📊 Employee Management vpf 23rd dec 2025

1. Employee Creation Process

Frontend (EmployeeAddForm.js)

javascript

```
// Role selection करने पर
useEffect(() => {
  if (formData.role) {
    const generatedCode = generateEmployeeCode(formData.role); // Local generate
    // Auto-fill designation based on role
  }
}, [formData.role]);

// Save button click पर
const handleSaveEmployee = async (e) => {
  e.preventDefault();
  const response = await axiosInstance.post(
    "/api/employee/addEmployee", // ✅ API 1
    formData
  );
};
```

Backend Flow

text

```
🚀 API: POST /api/employee/addEmployee
  ├─── ✅ Employee मॉडल में नया employee create होता है
  ├─── ✅ Password hash होता है
  ├─── ✅ Role के हिसाब से auto-save होता है
  │     ├─── if role = "Telecaller" → Telecaller मॉडल में save
```

```
|     └──── if role = "HR" → HR मॉडल में save
└──── ✅ Response भेजा जाता है
```

2. Employee Listing Process

Frontend (EmployeeList.js)

javascript

```
// Component load होने पर और role tab change करने पर
useEffect(() => {
  fetchEmployeesByRole(activeTab);
}, [activeTab]);

const fetchEmployeesByRole = async (role) => {
  switch (role) {
    case "telecaller":
      // ✅ API 2
      const telecallerResponse = await axiosInstance.get("/api/telecaller");
      break;

    case "hr":
      // ✅ API 3
      const hrResponse = await axiosInstance.get("/api/hr");
      break;

    default:
      // ✅ API 4 (सभी employees लाना)
      const employeeResponse = await axiosInstance.get(
        "/api/employee/getAllEmployees"
      );
```

```
    // फिर role के हिसाब से filter करना
  }
};
```

🔗 API Calls और Operations Summary

📍 EMPLOYEE CREATION APIs

API 1: Employee Add

http

POST /api/employee/addEmployee

Purpose: Master employee record create करना
Request Body: Full employee data (name, role, email, etc.)
Backend Actions:

1. Employee मॉडल में save करना
2. Password hash करना
3. Role-based auto-save:
   - Telecaller → Telecaller मॉडल में save
   - HR → HR मॉडल में save
   - Others → जैसे है वैसे save
4. Response भेजना

API 2: Last Employee Code Get (Optional)

http

GET /api/employee/getLastEmployeeCode?role=Telecaller

Purpose: Auto generate employee code के लिए
Response: { lastCode: "TC001", roleCode: "TC" }

📍 EMPLOYEE LISTING APIs

API 3: All Telecallers Fetch

http

GET /api/telecaller

Purpose: सभी Telecaller employees लाना
Source: Telecaller मॉडल से
Response: { telecallers: [...] }

API 4: All HR Fetch

http

GET /api/hr

Purpose: सभी HR employees लाना
Source: HR मॉडल से
Response: { HRs: [...] }

API 5: All Employees Fetch

http

GET /api/employee/getAllEmployees

Purpose: सभी employees लाना (सभी roles के)
Source: Employee मॉडल से
Response: { success: true, data: [...] }

📍 EMPLOYEE OPERATIONS APIs

API 6: Employee Details Fetch

http

GET /api/employee/getEmployeeById?employeeId=123

Purpose: Single employee details लाना
Used in: View/Edit employee

API 7: Employee Update

http

PUT /api/employee/updateEmployee

Purpose: Employee update करना
Request Body: { employeeId, ...updates }

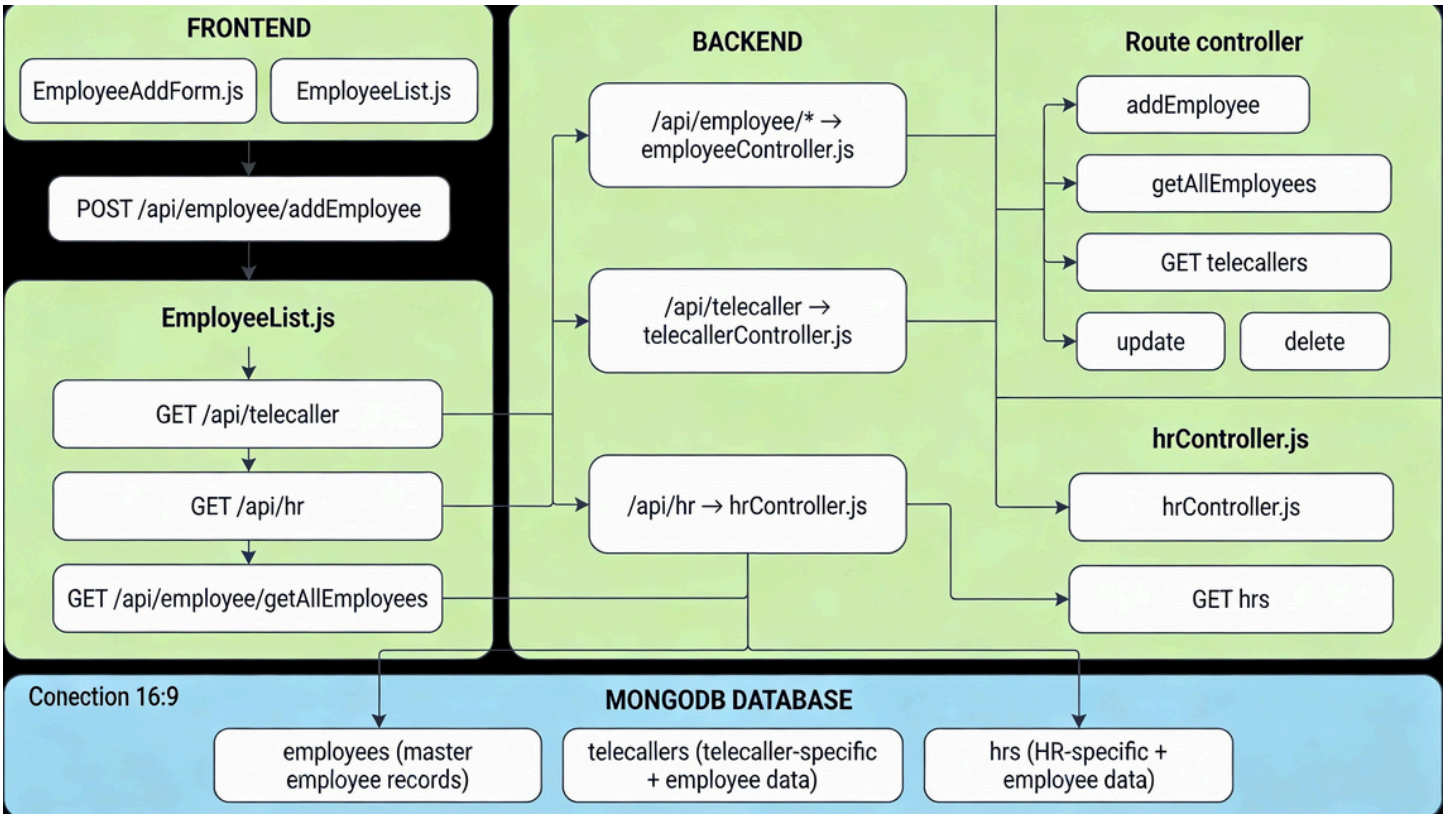
API 8: Employee Delete

http

DELETE /api/employee/deleteEmployee?employeeId=123

Purpose: Employee delete करना
Note: Role-specific delete भी हो सकता है

🔄 Complete Data Flow Diagram



📝 Step-by-Step Operations

1. Add New Employee

text

Step 1: User fills form in frontend

Step 2: Click "Add Employee" button

Step 3: POST /api/employee/addEmployee

Step 4: Backend:

- Employee मॉडल में save (primary record)
- Role check करो:
  - Telecaller → Telecaller मॉडल में auto-save
  - HR → HR मॉडल में auto-save
- Success response भेजो

Step 5: Frontend shows success message

2. View Employees by Role

text

Step 1: User clicks "Telecaller" tab

Step 2: fetchEmployeesByRole("telecaller") call होता है

Step 3: GET /api/telecaller API call

Step 4: Backend Telecaller मॉडल से data fetch करता है

Step 5: Frontend table में display करता है

3. Search & Filter

text

Step 1: User types in search box

Step 2: Frontend में filter होता है (no API call)

Step 3: Only matching employees show होते हैं