'Chennai-28',
    createdAt: ISODate('2024-05-26T07:55:34.061Z'),
    updatedAt: ISODate('2024-05-26T07:55:34.061Z'),
    __v: 0
  }
]
```

---

---

---

# **GET By ID/PK**

# **Employee API**



# GET Employee By Id/Pk – Implementation – EmployeeController.js

```
116 //Handler method for GET Employee By Id/Pk
117 const GetEmployee = async(request, response) => {
118     try {
119         const employeePk = request.params.id //Getting path variable
120         const employee = await EmployeeModel.findById(employeePk)
121
122         if(!employee) {
123             return response.status(404).json({
124                 success: false,
125                 message: "Employee Data NOT Found in Database"
126             })
127         }
128
129         // If Employee found then return employee
130         return response.status(200).json({
131             success: true,
132             message: "Got Employee successfully",
133             employee
134         })
135
136     } catch(error) {
137         console.log(error)
138         return response.status(500).json({
139             success: false,
140             message: "Unable to Get Employee due to Internal ERROR"
141         })
142     }
143 }
144
145 export {CreateEmployee, GetAllEmployees, GetEmployee, UpdateEmployee, DeleteEmployee}
```

# GET Employee By Id – Implementation – routes.js

```
routes.js  X  
emp_mern_proj > emp_backend > routes > routes.js > ...  
1 import express from 'express'  
2 import { CreateEmployee, GetAllEmployees, GetEmployee, UpdateEmployee, DeleteEmployee } from '../controller/EmployeeController.js'  
3  
4 const routers = express.Router()  
5  
6 //Router for Create Employee  
7 routers.post('/employee', CreateEmployee)  
8  
9 //Router for Get All Employees  
10 routers.get('/employee', GetAllEmployees)  
11  
12 //Router for Get Employee By Id  
13 routers.get('/employee/:id', GetEmployee)  
14  
15 //Router for Update Employee By Id  
16 routers.put('/employee/:id', UpdateEmployee)  
17  
18 //Router for Delete Employee By Id  
19 routers.delete('/employee/:id', DeleteEmployee)  
20  
21 export default routers
```

# Testing - GET Employee By ID/PK End-Point using POSTMAN

The screenshot shows the POSTMAN interface with the following details:

- Method:** GET
- URL:** http://127.0.0.1:4000/api/employee/6652c10d0bb00293ff2f392a
- Headers:** (6)
- Body:** (Empty)
- Tests:** (Empty)
- Settings:** (Empty)
- Query Params:** (Empty)
- Body (Pretty):** (JSON view)

```
1 "success": true,
2 "message": "Got Employee successfully",
3 "employee": {
4     "_id": "6652c10d0bb00293ff2f392a",
5     "empid": "1001",
6     "name": "Md Ali",
7     "age": 25,
8     "salary": 25000,
9     "address": "Bengaluru",
10    "createdAt": "2024-05-26T04:56:45.493Z",
11    "updatedAt": "2024-05-26T04:56:45.493Z",
12    "__v": 0
13 }
14
15 }
```
- Test Results:** (Empty)
- Status:** 200 OK
- Time:** 11 ms
- Size:** 532 B
- Save Response:** (Available)

```
astidb> db.employees.find({"empid": "1001"})
[
  {
    _id: ObjectId('6652c10d0bb00293ff2f392a'),
    empid: '1001',
    name: 'Md Ali',
    age: 25,
    salary: 25000,
    address: 'Bengaluru',
    createdAt: ISODate('2024-05-26T04:56:45.493Z'),
    updatedAt: ISODate('2024-05-26T04:56:45.493Z'),
    __v: 0
  }
]
astidb> |
```

---

# **UPDATE By ID/PK**

# **Employee API**

# Update Employee By Id – Implementation – EmployeeController.js

```
54 //Handler method for UPDATE Employee
55 const UpdateEmployee = async(request, response) => {
56     try {
57         const employeePk = request.params.id //Getting path variable
58
59         const updatedEmployee = await EmployeeModel.findByIdAndUpdate(
60             employeePk,
61             request.body,
62             {new: true}
63         )
64
65         if(!updatedEmployee) {
66             return response.status(404).json({
67                 success: false,
68                 message: "Employee Data NOT Found in Database for UPDATE"
69             })
70         }
71
72         response.status(200).json({
73             success: true,
74             message: "Employee updated successfully",
75             updatedEmployee
76         })
77
78     }catch(error) {
79         console.log(error)
80         response.status(500).json({
81             success: false,
82             message: "Employee NOT Updated due to Internal ERROR"
83         })
84     }
85 }
86
87 export {CreateEmployee, GetAllEmployees, UpdateEmployee}
```

# Update Employee By Id – Implementation – routes.js

```
routes.js  X  
emp_mern_proj > emp_backend > routes > routes.js > default  
1 import express from 'express'  
2 import { CreateEmployee, GetAllEmployees, UpdateEmployee } from '../controller/EmployeeController.js'  
3  
4 const routers = express.Router()  
5  
6 //Router for Create Employee  
7 routers.post('/employee', CreateEmployee)  
8  
9 //Router for Get All Employees  
10 routers.get('/employee', GetAllEmployees)  
11  
12 //Router for Update Employee By Id  
13 routers.put('/employee/:id', UpdateEmployee)  
14  
15 export default routers
```

# Testing - Update Employee By PK End-Point using POSTMAN

The screenshot shows the Postman interface for a PUT request to the endpoint `http://127.0.0.1:4000/api/employee/6652eaf6f39e431e0f78dde8`. The request body is a JSON object:

```
1
2   ...
3   "empid": 1002,
4   ...
5   "name": "Malli Karjuna Reddy",
6   ...
7   "age": 35,
8   ...
9   "salary": 35000,
10  ...
11  "address": "Chennai-2828"
12
13 }
```

The response status is 200 OK, with a response body indicating success:

```
1
2   ...
3   "success": true,
4   ...
5   "message": "Employee updated successfully",
6   ...
7   "updatedEmployee": {
8     ...
9     "_id": "6652eaf6f39e431e0f78dde8",
10    ...
11    "empid": "1002",
12    ...
13    "name": "Malli Karjuna Reddy",
14    ...
15    "age": 35,
16    ...
17    "salary": 35000,
18    ...
19    "address": "Chennai-2828",
20    ...
21    "createdAt": "2024-05-26T07:55:34.061Z",
22    ...
23    "updatedAt": "2024-05-26T08:46:08.692Z",
24    ...
25    "__v": 0
26
27 }
```

```
astidb> db.employees.find({"empid": "1002"})
[
  {
    ...
    "_id": ObjectId('6652eaf6f39e431e0f78dde8'),
    "empid": '1002',
    "name": 'Malli Karjuna Reddy',
    "age": 35,
    "salary": 35000,
    "address": 'Chennai-2828',
    "createdAt": ISODate('2024-05-26T07:55:34.061Z'),
    "updatedAt": ISODate('2024-05-26T08:46:08.692Z'),
    "__v": 0
  }
]
1db> |
```

---

# **DELETE By ID/PK**

# **Employee API**



# Delete Employee By Id – Implementation – EmployeeController.js

```
JS EmployeeController.js X
emp_mern_proj > emp_backend > controller > JS EmployeeController.js > ...
87 //Handler method for Delete Employee By Id/Pk
88 const DeleteEmployee = async(request, response) => {
89     try {
90         const employeePk = request.params.id //Getting path variable
91
92         const deletedEmployee = await EmployeeModel.findByIdAndDelete(employeePk)
93
94         if(!deletedEmployee) {
95             return response.status(404).json({
96                 success: false,
97                 message: "Employee Data NOT Found in Database for DELETE"
98             })
99         }
100
101         response.status(200).json({
102             success: true,
103             message: "Employee deleted successfully",
104             deletedEmployee
105         })
106
107     } catch(error) {
108         console.log(error)
109         response.status(500).json({
110             success: false,
111             message: "Employee NOT Deleted due to Internal ERROR"
112         })
113     }
114 }
115
116 export {CreateEmployee, GetAllEmployees, UpdateEmployee, DeleteEmployee}
```

# Delete Employee By Id – Implementation – routes.js

```
routes.js X  
emp_mern_proj > emp_backend > routes > routes.js > ...  
1 import express from 'express'  
2 import { CreateEmployee, GetAllEmployees, UpdateEmployee, DeleteEmployee } from '../controller/EmployeeController.js'  
3  
4 const routers = express.Router()  
5  
6 //Router for Create Employee  
7 routers.post('/employee', CreateEmployee)  
8  
9 //Router for Get All Employees  
10 routers.get('/employee', GetAllEmployees)  
11  
12 //Router for Update Employee By Id  
13 routers.put('/employee/:id', UpdateEmployee)  
14  
15 //Router for Update Employee By Id  
16 routers.delete('/employee/:id', DeleteEmployee)  
17  
18 export default routers
```

# Testing - Delete Employee By PK End-Point using POSTMAN

The screenshot shows the Postman interface with a red box highlighting the URL field containing `http://127.0.0.1:4000/api/employee/665322cfdc3542cca4f20319`. The status bar at the top right indicates `Status: 200 OK Time: 19 ms Size: 543 B`. The Body tab displays a JSON response with a red box around it:

```
1 "success": true,
2 "message": "Employee deleted successfully",
3 "deletedEmployee": {
4     "_id": "665322cfdc3542cca4f20319",
5     "empid": "1003",
6     "name": "Priya",
7     "age": 30,
8     "salary": 30000,
9     "address": "Chennai-28",
10    "createdAt": "2024-05-26T11:53:51.358Z",
11    "updatedAt": "2024-05-26T11:53:51.358Z",
12    "__v": 0
13 }
14
15 }
```

```
astidb> db.employees.find()
[
  {
    _id: ObjectId('6652c10d0bb00293ff2f392a'),
    empid: '1001',
    name: 'Md Ali',
    age: 25,
    salary: 25000,
    address: 'Bengaluru',
    createdAt: ISODate('2024-05-26T04:56:45.493Z'),
    updatedAt: ISODate('2024-05-26T04:56:45.493Z'),
    __v: 0
  },
  {
    _id: ObjectId('6652eaf6f39e431e0f78dde8'),
    empid: '1002',
    name: 'Malli Karjuna Reddy',
    age: 35,
    salary: 35000,
    address: 'Chennai-2828',
    createdAt: ISODate('2024-05-26T07:55:34.061Z'),
    updatedAt: ISODate('2024-05-26T08:46:08.692Z'),
    __v: 0
  }
]
```

```
astidb> db.employees.find({"empid": "1003"})
astidb> |
```

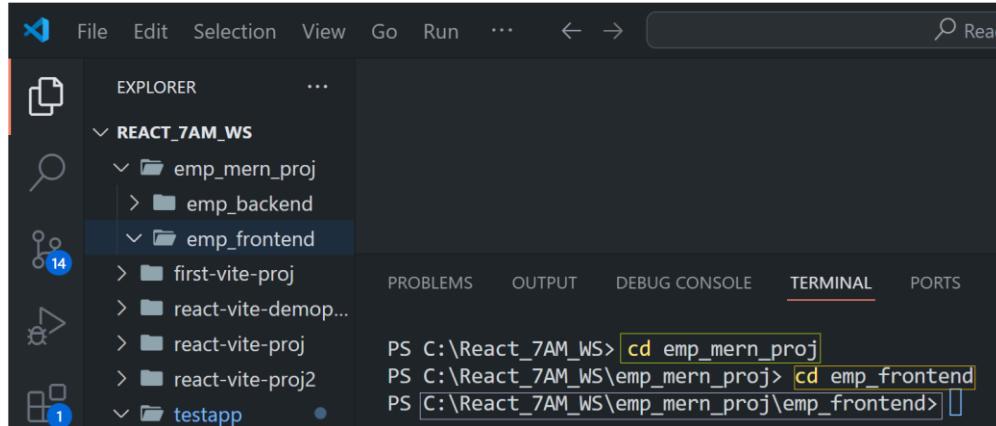
---

# Frontend Implementation



# Create React Application

- ✓ From Terminal go to emp\_frontend folder



- ✓ Create React Application using **vite** command

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\React_7AM_WS\emp_mern_proj\emp_frontend> npm create vite@latest
✓ Project name: ... emp-client
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client...

Done. Now run:
cd emp-client
npm install
npm run dev

PS C:\React_7AM_WS\emp_mern_proj\emp_frontend>
```

The terminal window shows the execution of the `npm create vite@latest` command. It prompts for a project name ('emp-client'), selects 'React' as the framework, and 'JavaScript' as the variant. It then starts scaffolding the project in the current directory. Once completed, it provides instructions to run the project with `cd emp-client`, `npm install`, and `npm run dev`. The final prompt is `PS C:\React_7AM_WS\emp_mern_proj\emp_frontend>`.

```
cd emp-client
npm install
npm run dev

PS C:\React_7AM_WS\emp_mern_proj\emp_frontend> cd emp-client
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> npm install
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check
out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much
more comprehensive and powerful.
npm WARN deprecated @humanwhocodes/config-array@0.11.14: Use @eslint/config-array instead
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated @humanwhocodes/object-schema@2.0.3: Use @eslint/object-schema instead

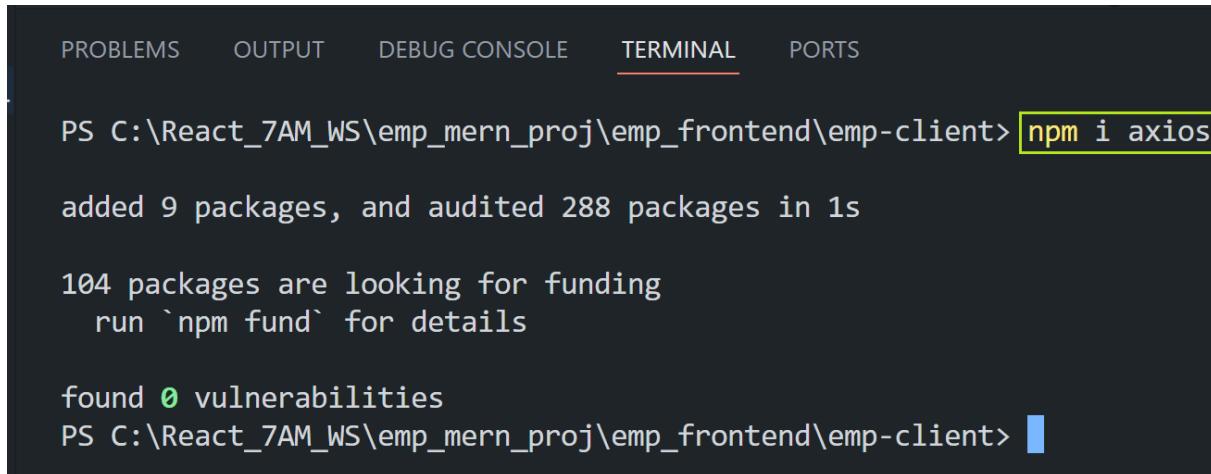
added 278 packages, and audited 279 packages in 39s

103 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client>
```

# Installing Required Software/packages

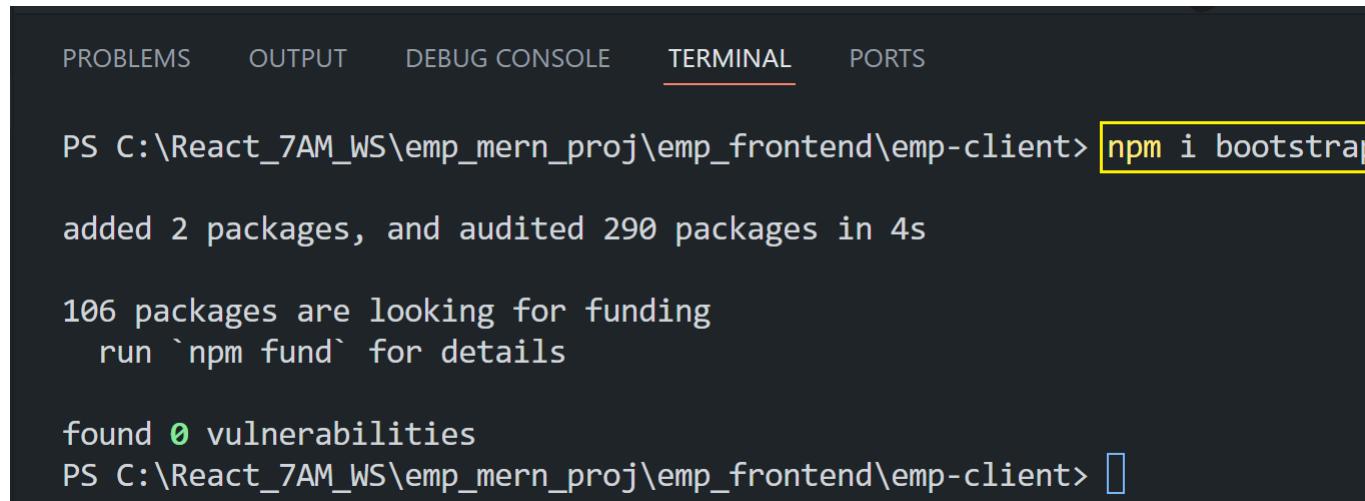
## ✓ Install Axios



A screenshot of a terminal window within a dark-themed code editor interface. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined in red), and PORTS. The terminal output shows the command `npm i axios` being run in a directory path `C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client>`. The response indicates that 9 packages were added, 288 packages were audited in 1 second, 104 packages were looking for funding (with details available via `npm fund`), and 0 vulnerabilities were found. The command prompt ends with a blue square icon.

```
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> npm i axios
added 9 packages, and audited 288 packages in 1s
104 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> 
```

## ✓ Install Bootstrap



A screenshot of a terminal window within a dark-themed code editor interface. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined in red), and PORTS. The terminal output shows the command `npm i bootstrap` being run in a directory path `C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client>`. The response indicates that 2 packages were added, 290 packages were audited in 4 seconds, 106 packages were looking for funding (with details available via `npm fund`), and 0 vulnerabilities were found. The command prompt ends with a blue square icon.

```
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> npm i bootstrap
added 2 packages, and audited 290 packages in 4s
106 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> 
```

# Installing Required Software/packages ...

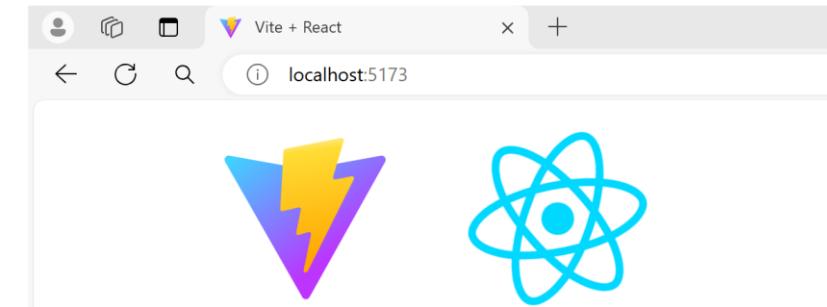
- ✓ Install **Hot toast** For adding Beautiful Notifications to React Application

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> npm i react-hot-toast
added 2 packages, and audited 292 packages in 3s

106 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client>
```



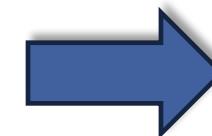
- ✓ Run the client application

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\React_7AM_WS\emp_mern_proj\emp_frontend\emp-client> npm run dev
> emp-client@0.0.0 dev
> vite

VITE v5.3.1 ready in 238 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```



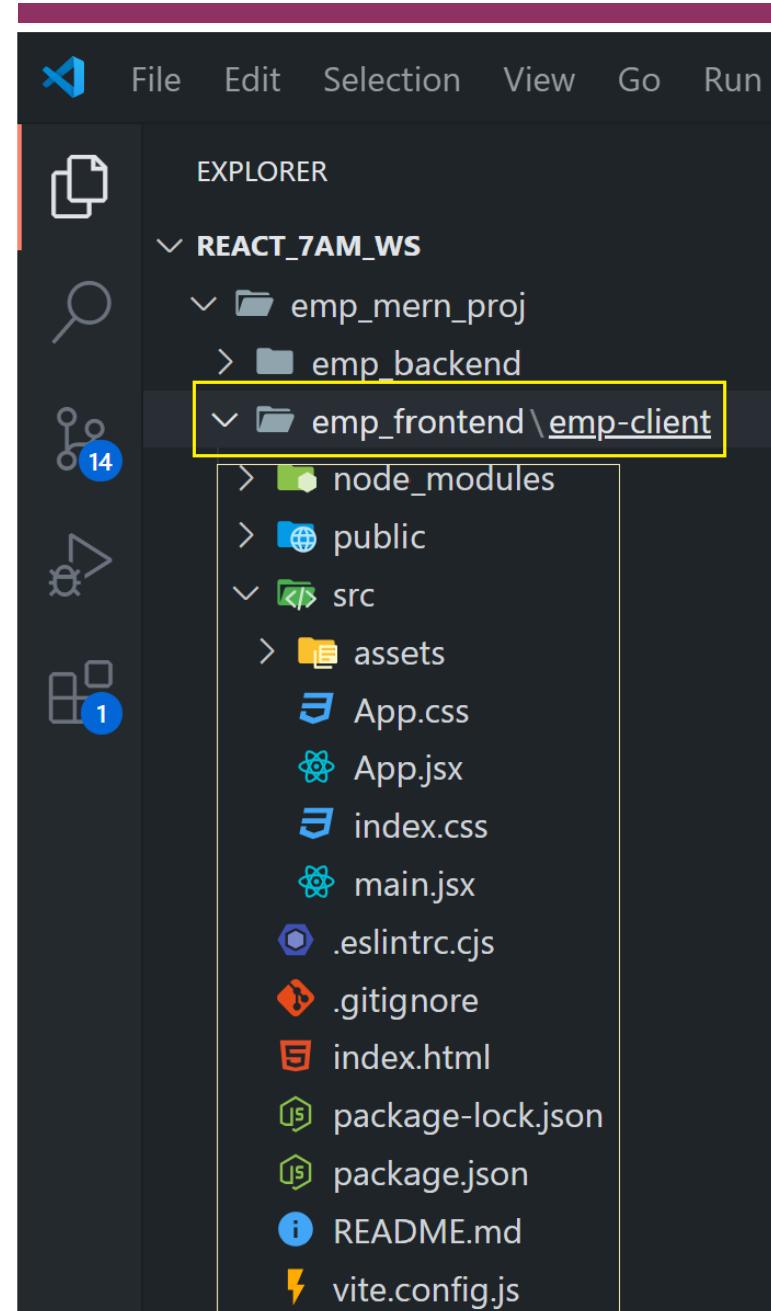
## Vite + React

count is 0

Edit src/App.jsx and save to test HMR

Click on the Vite and React logos to learn more

# React Application Folder Structure

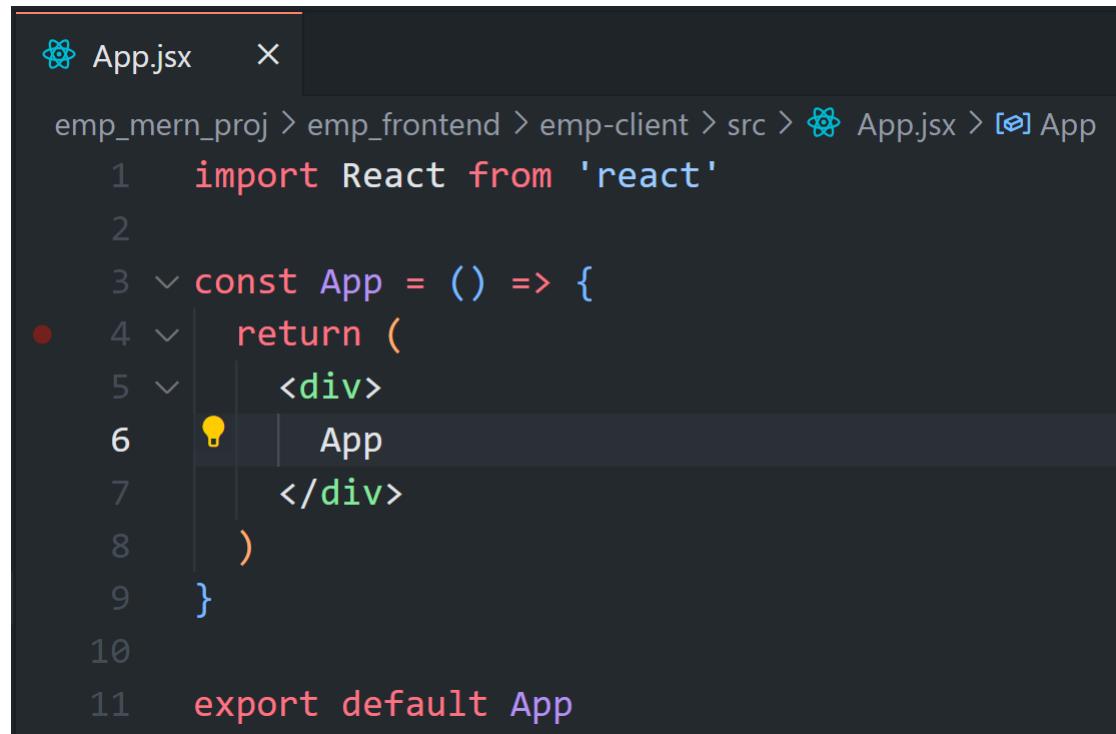


# Existing React Application Cleanup

- ✓ Delete existing code from **App.jsx** & Create a function



A screenshot of the VS Code interface showing the file 'App.jsx' open. The cursor is at the start of the word 'rafce'. A code completion dropdown menu is displayed, listing three suggestions: 'rafce' (highlighted with an orange border), 'reactArrowFunctionExportComponent', and 'ReadableStreamDefaultController'.

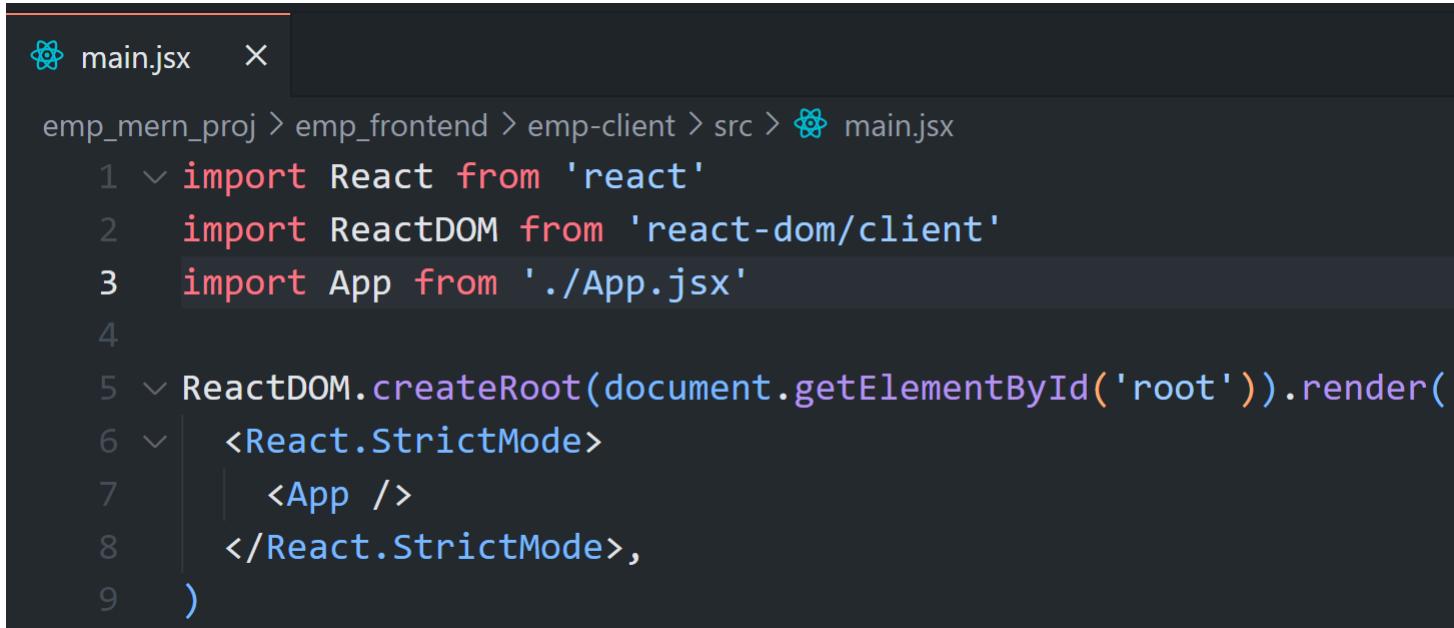


```
App.jsx

emp_mern_proj > emp_frontend > emp-client > src > App.jsx > [e] App
1 import React from 'react'
2
3 const App = () => {
4   return (
5     <div>
6       | App
7       </div>
8     )
9   }
10
11 export default App
```

A screenshot of the VS Code interface showing the file 'App.jsx' open. The code defines a functional component 'App' that returns a single 'div' element containing the text 'App'. The code editor highlights the 'App' text with a yellow lightbulb icon, indicating it might be a candidate for refactoring or extraction into a separate component.

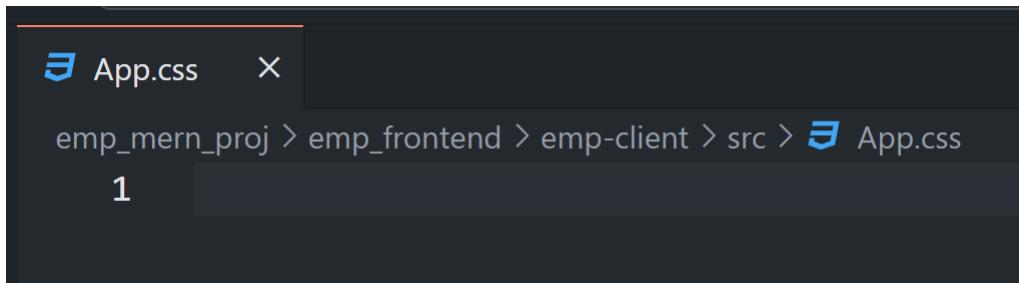
✓ Delete import index.css from **main.jsx**



The screenshot shows a code editor with a dark theme. The file being edited is 'main.jsx'. The code is as follows:

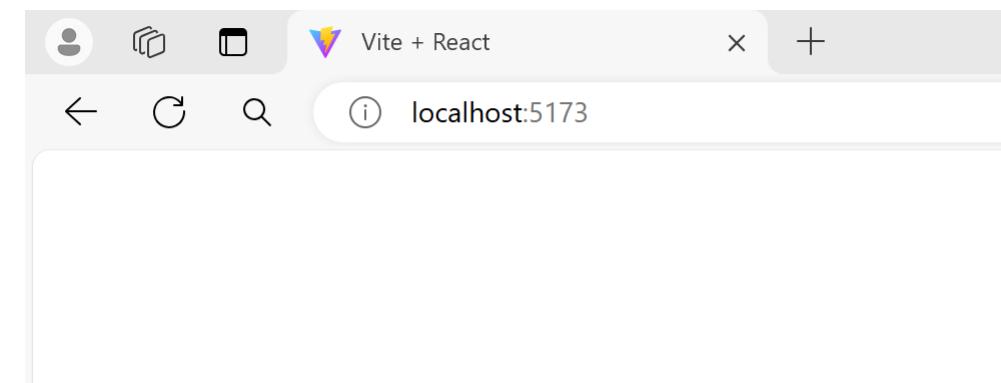
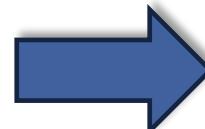
```
main.jsx  X  
emp_mern_proj > emp_frontend > emp-client > src > main.jsx  
1  import React from 'react'  
2  import ReactDOM from 'react-dom/client'  
3  import App from './App.jsx'  
4  
5  ReactDOM.createRoot(document.getElementById('root')).render(  
6    <React.StrictMode>  
7      <App />  
8    </React.StrictMode>,  
9  )
```

✓ Delete all code from **App.css**

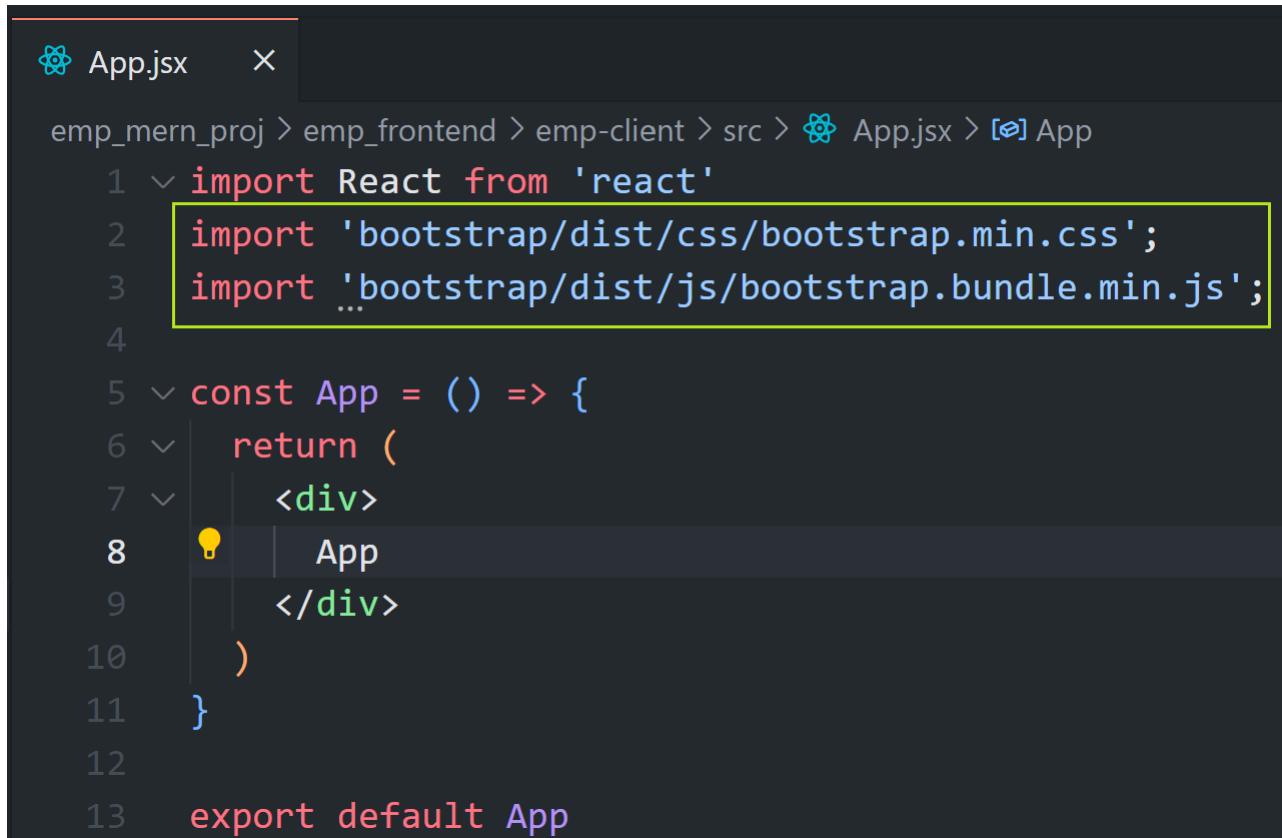


The screenshot shows a code editor with a dark theme. The file being edited is 'App.css'. The code is as follows:

```
App.css  X  
emp_mern_proj > emp_frontend > emp-client > src > App.css  
1
```



## ✓ Integrating Bootstrap into React application – App.jsx



```
App.jsx  X  
emp_mern_proj > emp_frontend > emp-client > src > App.jsx > [App]  
1 import React from 'react'  
2 import 'bootstrap/dist/css/bootstrap.min.css';  
3 import 'bootstrap/dist/js/bootstrap.bundle.min.js';  
4  
5 const App = () => {  
6   return (  
7     <div>  
8       App  
9     </div>  
10    )  
11  }  
12  
13 export default App
```

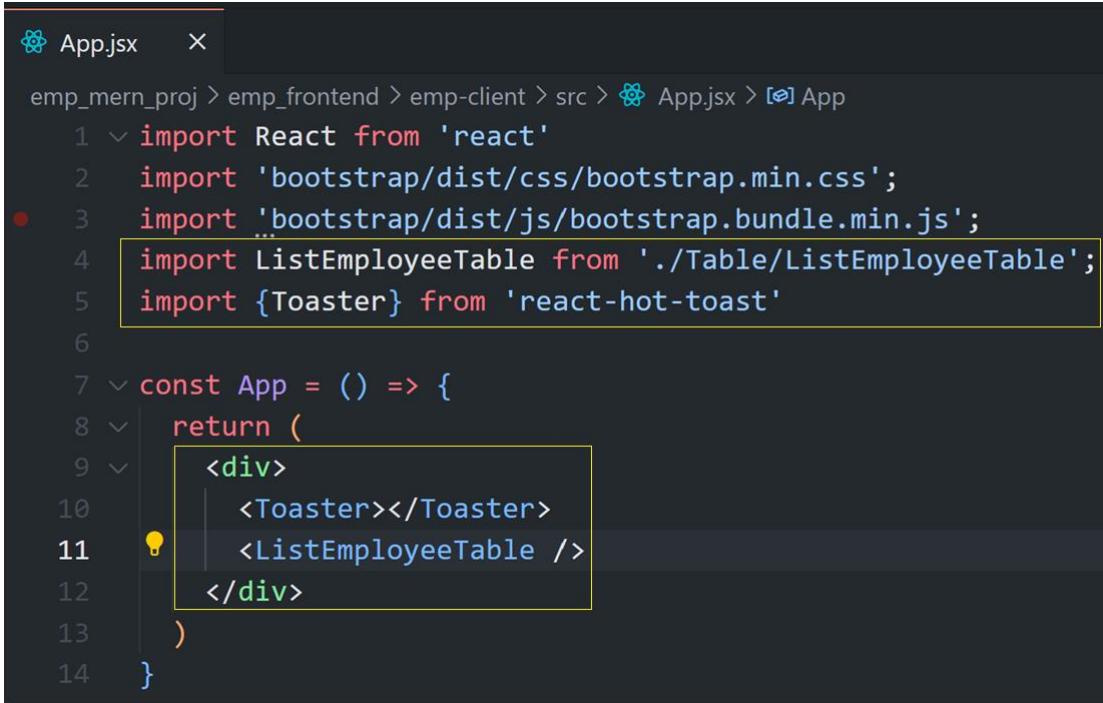
- ✓ Create a new folder called “**Table**” in “**src**” folder
- ✓ Create react component called “**ListEmployeeTable.jsx**” inside “**Table**” folder
- ✓ Create **rafce**

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar is the Explorer view, displaying the project structure of 'REACT\_7AM\_WS'. Inside 'src', there is a 'Table' folder which contains 'ListEmployeeTable.jsx'. This file is currently open in the main editor area. The code in the editor is:

```
1 import React from 'react'
2
3 const ListEmployeeTable = () => {
4     return (
5         <div>
6             ListEmployeeTable Component
7         </div>
8     )
9 }
10
11 export default ListEmployeeTable
```

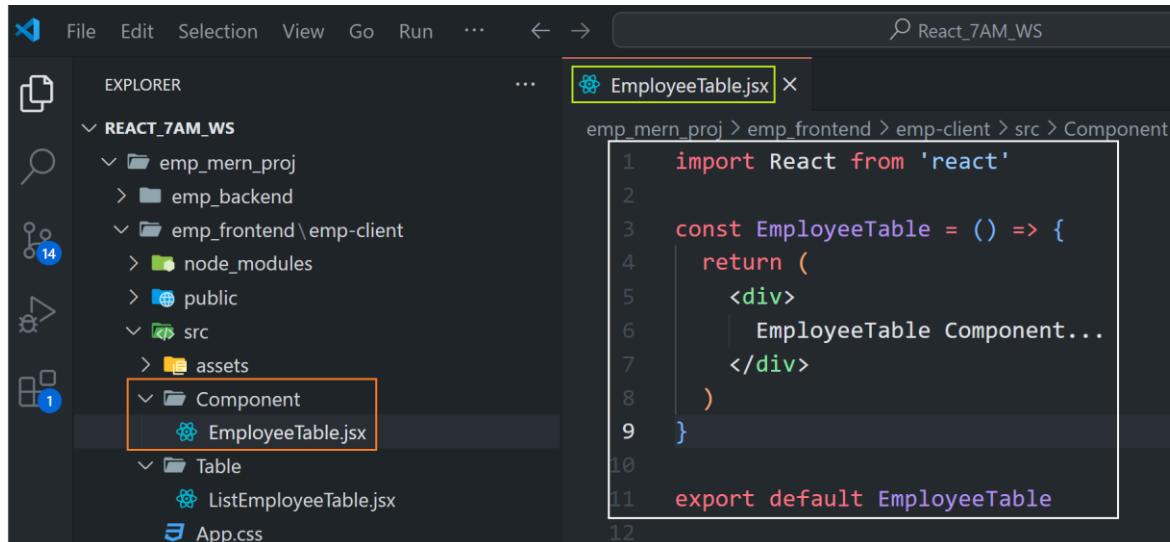
The 'ListEmployeeTable.jsx' file is highlighted with a yellow box. The status bar at the bottom left shows '1' file has changes.

✓ Call `ListEmployeeTable.jsx` component from `App.jsx`



```
App.jsx  X  
emp_mern_proj > emp_frontend > emp-client > src > App.jsx > App  
1 import React from 'react'  
2 import 'bootstrap/dist/css/bootstrap.min.css';  
● 3 import './bootstrap/dist/js/bootstrap.bundle.min.js';  
4 import ListEmployeeTable from './Table/ListEmployeeTable';  
5 import {Toaster} from 'react-hot-toast'  
6  
7 const App = () => {  
8   return (  
9     <div>  
10       <Toaster></Toaster>  
11       <ListEmployeeTable />  
12     </div>  
13   )  
14 }
```

- ✓ Create a new folder called “**Component**” in “**src**” folder
- ✓ Create react component called “**EmployeeTable.jsx**” inside “**Component**” folder
- ✓ Create **rafce**



The screenshot shows the VS Code interface with the following details:

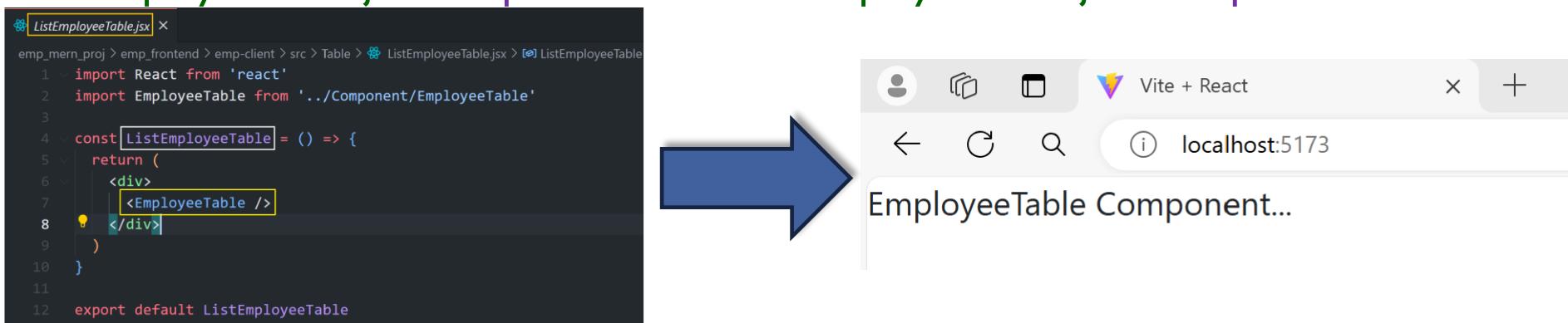
- File Explorer:** Shows the project structure under "REACT\_7AM\_WS". A folder named "Component" is highlighted with an orange border.
- Editor:** The file "EmployeeTable.jsx" is open. The code is as follows:

```

1 import React from 'react'
2
3 const EmployeeTable = () => {
4   return (
5     <div>
6       | EmployeeTable Component...
7     </div>
8   )
9 }
10
11 export default EmployeeTable

```

- ✓ Call **EmployeeTable.jsx** component from **ListEmployeeTable.jsx** Component



The screenshot shows two views side-by-side:

- VS Code Editor:** The file "ListEmployeeTable.jsx" is open. The code is as follows:

```

1 import React from 'react'
2 import EmployeeTable from '../Component/EmployeeTable'
3
4 const ListEmployeeTable = () => {
5   return (
6     <div>
7       <EmployeeTable />
8     </div>
9   )
10
11 export default ListEmployeeTable

```


- Browser Preview:** A screenshot of a browser window titled "Vite + React" showing the output of the code. The page displays the text "EmployeeTable Component...".

## Adding JSX code in List Employee

---

- ✓ Add JSX code in “**EmployeeTable.jsx**” for listing existing Employees in DB
- ✓ Add CSS Code in App.css

Vite + React

localhost:5173

## Ascent Employee Management System

Add New Employee

Emp Id	Name	Age	Salary	Address	Actions
1001	Md Ali	25	25000	Bengaluru	<span style="color: yellow;">Edit</span> <span style="color: red;">Delete</span>

# Design Create Employee Form

Vite + React

localhost:5173

## Ascent Employee Management System

### Create Employee

Emp Id	Name
1001	Md Ali

Address: Bengaluru

Add New Employee

Address	Actions
Bengaluru	<span>Edit</span> <span>Delete</span>

Emp ID  
Name  
Age  
Salary  
Address

Cancel Add

# Design Update Employee Form

The screenshot displays a web application interface for managing employees. At the top, a header bar shows the title "Ascent Employee Management System". Below the header, a table lists employees with columns for "Emp Id" and "Name". In the table, the first row shows "1001" and "Md Ali". To the right of the table, a modal window titled "Edit Employee" is open, containing input fields for "Emp ID", "Name", "Age", "Salary", and "Address". The "Address" field contains the value "Bengaluru". At the bottom of the modal, there are "Cancel" and "Update" buttons. The background of the page shows a dark grey sidebar with a green header section containing a "Add New Employee" button.

Vite + React

localhost:5173

Ascent Employee Management System

Edit Employee x

Emp Id	Name
1001	Md Ali

Emp ID

Name

Age

Salary

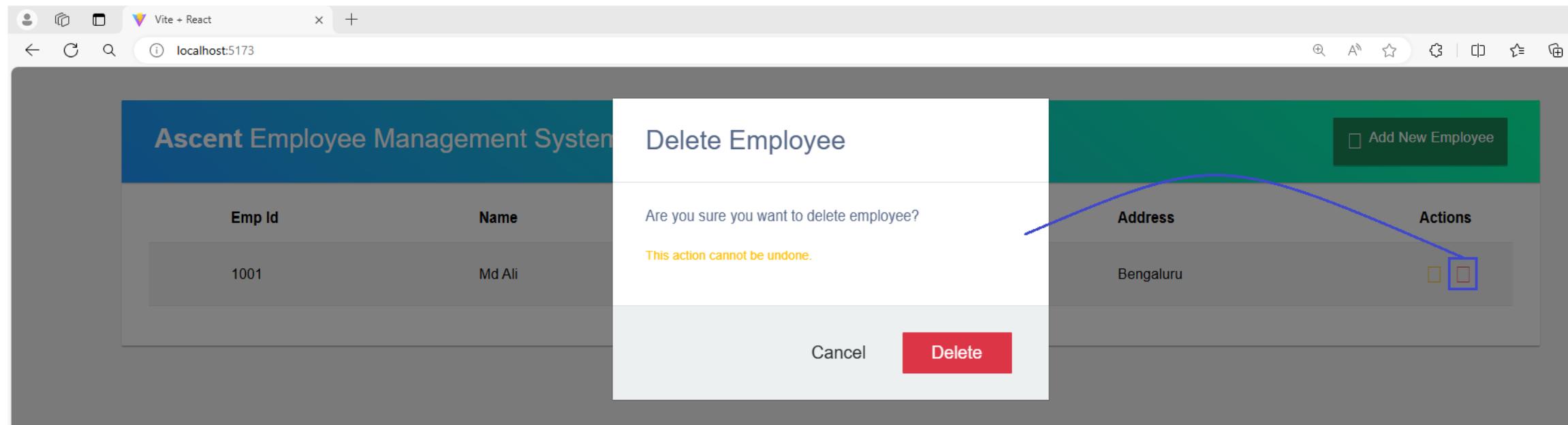
Address

Cancel Update

Add New Employee

Actions

# Design Delete Employee Form



---

---

# Update Employee



# Update Functionality

-----> *ListEmployeeTable.js*

The screenshot shows the Ascent Employee Management System interface. On the left, there is a table with columns: Emp Id, Name, Age, Salary, Address, and Actions. The table contains several rows of employee data. In the center, a modal window titled "Edit Employee" is open, showing form fields for Emp ID, Name, Age, Salary, and Address, along with a "Cancel" and "Update" button. To the right of the modal is a list of addresses: Bengaluru-104, Chennai-28, Bengaluru, Mumbai, Bengaluru, Chennai-28, and Chennai-28. Each address has edit and delete icons next to it. A dashed arrow points from the "Actions" column in the table to the "Edit Employee" modal, indicating the flow of data or function calls between them.

*EmployeeTable.js*

- Two Activities has to be done
  - Need to pass id which is going to update
  - Need to pass employee object
- From EmployeeTable.js, When we include EmployeeTable Component we need to pass a function called UpdateEmployee through which we can send id, employee object
- From EmployeeTable, when click on update icon will call UpdateEmployee Function with **id** so that this function will call get employee end point to get employee from DB
- From ListEmployeeTable we include EditEmployee component and we pass handleOnSubmit(), employee obj, handleChange()
- handleChange() will get latest employee values
- When user clicks on update button handleOnSubmit() will call update end-point to update the employee details in DB.