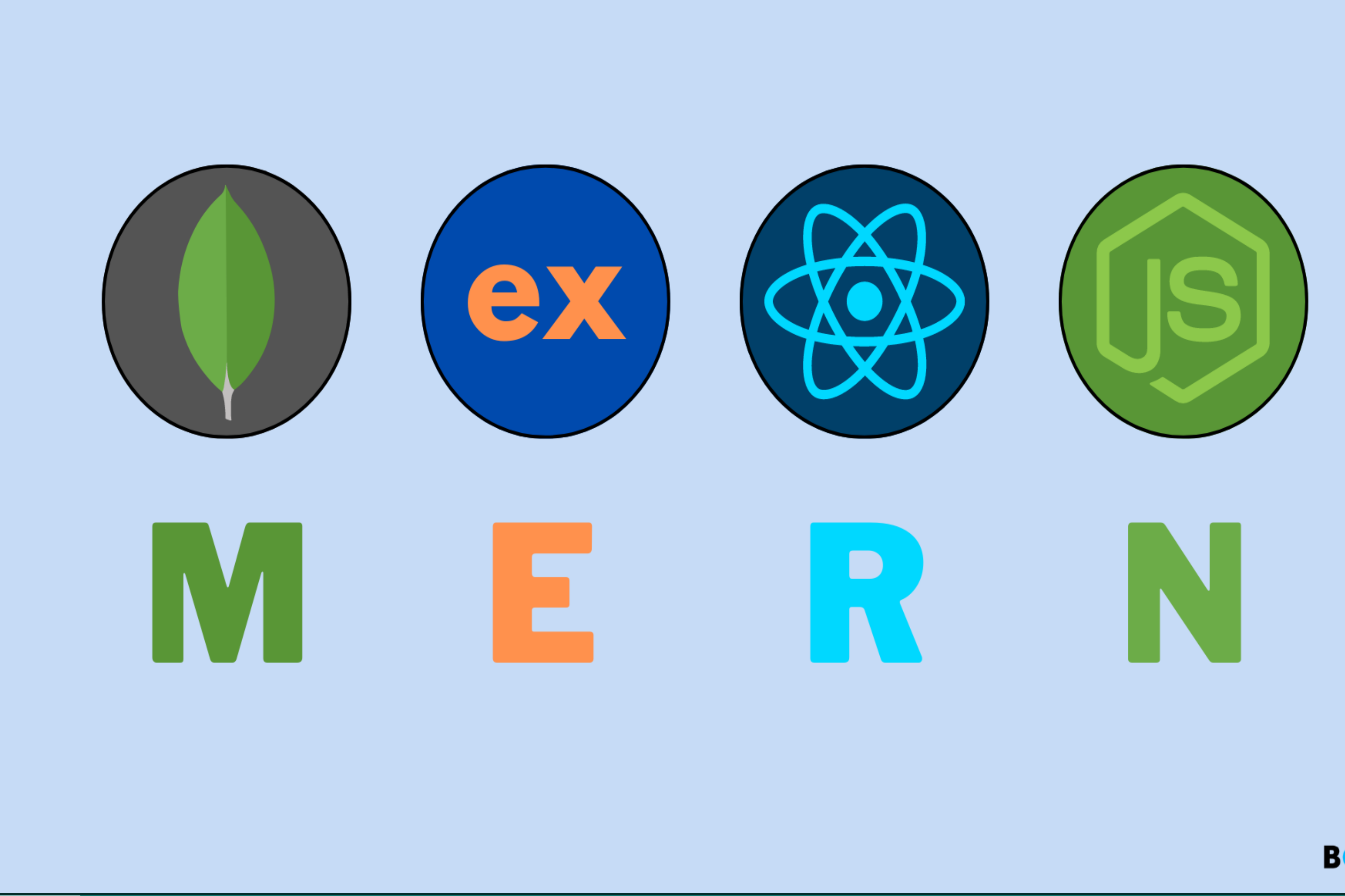


MERN Curd Application Report

Employee Management Web Application

By M.M. S. Hansaja, SLTC Research University,01-05-2024

Introduction

In today's dynamic business landscape, the effective management of human resources stands as a cornerstone for organizational success. The ability to streamline processes, optimize workflows, and nurture a productive workforce is paramount in achieving strategic objectives and maintaining a competitive edge. Recognizing this imperative, the development of the Employee Management Web Application emerges as a pivotal solution. By harnessing the power of modern technology, this platform offers a comprehensive suite of tools tailored to meet the diverse needs of HR departments. From simplifying recruitment processes to facilitating performance evaluations and fostering a collaborative work environment, this application serves as a catalyst for driving efficiency and empowering businesses to thrive in a rapidly evolving marketplace.

Built upon a foundation of cutting-edge technologies and industry best practices, the Employee Management Web Application represents a paradigm shift in how organizations approach HR management. Leveraging the scalability and flexibility of Node.js, Express.js, MongoDB, and Docker, this platform delivers unparalleled performance and reliability. From startups to multinational corporations, businesses of all sizes can harness the power of this application to drive operational efficiency, enhance employee satisfaction, and ultimately, achieve sustainable growth. As the business landscape continues to evolve, the Employee Management Web Application stands as a testament to the transformative potential of technology in shaping the future of HR management.

Technologies Used

1. Node.js

Node.js is a powerful, server-side JavaScript runtime environment that enables the execution of JavaScript code outside of a web browser. With its event-driven architecture and non-blocking I/O operations, Node.js allows for the development of highly scalable and efficient web applications. In the Employee Management Web Application, Node.js serves as the foundation for the server-side logic, facilitating seamless communication between the client and the database.

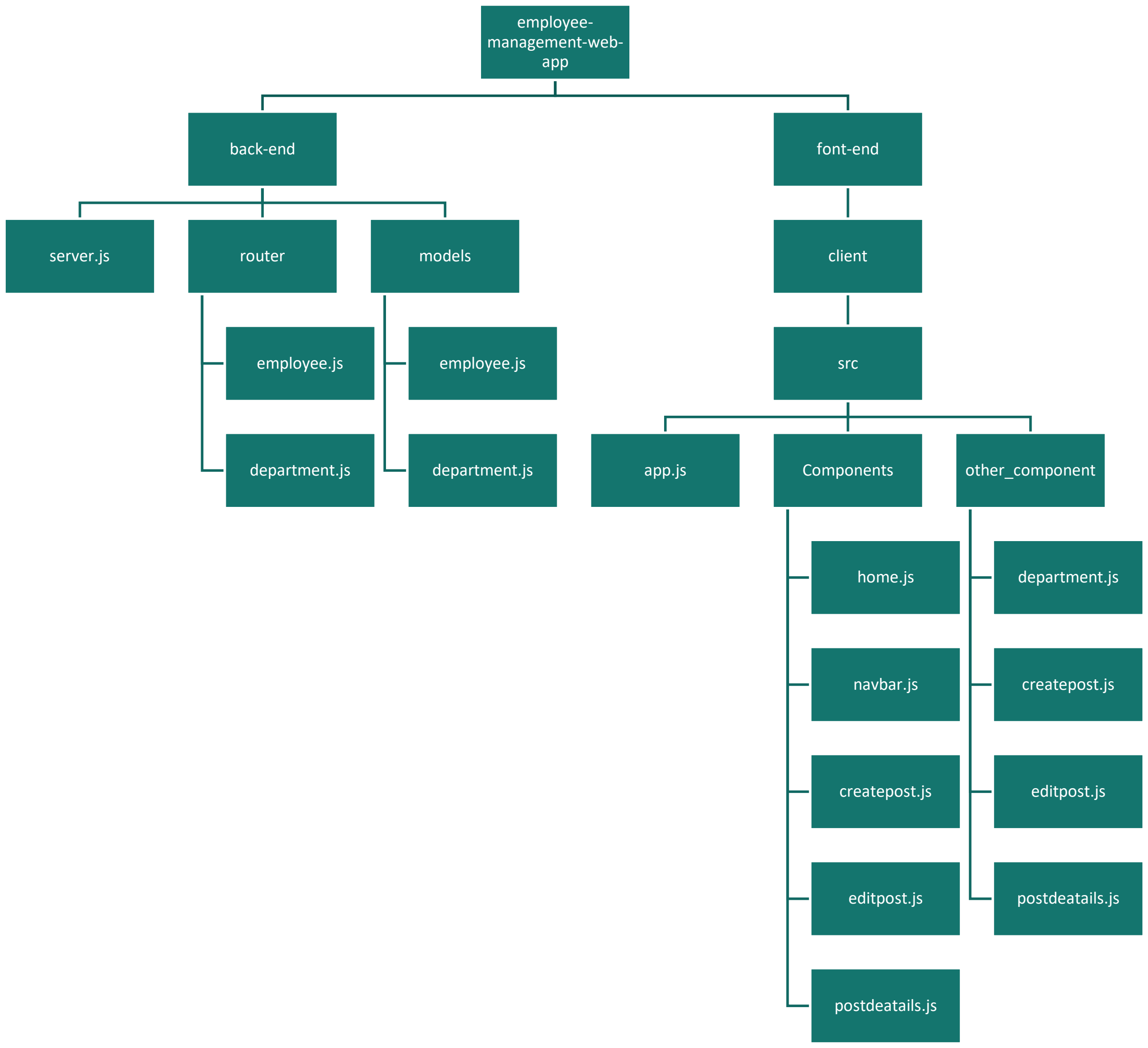
2. Express.js

Express.js is a minimalist and flexible Node.js web application framework that provides a robust set of features for building web applications and APIs. By simplifying the process of defining routes, handling HTTP requests, and managing middleware, Express.js accelerates the development process and enhances code maintainability. In the Employee Management Web Application, Express.js acts as the middleware layer, enabling the implementation of RESTful APIs for CRUD operations on employee and department data.

3. MongoDB

MongoDB is a NoSQL database that offers a flexible and scalable approach to data storage and retrieval. With its document-oriented architecture and dynamic schema, MongoDB is well-suited for managing unstructured or semi-structured data, such as employee and department records. In the Employee Management Web Application, MongoDB serves as the backend database, storing and managing the persistent data required for employee and department management.

Project Structure



- *models: Defines MongoDB schemas and models for employees and departments.*
- *routes: Defines API routes for CRUD operations on employees and departments.*
- *.Env : difines PORT number.*

Section Title

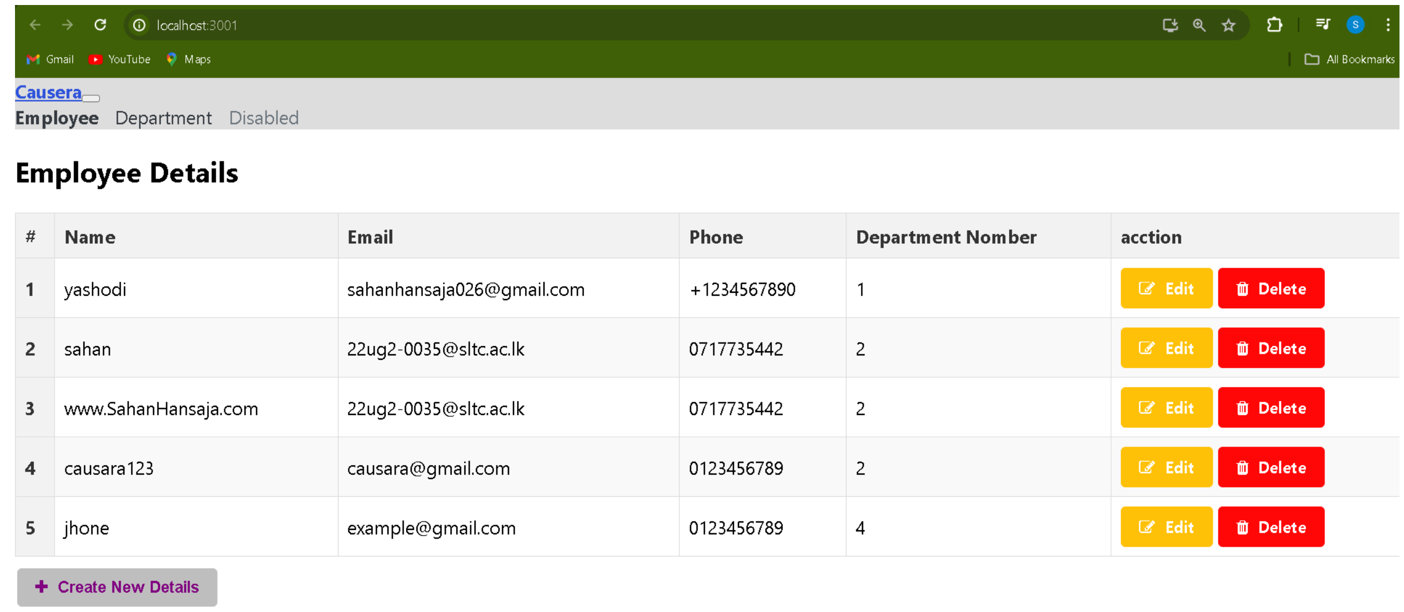
employee-management-web-app	employee	name
		email
		phone
		address
		salary
		department_id
		hire_date
		date_of_birth
	department	department_name
		department_id
		location
		department_head
		description
		email
		operation_hours
		total_number_of_employee

Employee Schema

- ❖ The employee schema defines the structure of employee data stored in the MongoDB database. It includes fields

Department Schema

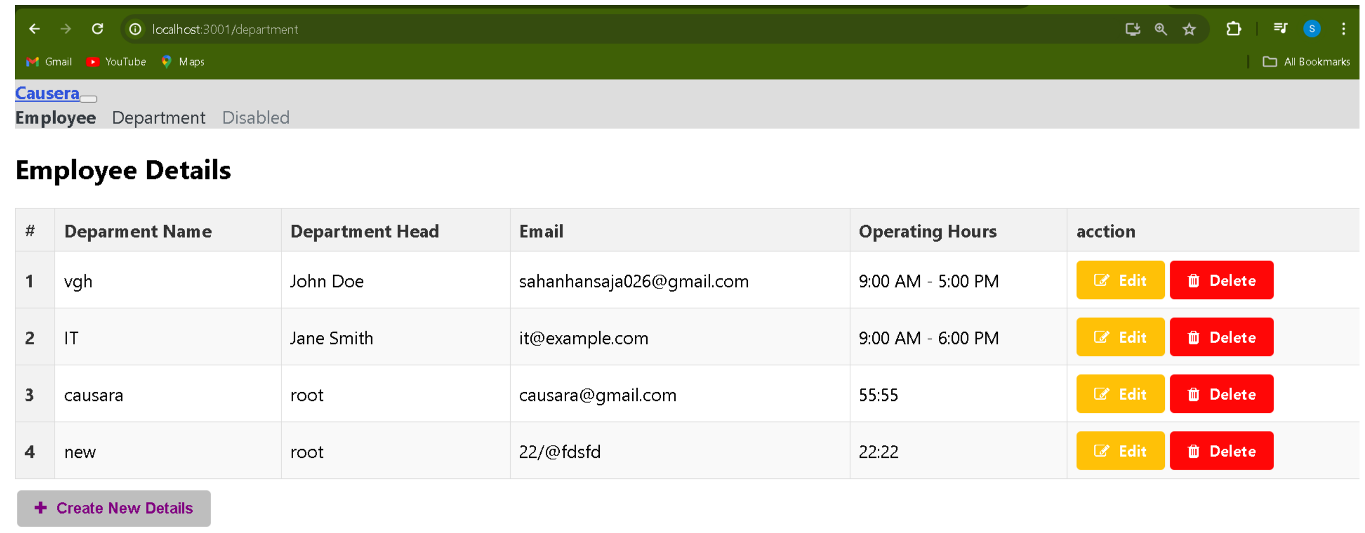
- ❖ The department schema defines the structure of department data stored in the MongoDB database. It includes fields



Employee Details

#	Name	Email	Phone	Department Nomber	acction
1	yashodi	sahanhansaja026@gmail.com	+1234567890	1	Edit Delete
2	sahan	22ug2-0035@sltc.ac.lk	0717735442	2	Edit Delete
3	www.SahanHansaja.com	22ug2-0035@sltc.ac.lk	0717735442	2	Edit Delete
4	causara123	causara@gmail.com	0123456789	2	Edit Delete
5	jhone	example@gmail.com	0123456789	4	Edit Delete

Figer:- 01 font-end of the employee details



Employee Details

#	Deparment Name	Department Head	Email	Operating Hours	acction
1	vgh	John Doe	sahanhansaja026@gmail.com	9:00 AM - 5:00 PM	Edit Delete
2	IT	Jane Smith	it@example.com	9:00 AM - 6:00 PM	Edit Delete
3	causara	root	causara@gmail.com	55:55	Edit Delete
4	new	root	22/@fdsfd	22:22	Edit Delete

Figer:- 02 font-end of the department details

Features Implemented

1. CRUD Operations

Create: Users can add new employees and departments to the system by filling out dedicated forms. Upon submission, the data is validated and stored in the MongoDB database.

Read: The application allows users to view a list of all employees and departments currently stored in the database. This information is presented in a user-friendly format, facilitating easy navigation and reference.

Update: Users can edit existing employee and department details, such as their name, email, position, or description, through intuitive edit forms. Any changes made are promptly reflected in the database.

Delete: The application supports the deletion of employees and departments. Users can remove obsolete records from the system with a simple click, ensuring data cleanliness and integrity.

2. Web UI

Employee Management: The web interface provides dedicated pages for managing employees, including forms for adding/editing employees and a list view for browsing existing records. The UI is designed to be intuitive and responsive, catering to users of all levels of technical proficiency.

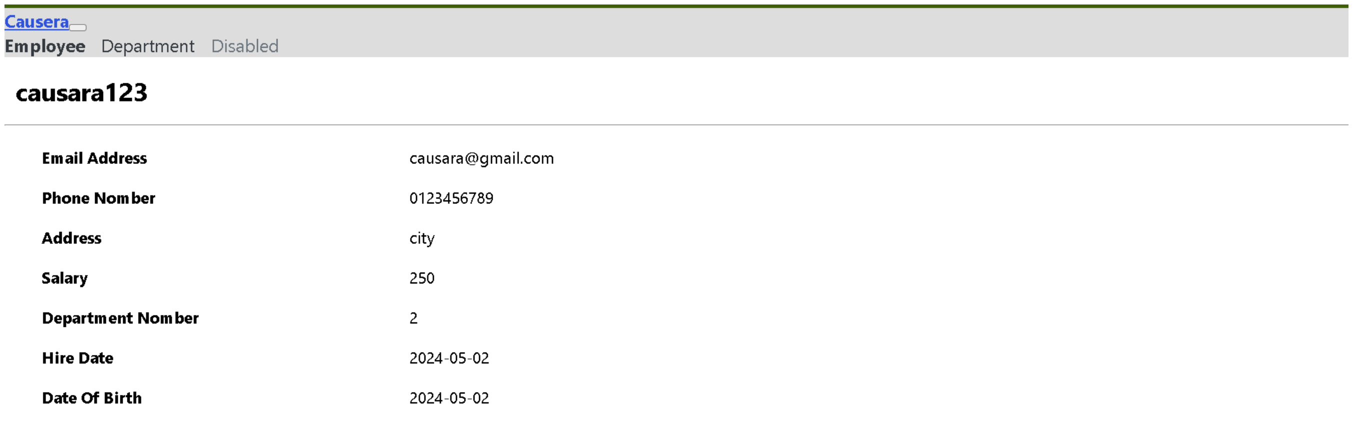
Department Management: Similarly, users can manage departments through dedicated forms for adding/editing departments and a list view for viewing department details. The UI maintains consistency with the employee management section, ensuring a cohesive user experience.

Navigation: The application features a streamlined navigation system, with clear and intuitive menus guiding users to the desired functionality. Users can easily switch between employee and department management sections, facilitating efficient workflow.

3. Error Handling and Validation

Input Validation: All user inputs are thoroughly validated on the client and server sides to prevent invalid or malicious data from being submitted. Users are prompted with informative error messages if any validation rules are violated, ensuring data integrity.

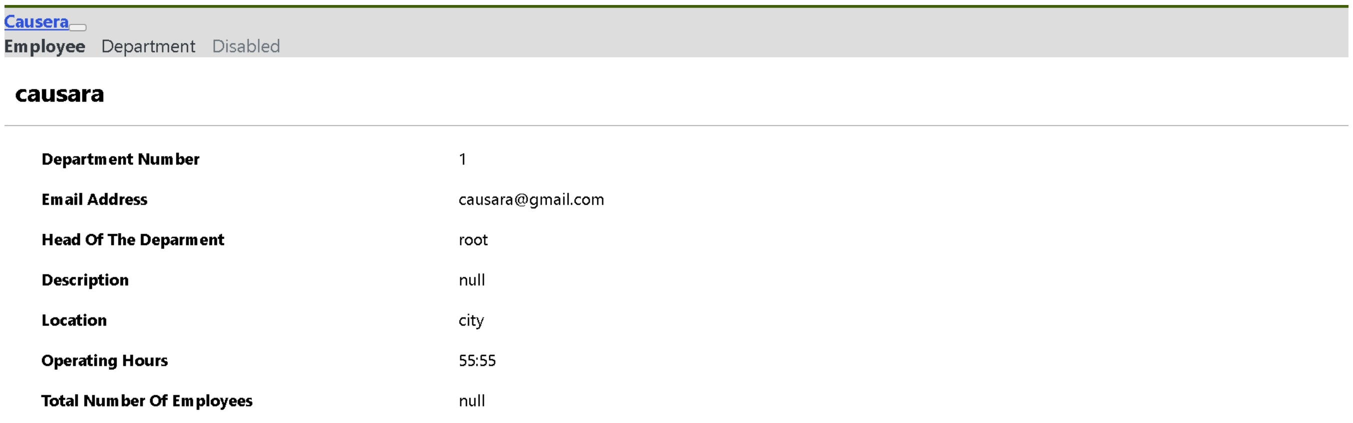
Error Handling: Robust error handling mechanisms are in place to gracefully handle unexpected errors or exceptions during the execution of CRUD operations.



causara123

Email Address	causara@gmail.com
Phone Nomber	0123456789
Address	city
Salary	250
Department Nomber	2
Hire Date	2024-05-02
Date Of Birth	2024-05-02

Figer:- 03 font-end of Selected Special Employee Details



causara

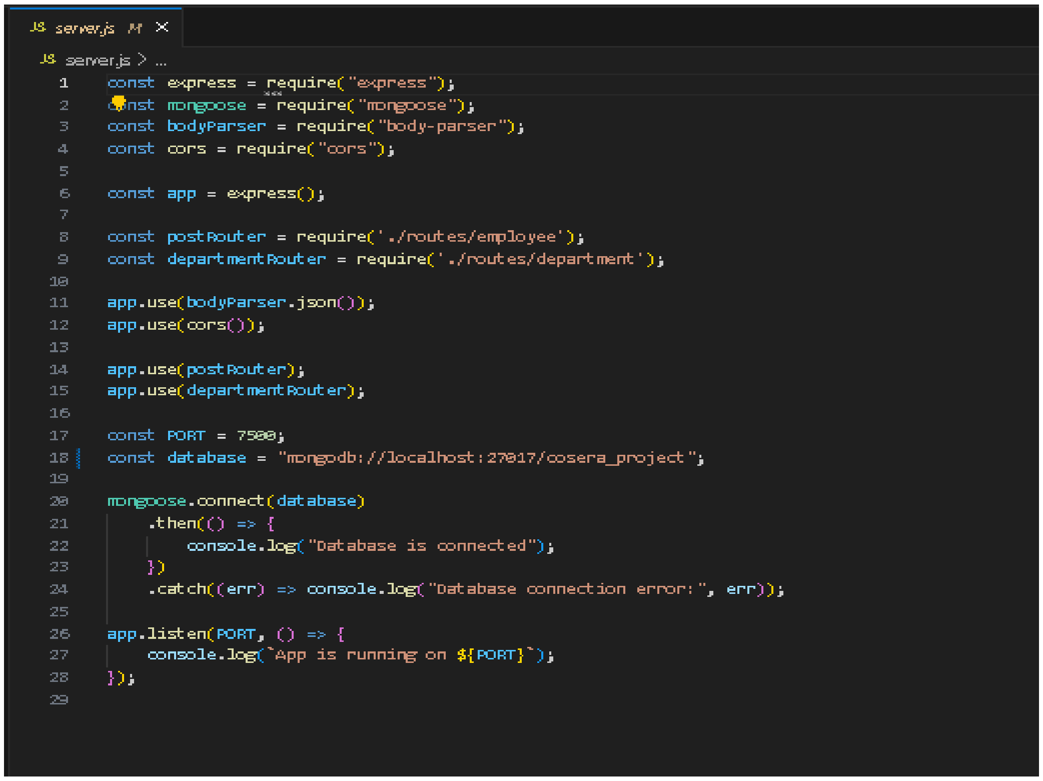
Department Number	1
Email Address	causara@gmail.com
Head Of The Deparment	root
Description	null
Location	city
Operating Hours	55:55
Total Number Of Employees	null

Figer:- 04 font-end of Selected Special Department Details

Code Highlights

Back-End code

❖Server



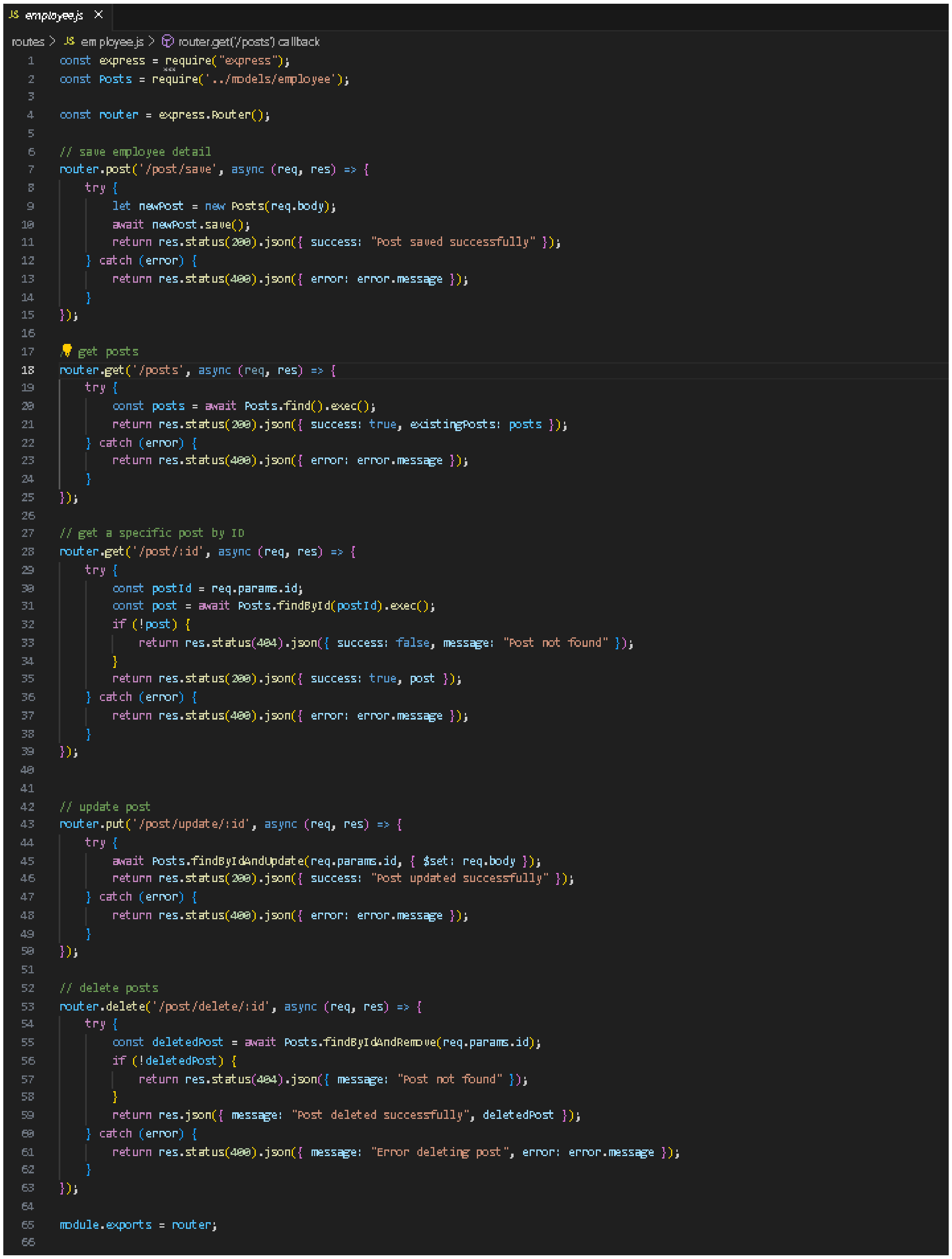
server.js M ×

server.js > ...

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const cors = require("cors");

const app = express();

const postRouter = require('./routes/employee');
const departmentRouter = require('./routes/department');

app.use(bodyParser.json());
app.use(cors());

app.use(postRouter);
app.use(departmentRouter);

const PORT = 7500;
const database = "mongodb://localhost:27017/cosera_project";

mongoose.connect(database)
    .then(() => {
        console.log("Database is connected");
    })
    .catch((err) => console.log("Database connection error:", err));

app.listen(PORT, () => {
    console.log(`App is running on ${PORT}`);
});
```

Figer:- 05 back-end server.js code

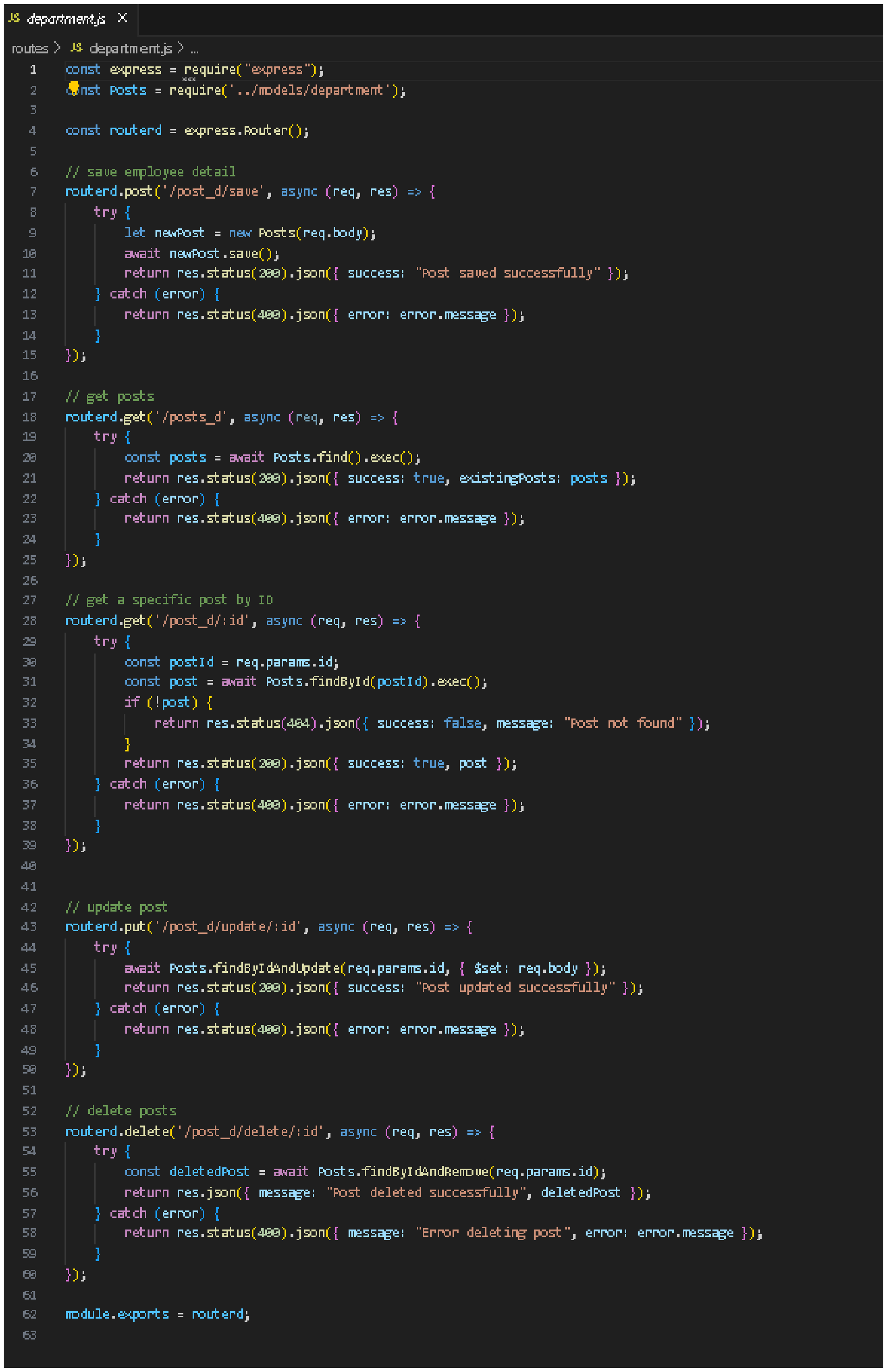
❖Routes


employee.js ×

routes > employee.js > router.get('/posts') callback

```javascript
const express = require("express");
const Posts = require('../models/employee');

const router = express.Router();

// save employee detail
router.post('/post/save', async (req, res) => {
    try {
        let newPost = new Posts(req.body);
        await newPost.save();
        return res.status(200).json({ success: "Post saved successfully" });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// get posts
router.get('/posts', async (req, res) => {
    try {
        const posts = await Posts.find().exec();
        return res.status(200).json({ success: true, existingPosts: posts });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// get a specific post by ID
router.get('/post/:id', async (req, res) => {
    try {
        const postId = req.params.id;
        const post = await Posts.findById(postId).exec();
        if (!post) {
            return res.status(404).json({ success: false, message: "Post not found" });
        }
        return res.status(200).json({ success: true, post });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});


// update post
router.put('/post/update/:id', async (req, res) => {
    try {
        await Posts.findByIdAndUpdate(req.params.id, { $set: req.body });
        return res.status(200).json({ success: "Post updated successfully" });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// delete posts
router.delete('/post/delete/:id', async (req, res) => {
    try {
        const deletedPost = await Posts.findByIdAndRemove(req.params.id);
        if (!deletedPost) {
            return res.status(404).json({ message: "Post not found" });
        }
        return res.json({ message: "Post deleted successfully", deletedPost });
    } catch (error) {
        return res.status(400).json({ message: "Error deleting post", error: error.message });
    }
});

module.exports = router;
```

Figer:- 06 (01)back-end routes , employee.js code

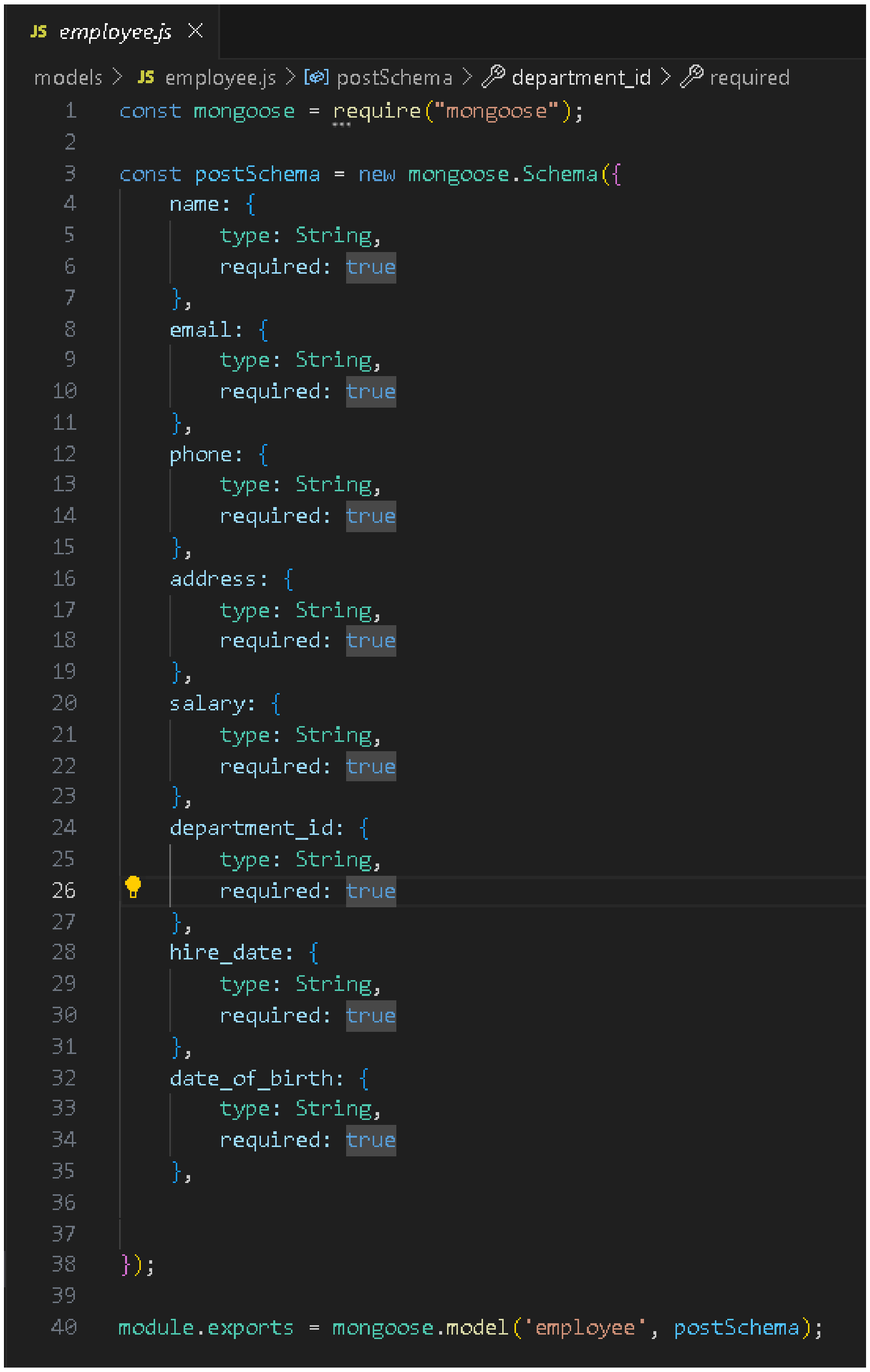

department.js

routes > department.js > ...

```
const express = require("express");
const Posts = require('../models/department');

const routerd = express.Router();

// save employee detail
routerd.post('/post_d/save', async (req, res) => {
    try {
        let newPost = new Posts(req.body);
        await newPost.save();
        return res.status(200).json({ success: "Post saved successfully" });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// get posts
routerd.get('/posts_d', async (req, res) => {
    try {
        const posts = await Posts.find().exec();
        return res.status(200).json({ success: true, existingPosts: posts });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// get a specific post by ID
routerd.get('/post_d/:id', async (req, res) => {
    try {
        const postId = req.params.id;
        const post = await Posts.findById(postId).exec();
        if (!post) {
            return res.status(404).json({ success: false, message: "Post not found" });
        }
        return res.status(200).json({ success: true, post });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});


// update post
routerd.put('/post_d/update/:id', async (req, res) => {
    try {
        await Posts.findByIdAndUpdate(req.params.id, { $set: req.body });
        return res.status(200).json({ success: "Post updated successfully" });
    } catch (error) {
        return res.status(400).json({ error: error.message });
    }
});

// delete posts
routerd.delete('/post_d/delete/:id', async (req, res) => {
    try {
        const deletedPost = await Posts.findByIdAndRemove(req.params.id);
        return res.json({ message: "Post deleted successfully", deletedPost });
    } catch (error) {
        return res.status(400).json({ message: "Error deleting post", error: error.message });
    }
});

module.exports = routerd;
```

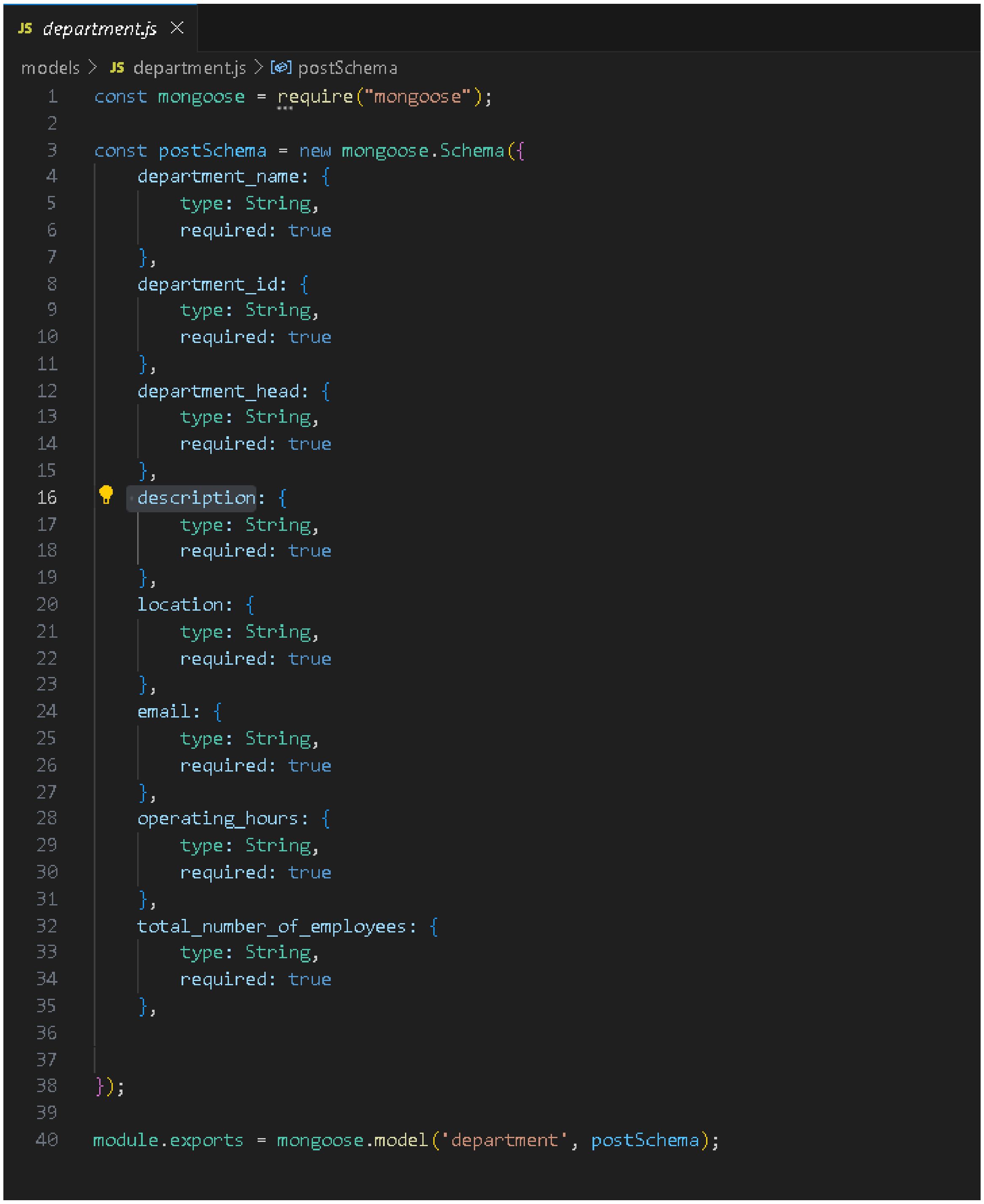
Figer:- 06 (02)back-end routes , department.js code

❖Models


employee.js
models > employee.js > postSchema > department_id > required

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

const postSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    phone: {
        type: String,
        required: true
    },
    address: {
        type: String,
        required: true
    },
    salary: {
        type: String,
        required: true
    },
    department_id: {
        type: String,
        required: true
    },
    hire_date: {
        type: String,
        required: true
    },
    date_of_birth: {
        type: String,
        required: true
    },


});

module.exports = mongoose.model('employee', postSchema);
```

Figer:- 07 (01)back-end models , employee.js code



JS department.js ×

models > JS department.js > postSchema

```javascript
const mongoose = require("mongoose");

const postSchema = new mongoose.Schema({
    department_name: {
        type: String,
        required: true
    },
    department_id: {
        type: String,
        required: true
    },
    department_head: {
        type: String,
        required: true
    },
    description: {
        type: String,
        required: true
    },
    location: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    operating_hours: {
        type: String,
        required: true
    },
    total_number_of_employees: {
        type: String,
        required: true
    },


});

module.exports = mongoose.model('department', postSchema);
```

Figer:- 07 (02)back-end models , department.js code

Conclusion

The development of the Employee Management Web Application marks a significant milestone in addressing the crucial need for efficient human resources management in today's dynamic business environment. Through the utilization of modern technologies and best practices, this project has succeeded in delivering a comprehensive solution for managing employees and departments seamlessly.

Impact on HR Management

By streamlining essential tasks such as recruitment, performance evaluation, and departmental organization, the application empowers organizations to optimize their HR processes and foster a more productive work environment. The ability to perform CRUD operations on employee and department data with ease facilitates better decision-making and resource allocation, ultimately driving organizational success.

Lessons Learned

Throughout the development process, valuable lessons were learned regarding software design, database management, and user interface development. Challenges were encountered and overcome, leading to a deeper understanding of best practices and potential pitfalls in web application development. These experiences will serve as valuable insights for future projects and endeavors.

Future Directions

While the Employee Management Web Application represents a significant achievement, there is always room for improvement and expansion. Future iterations of the application could include additional features such as advanced reporting and analytics, integration with third-party HR systems, and enhanced security measures. Moreover, ongoing maintenance and updates will be essential to ensure the application remains relevant and effective in meeting evolving business needs.

In conclusion, the Employee Management Web Application stands as a testament to the transformative potential of technology in revolutionizing HR management practices. By leveraging innovation and embracing continuous improvement, organizations can harness the power of digital tools to drive operational efficiency, enhance employee engagement, and achieve sustainable growth in today's competitive landscape.

Thank you

sahan hansaja portfolio websie