

|  |  |                                    |
|--|--|------------------------------------|
|  <b>Marwadi University</b><br>Marwadi Chandarana Group  | <b>Marwadi University</b><br><b>Faculty of Technology</b><br><b>Department of Information and Communication Technology</b> |                                    |
| <b>Subject: Advanced Web Technology</b>  | <b>Aim:</b> Create a RESTful service with Node, Express and MongoDB  |                                    |
| <b>Experiment :- 6</b>   | <b>Date:- 10-03-2025</b>   | <b>Enrollment No:- 92200133003</b> |

**Objective :** Create a RESTful service with Node, Express and MongoDB

#### CODE:

##### Model

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

const employeeSchema = new mongoose.Schema({
  name: { type: String, required: true },
  position: String,
  salary: Number,
  email: { type: String, unique: true }
});

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

##### Routes

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

// Create a new employee
router.post('/', async (req, res) => {
  try {
    const employee = new Employee(req.body);
    await employee.save();
    res.status(201).json(employee);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Get all employees
router.get('/', async (req, res) => {
  const employees = await Employee.find();
  res.json(employees);
});

// Get a specific employee
router.get('/:id', async (req, res) => {
  try {
    const employee = await Employee.findById(req.params.id);
    if (!employee) return res.status(404).json({ error: 'Not found' });
    res.json(employee);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Update an employee
router.put('/:id', async (req, res) => {
  try {
    const employee = await Employee.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(employee);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Delete an employee
router.delete('/:id', async (req, res) => {
  try {
    await Employee.findByIdAndDelete(req.params.id);
    res.json({ message: 'Employee deleted' });
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});
```

|  |  |                                    |
|--|--|------------------------------------|
|  <b>Marwadi University</b><br>Marwadi Chandarana Group  | <b>Marwadi University</b><br><b>Faculty of Technology</b><br><b>Department of Information and Communication Technology</b> |                                    |
| <b>Subject: Advanced Web Technology</b>  | <b>Aim:</b> Create a RESTful service with Node, Express and MongoDB  |                                    |
| <b>Experiment :- 6</b>   | <b>Date:- 10-03-2025</b>   | <b>Enrollment No:- 92200133003</b> |

```

    } catch (err) {
      res.status(400).json({ error: err.message });
    }
  });

module.exports = router;

```

### Server

```

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

const app = express();
const PORT = 3001;

// Middleware
app.use(bodyParser.json());

// MongoDB connection
mongoose.connect("mongodb://127.0.0.1:27017/employeeDb")
  .then(() => console.log("✔ Connected to MongoDB"))
  .catch((err) => console.error("✗ MongoDB connection error:", err));

// Schema
const employeeSchema = new mongoose.Schema({
  name: { type: String, required: true },
  position: String,
  salary: Number,
  email: String
});

// Model
const Employee = mongoose.model("Employee", employeeSchema);

// Routes

// POST: Add new employee(s)
app.post("/api/employees", async (req, res) => {
  try {
    const data = req.body;

    const result = Array.isArray(data)
      ? await Employee.insertMany(data)
      : await new Employee(data).save();

    res.status(201).json(result);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

// GET: All employees
app.get("/api/employees", async (req, res) => {
  try {
    const employees = await Employee.find();
    res.json(employees);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// GET: Single employee
app.get("/api/employees/:id", async (req, res) => {
  try {
    const employee = await Employee.findById(req.params.id);
    if (!employee) return res.status(404).json({ message: "Employee not found" });
    res.json(employee);
  } catch (error) {

```

|  |  |                                    |
|--|--|------------------------------------|
|  <b>Marwadi University</b><br>Marwadi Chandarana Group  | <b>Marwadi University</b><br><b>Faculty of Technology</b><br><b>Department of Information and Communication Technology</b> |                                    |
| <b>Subject: Advanced Web Technology</b>  | <b>Aim:</b> Create a RESTful service with Node, Express and MongoDB  |                                    |
| <b>Experiment :- 6</b>   | <b>Date:- 10-03-2025</b>   | <b>Enrollment No:- 92200133003</b> |

```

res.status(500).json({ error: error.message });
}
});

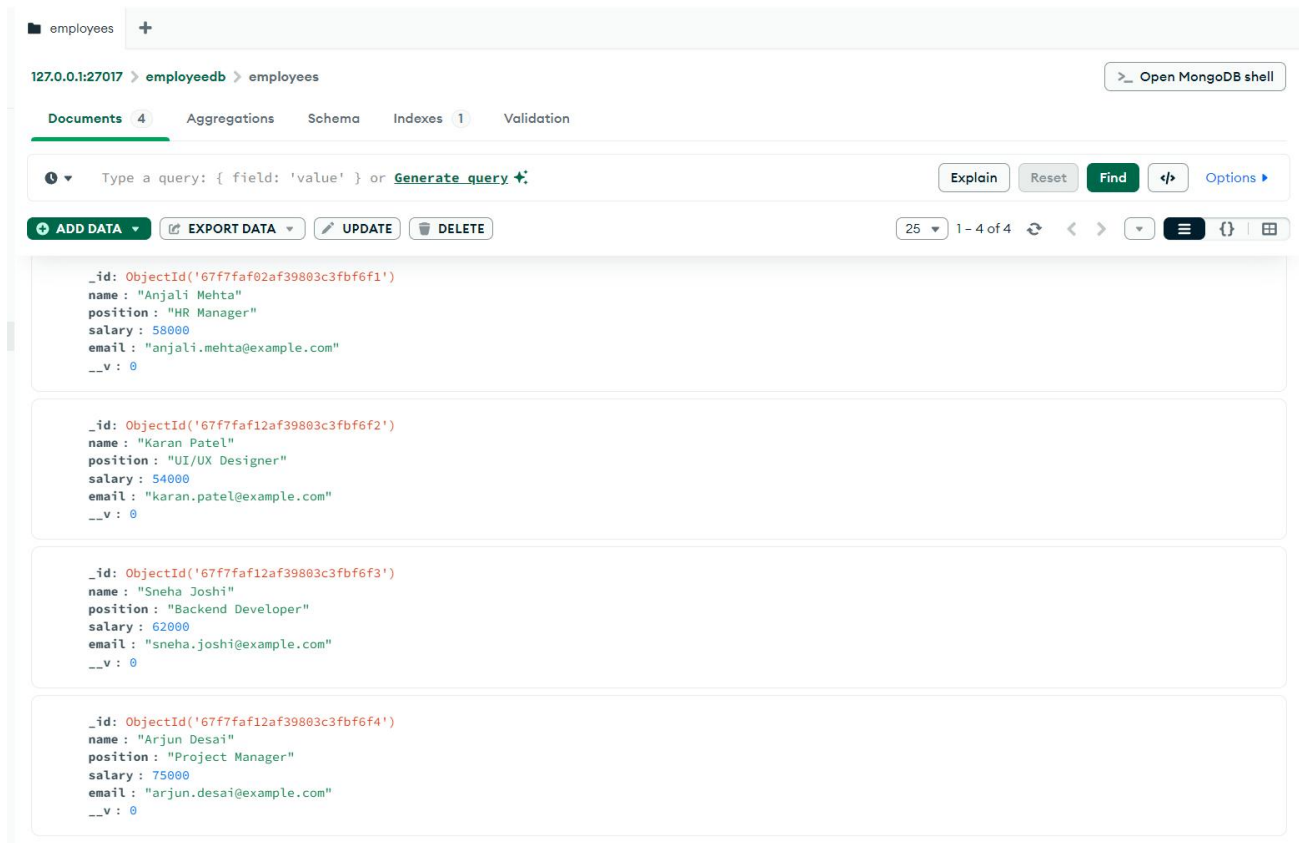
// PUT: Update employee
app.put("/api/employees/:id", async (req, res) => {
  try {
    const updated = await Employee.findByIdAndUpdate(req.params.id, req.body, { new: true });
    if (!updated) return res.status(404).json({ message: "Employee not found" });
    res.json(updated);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// DELETE: Remove employee
app.delete("/api/employees/:id", async (req, res) => {
  try {
    const deleted = await Employee.findByIdAndDelete(req.params.id);
    if (!deleted) return res.status(404).json({ message: "Employee not found" });
    res.json({ message: "Employee deleted" });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Start server
app.listen(PORT, () => {
  console.log(` Server running at http://localhost:${PORT}`);
});

```

## OUTPUT :



The screenshot shows the MongoDB Compass interface for the 'employees' collection. The interface includes a top navigation bar with 'employees' and a '+' icon. Below this, the address bar shows '127.0.0.1:27017 > employeeedb > employees' and a button to 'Open MongoDB shell'. The main area has tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. The 'Documents' tab is active, showing a list of 4 documents. Each document is displayed in a card format with its JSON structure, including fields like '\_id', 'name', 'position', 'salary', and 'email'.

| Document ID                          | name         | position          | salary | email                    |
|--------------------------------------|--------------|-------------------|--------|--------------------------|
| ObjectId('67f7faf02af39803c3fbf6f1') | Anjali Mehta | HR Manager        | 58000  | anjali.mehta@example.com |
| ObjectId('67f7faf12af39803c3fbf6f2') | Karan Patel  | UI/UX Designer    | 54000  | karan.patel@example.com  |
| ObjectId('67f7faf12af39803c3fbf6f3') | Sneha Joshi  | Backend Developer | 62000  | sneha.joshi@example.com  |
| ObjectId('67f7faf12af39803c3fbf6f4') | Arjun Desai  | Project Manager   | 75000  | arjun.desai@example.com  |

|   |  |  |  |
|---|--|--|--|
|  <b>Marwadi University</b><br>Marwadi Chandarana Group |  | <b>Marwadi University</b><br><b>Faculty of Technology</b><br><b>Department of Information and Communication Technology</b> |  |
| <b>Subject: Advanced Web Technology</b>   | <b>Aim:</b> Create a RESTful service with Node, Express and MongoDB              |  |  |
| <b>Experiment :- 6</b>  | <b>Date:- 10-03-2025</b>   | <b>Enrollment No:- 92200133003</b>   |  |

## GET

GET

http://localhost:3001/api/employees

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1 |

Status: 200 OK Size: 567 Bytes Time: 15 ms

Response Headers 6 Cookies Results Docs

```

1 [
2   {
3     "_id": "67f7faf02af39803c3fbf6f1",
4     "name": "Anjali Mehta",
5     "position": "HR Manager",
6     "salary": 58000,
7     "email": "anjali.mehta@example.com",
8     "__v": 0
9   },
10  {
11    "_id": "67f7faf12af39803c3fbf6f2",
12    "name": "Karan Patel",
13    "position": "UI/UX Designer",
14    "salary": 54000,
15    "email": "karan.patel@example.com",
16    "__v": 0
17  }

```

## POST

POST

http://localhost:3001/api/employees

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```

1 {
2   "name": "Rekha Joshi",
3   "position": "PHP Developer",
4   "salary": 62000,
5   "email": "rekha.joshi@example.com"
6 }

```

Status: 201 Created Size: 139 Bytes Time: 19 ms

Response Headers 6 Cookies Results Docs

```

1 {
2   "name": "Rekha Joshi",
3   "position": "PHP Developer",
4   "salary": 62000,
5   "email": "rekha.joshi@example.com",
6   "_id": "67f7fe452af39803c3fbf6fc",
7   "__v": 0
8 }

```

## PUT

PUT

http://localhost:3001/api/employees/67f7fe452af39803c3fbf6fc

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```

1 {
2   "name": "Rekha Joshi",
3   "position": "PHP Developer",
4   "salary": 88000,
5   "email": "rekha.joshi@example.com"
6 }

```

Status: 200 OK Size: 139 Bytes Time: 22 ms

Response Headers 6 Cookies Results Docs

```

1 {
2   "_id": "67f7fe452af39803c3fbf6fc",
3   "name": "Rekha Joshi",
4   "position": "PHP Developer",
5   "salary": 88000,
6   "email": "rekha.joshi@example.com",
7   "__v": 0
8 }

```

## DELETE

DELETE

http://localhost:3001/api/employees/67f7fe452af39803c3fbf6fc

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

Status: 200 OK Size: 30 Bytes Time: 15 ms

Response Headers 6 Cookies Results Docs

```

1 {
2   "message": "Employee deleted"
3 }

```