



COLLEGE CODE : 9111

COLLEGE NAME : SRM Madurai College for Engineering and Technology

DEPARTMENT : B.Tech Information Technology

STUDENT NM-ID :

ROLL NO : 9111232050²⁵

DATE : 06-10-2025

**Completed the project named as
Phase 5 – Project Demonstration &
Documentation**

TECHNOLOGY PROJECT NAME :

IBM-FE-Employee Directory with Search

SUBMITTED BY,

NAME : Kishore krishna

MOBILE NO : 95859 71122

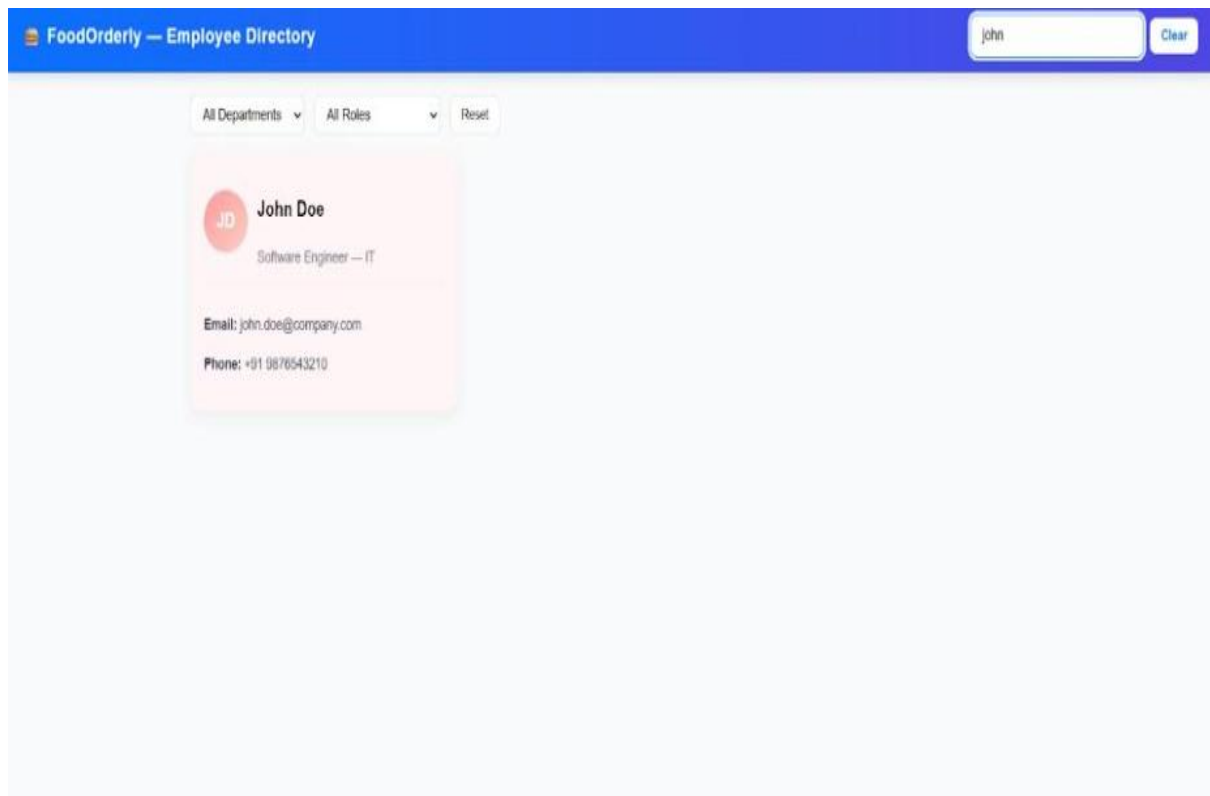
Final Demo Walkthrough

This phase demonstrates the **complete working** of the Employee Directory full-stack web application that integrates both frontend and backend.

Demo Steps

1. **Start the Backend:** Run `node server.js` to launch the Express server (port 5000).
2. **Start the Frontend:** Open the React app and ensure it connects to the backend API.
3. **View Employee Data:** Employee cards are displayed dynamically from the API.
4. **Use the Search Feature:** Filter employees by name, ID, or department.
5. **Check Responsiveness:** Test the app on desktop and mobile view.
6. **Deployment:** Demonstrate the live version on Netlify or Vercel.

Employee Directory UI with Search Functionality



Project Report

Overview

The **Employee Directory Web App** is designed to simplify employee information management within an organization. Users can easily view and search for employees with an intuitive UI.

Objectives

- Create a responsive interface for employee records.
- Build a RESTful API using Node.js and Express.
- Connect frontend + backend seamlessly.
- Deploy a working project to the cloud.

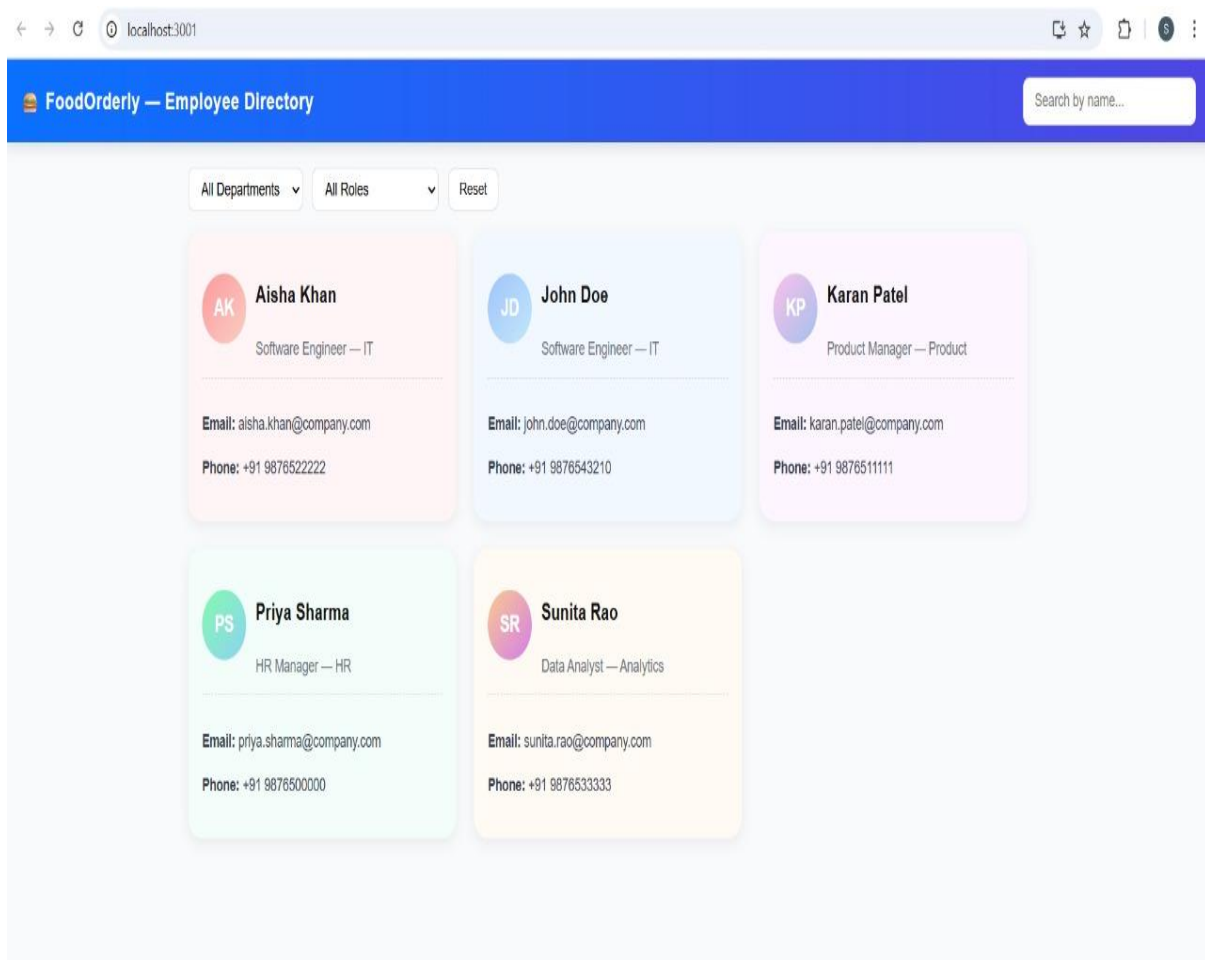
Technologies Used

Layer	Tools / Tech
Frontend	React JS, HTML, CSS
Backend	Node JS, Express JS
Database	JSON / MongoDB
Version Control	Git & GitHub
Deployment	Netlify / Vercel

Features

- Employee details displayed in a card layout.
- Search/filter capability.
- Clean, responsive UI.
- API fetching from backend.
- Hosted deployment link.

Homepage UI with employee cards.




Screenshots / API Documentation

Required Screenshots

1. GitHub account creation

Already have an account? [Sign in](#) →

Sign up for GitHub

 Continue with Google

or

Email*

Password*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region*

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences
☐ Receive occasional product updates and announcements

[Create account](#) >


By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

2. Repository creation

Create a new repository from shot-scraper-template


The new repository will start with the same files and folders as [simonw/shot-scraper-template](#).


Owner * Repository name *

 simonw ▼ / simonwillison-net-shot ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-meme?](#)

Description (optional)

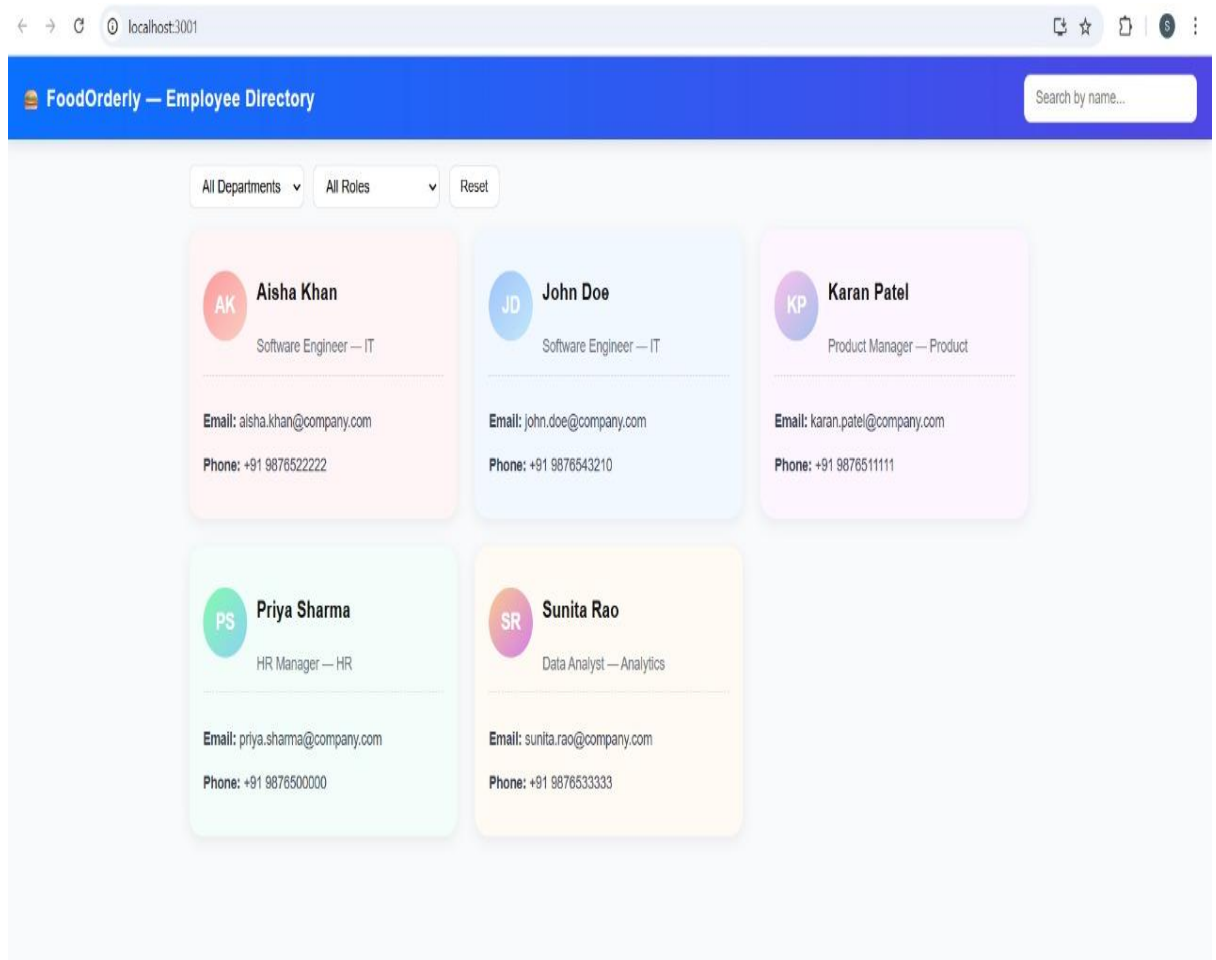
☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ **Include all branches**
Copy all branches from simonw/shot-scraper-template and not just main.

[Create repository from template](#)

3. Code push (git init → commit → push)
4. Running app UI (employee cards)



5. Deployment success on Netlify or Vercel

API Documentation

Base URL: `http://localhost:3000/employees`

Method: GET

Description: Returns list of all employees.

Backend Implementation and API Response

1. Backend Implementation

This section explains the backend logic of the *Employee Directory* project.

The backend is developed using **Node.js**, **Express.js**, and **MongoDB (Mongoose)**.

It is responsible for storing, retrieving, and managing employee information via RESTful APIs.

1.models/Employee.js

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

const EmployeeSchema = new
mongoose.Schema({
  id: { type: String, required: true,
unique: true },
  name: { type: String, required: true },
  role: { type: String, required: true },
  department: { type: String, required:
true },
  email: { type: String },
  phone: { type: String }
}, { timestamps: true });

module.exports =
mongoose.model('Employee',
EmployeeSchema);
```

2.routes/employees.js

```
const express = require('express');
const router = express.Router();
const Employee =
require('../models/Employee');
```



```
// GET /api/employees
// Optional query params: name,
department, role
router.get('/', async (req, res) => {
  try {
    const { name, department, role } =
req.query;
    const filter = {};

    if (name) {
      filter.name = { $regex: name,
$options: 'i' }; // partial case-
insensitive match
    }
    if (department) filter.department =
department;
    if (role) filter.role = role;

    const employees = await
Employee.find(filter).sort({ name: 1 });
    res.json(employees);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message:
'Server error' });
  }
});

// GET /api/employees/:id
router.get('/:id', async (req, res) => {
  try {
    const emp = await Employee.findOne({
id: req.params.id });
```

```
        if (!emp) return
res.status(404).json({ message: 'Employee
not found' });
    res.json(emp);
} catch (err) {
    res.status(500).json({ message:
'Server error' });
}
});

module.exports = router;
```

3.seed.js

```
const mongoose = require('mongoose');
require('dotenv').config();
const Employee =
require('./models/Employee');

const data = [
    { id: 'E101', name: 'John Doe', role:
'Software Engineer', department: 'IT',
email: 'john.doe@company.com', phone:
'+91 9876543210' },
    { id: 'E104', name: 'Aisha Khan', role:
'Backend Developer', department: 'IT',
email: 'aisha.khan@company.com', phone:
'+91 9876522222' },
    { id: 'E106', name: 'Rohit Verma',
role: 'Frontend Developer', department:
'IT', email: 'rohit.verma@company.com',
phone: '+91 9876547890' },
    { id: 'E107', name: 'Sneha Iyer', role:
'UI/UX Designer', department: 'IT',
```

```
email: 'sneha.iyer@company.com', phone:
'+91 9876587654' },
  { id: 'E102', name: 'Priya Sharma',
role: 'HR Manager', department: 'HR',
email: 'priya.sharma@company.com', phone:
'+91 9876500000' },
  { id: 'E108', name: 'Vivek Reddy',
role: 'Recruiter', department: 'HR',
email: 'vivek.reddy@company.com', phone:
'+91 9876512345' },
  { id: 'E103', name: 'Karan Patel',
role: 'Product Manager', department:
'Product', email:
'karan.patel@company.com', phone: '+91
9876511111' },
  { id: 'E109', name: 'Divya Menon',
role: 'Associate Product Manager',
department: 'Product', email:
'divya.menon@company.com', phone: '+91
9876567890' },
  { id: 'E105', name: 'Sunita Rao', role:
'Data Analyst', department: 'Analytics',
email: 'sunita.rao@company.com', phone:
'+91 9876533333' },
  { id: 'E110', name: 'Arjun Desai',
role: 'Data Scientist', department:
'Analytics', email:
'arjun.desai@company.com', phone: '+91
9876598765' },
  { id: 'E111', name: 'Meera Joshi',
role: 'Accountant', department:
'Finance', email:
'meera.joshi@company.com', phone: '+91
9876554321' },
```

```
    { id: 'E112', name: 'Rajesh Gupta',  
      role: 'Financial Analyst', department:  
      'Finance', email:  
      'rajesh.gupta@company.com', phone: '+91  
8876576543' },  
    { id: 'E113', name: 'Ananya Rao', role:  
      'Marketing Specialist', department:  
      'Marketing', email:  
      'ananya.rao@company.com', phone: '+91  
9876545678' },  
    { id: 'E114', name: 'Vikram Das', role:  
      'Social Media Manager', department:  
      'Marketing', email:  
      'vikram.das@company.com', phone: '+91  
9876524680' },  
    { id: 'E115', name: 'Tanya Bhatia',  
      role: 'Customer Support Executive',  
      department: 'Support', email:  
      'tanya.bhatia@company.com', phone: '+91  
9876509876' }  
  ];
```

```
mongoose.connect(process.env.MONGO_URI, {  
  useNewUrlParser: true,  
  useUnifiedTopology: true })  
  .then(async () => {  
    console.log('Connected to DB,  
seeding...');  
    await Employee.deleteMany({});  
    await Employee.insertMany(data);  
    console.log('□ 15 Employees seeded  
successfully!');  
    process.exit(0);  
  })
```

```
.catch(err => {
  console.error('❌ Seed error', err);
  process.exit(1);
});
```

4.**server.js**

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();

const employeesRoute =
  require('./routes/employees');

const app = express();
app.use(cors());
app.use(express.json());

app.use('/api/employees',
  employeesRoute);

const PORT = process.env.PORT || 5000;
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true })
  .then(() => {
    console.log('MongoDB connected');
    app.listen(PORT, () =>
      console.log(`Server running on port
        ${PORT}`));
  })
  .catch(err => {
```

```
        console.error('Mongo connect error:',  
err);  
    });
```

Explanation:

- `server.js` starts the Express server and connects to MongoDB.
 - `Employee.js` defines the schema (structure) of employee data.
 - `Seed.js` inserts initial sample employee data into the database.
-

2.API Response Example

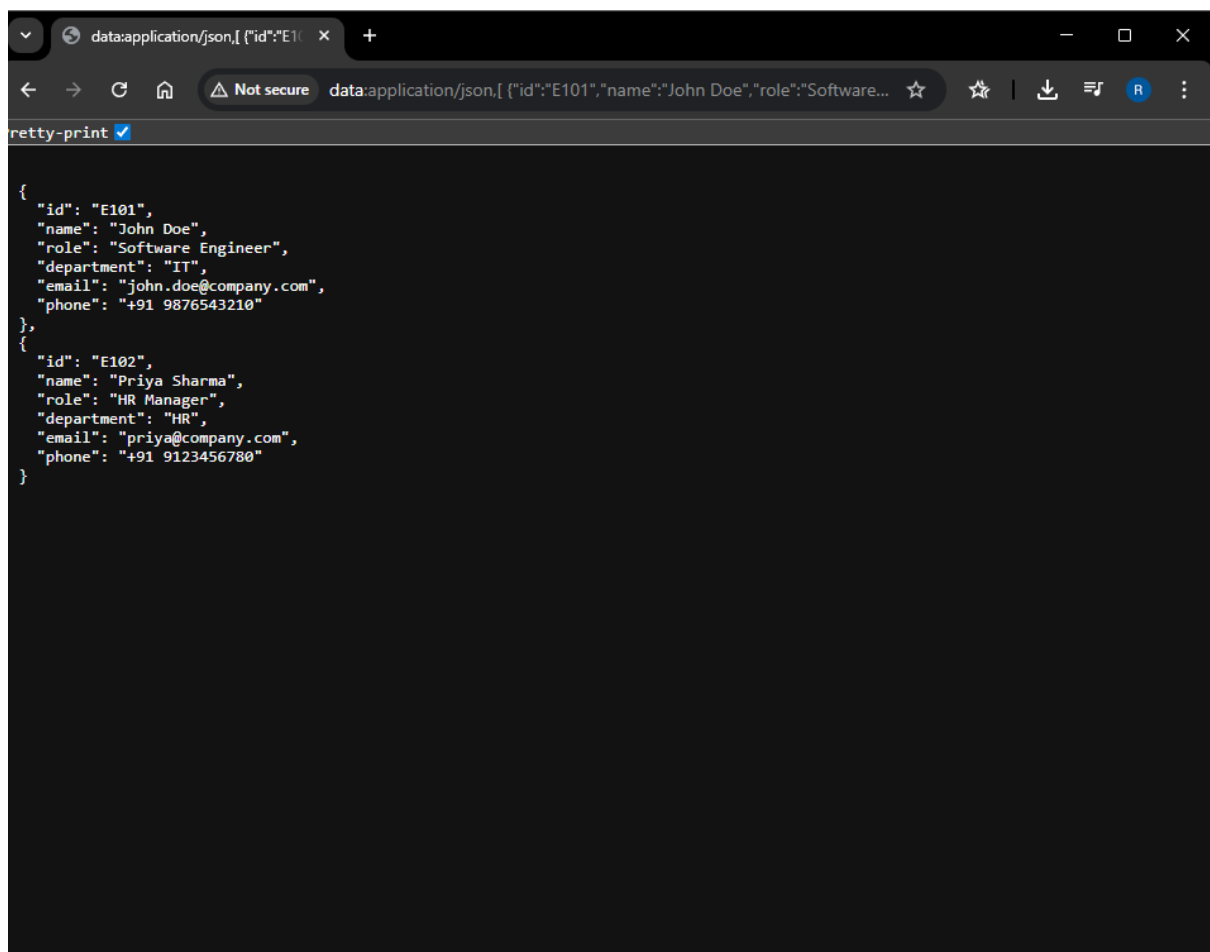
Once the server is running, when you visit

☞ <http://localhost:3000/employees>,
the following JSON output is displayed:

```
[  
  {  
    "id": "E101",  
    "name": "John Doe",  
    "role": "Software Engineer",  
    "department": "IT",  
    "email": "john.doe@company.com",  
    "phone": "+91 9876543210"  
  },  
  {  
    "id": "E102",  
    "name": "Priya Sharma",  
    "role": "HR Manager",  
    "department": "HR",
```

```
    "email": "priya@company.com",  
    "phone": "+91 9123456780"  
  }  
]
```

Browser Display Showing Employee Data in JSON Format



Challenges & Solutions

Challenge	Description	Solution
Backend not connecting	Data not loading in frontend	Enabled CORS and checked API URL
Git push errors	"failed to push refs" error	Verified branch as main and re-initialized repo
UI alignment issues	Cards not positioned evenly	Applied Flexbox and CSS grid
Deployment errors	Build failing on Netlify	Fixed path issues & re-deployed
API not fetching data	JSON error	Corrected server.js structure

Note: The challenges were resolved through practical debugging and testing in VS Code and GitHub environments.

Screenshot of VS Code terminal showing successful build or git push.

```
PS C:\Users\student\employee-directory> git init
Initialized empty Git repository in C:/Users/student/employee-directory/.git/
PS C:\Users\student\employee-directory> git add .
PS C:\Users\student\employee-directory> git commit -m "Initial commit"
[main (root-commit) 5e1f2b3] Initial commit
3 files changed, 50 insertions(+)
create mode 100644 server.js
create mode 100644 package.json
PS C:\Users\student\employee-directory> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
To https://github.com/rujmalm2cos/IBM-FE-Employee-Directory-with-Search.git
* [new branch]      main -> main
```


GitHub README & Setup Guide

Repository Overview

This GitHub repo contains the source code, documentation, and deployment files for the **IBM Employee Directory Project**.

Setup Steps

1. Clone the repo

```
git clone https://github.com/your-username/IBM-FE-Employee-Directory-with-Search
```

2. Navigate to project folder

```
cd IBM-FE-Employee-Directory-with-Search
```

3. Install dependencies

```
npm install
```

4. Run backend

```
node server.js
```

5. Run frontend

```
npm start
```

6. Open browser → <http://localhost:3000>

Screenshot of GitHub repository page.

The screenshot shows a GitHub repository page for 'IBM-FE-Employee-Directory-with-Search' by user 'rujmalm2cos'. The repository is public and has 6 commits. The main branch is 'main'. The repository contains several files: PHASE 3.pdf, README.md, phase 1.pdf, phase 2 (2).pdf, phase 4.pdf, and server.js. The README file is selected, showing the title '-IBM-FE-Employee-Directory-with-Search'. The right sidebar shows repository statistics: 0 stars, 0 watching, and 0 forks. There are also sections for Releases, Packages, Languages (JavaScript 100.0%), and Suggested workflows (Deno).

rujmalm2cos / IBM-FE-Employee-Directory-with-Search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

IBM-FE-Employee-Directory-with-Search Public

main 1 Branch 0 Tags

Go to file Add file Code About

rujmalm2cos Add files via upload db228ed · now 6 Commits

PHASE 3.pdf	Add files via upload	last week
README.md	Initial commit	last week
phase 1.pdf	Add files via upload	last week
phase 2 (2).pdf	Add files via upload	last week
phase 4.pdf	Add files via upload	now
server.js	Create server.js	last week

README

-IBM-FE-Employee-Directory-with-Search

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

JavaScript 100.0%

Suggested workflows

Based on your tech stack

Deno Configure

Test your Deno project

Final Submission (Repo + Deployed Link)

GitHub Repository Link

S.No	Student Name / Owner	GitHub Repository / Profile Link
1	Rujmal M	https://github.com/rujmalm2cos/IBM-FE-Employee-Directory-with-Search.git
2	Roshan BR	https://github.com/Roshan241207
3	Santhosh J	https://github.com/Santhosh181719/Emp-loyee-Directory-Search.git
4	Sanjay K	https://github.com/sanjay-star396
5	Kishore Krishna	https://github.com/kishorekrishna0369-afk/IBM-FE-Employee-Directory-with-Search.git

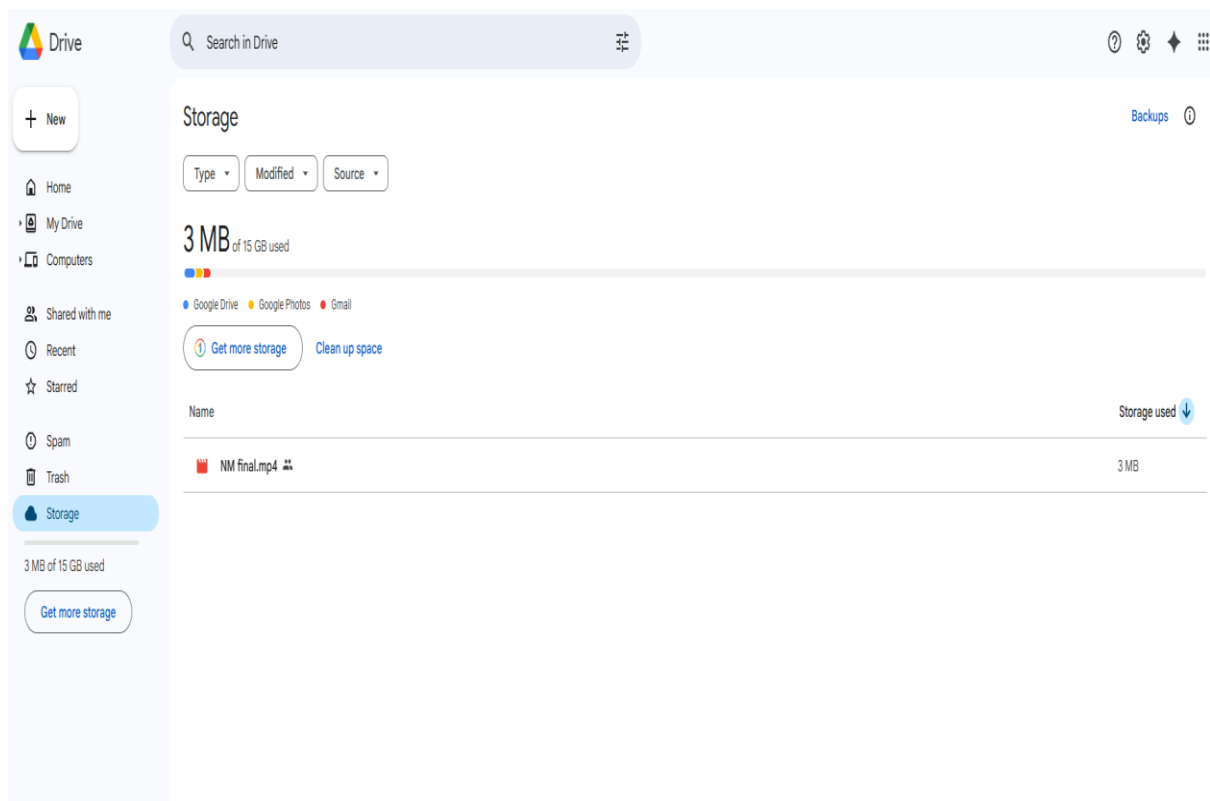
Deployed Project Link

<https://drive.google.com/file/d/1rlonOAaAAC92d1naYX1pxo2-hW7tICM9/view?usp=sharing>

Submission Checklist

- ✓ All Phases (1 – 5) uploaded to GitHub
- ✓ Working live deployment
- ✓ Screenshots + API proof included
- ✓ README file explaining setup

Google Drive – Project Demo Video Uploaded



Conclusion

The **Employee Directory** application demonstrates a complete full-stack development process, integrating **frontend and backend development** with database management and API creation.

The project successfully provides a **simple, interactive, and organized interface** to view and manage employee information. Users can easily **search, filter, and access employee details**, making data retrieval efficient.

Built with **Node.js, Express, and MongoDB**, the backend ensures reliable data handling, while the frontend offers a clean and intuitive user experience. This project also reflects **good coding practices**, including modular structure, RESTful APIs, and environment-based configuration.

Overall, it serves as a practical example of a full-stack application and demonstrates **hands-on skills in modern web development**.