

PROJECT DOCUMENTATION

TOPIC: Employee Data Collection

Submitted By

ATHIRA P V

ADIT/TVM/19/007

NST(W)TRIVANDRUM

ABSTRACT

Employee Data Collection System project used to manage the entire data of the employees such as personal details, education details, work experience etc. ... Through this project company can manage all data online. The admin can manage and control the all activities in an easier and quicker way.

CONTENTS

ABSTRACT

1. INTRODUCTION

1.1. OBJECTIVE

1.2. PROJECT DESCRIPTION

1.3. SCOPE OF WORK

2. HARADWARE & SOFTWARE REQUIREMENTS

2.1. HARDWARE REQUIUREMENT

2.2. SOFTWARE REQUIREMENTS

3. SYSTEM DESIGN

3.1. ER DIAGRAM

3.2. CLASS DIAGRAM

3.3. FLOW CHART

4. APPENDICES

4.1. DATABASE TABLES

4.2. SOURCE CODE

4.3. SCREENSHOTS

5. CONCLUSION

6. REFERENCE

1. INTRODUCTION

1.1. OBJECTIVE

The objective of this project is to store employees all details and know about the currently working and dis continuing employees and their all information about the period of the working. The administrator has all rights to login and add, edit, remove and view of any employees details in their work place.

1.2. PROJECT DESCRIPTION

An employee data collection software allows you to store, manage, and track all employee data. It holds a variety of employee personnel fields such as name, age, job title, salary, length of service, etc for an administrator can to refer from. A good employee data collection software should be self-service based where the employees themselves can add and edit their information. This project is fully worked on the programming language html & java script, Mongo dB, Node JS.

2. HARADWARE & SOFTWARE REQUIREMENTS

2.1. HARADWARE REQUIREMENTS

- Personal Computer or Laptop
- Processor – Intel Core i3
- Hard Disk Capacity – 1 Tb

2.1. SOFTWARE REQUIREMENTS

- Operating System – Windows OS (10th Gen)
 - Text Editor – Notepad, Visual Studio Code
 - Browser – Microsoft edge, Google Chrome
 - Languages – Front End (Java Script) Back End (Mongo dB, Nodejs)
-
- **JavaScript**

JavaScript is a programming language commonly used in web development. JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functions can run after a webpage has loaded without communicating with the server. For example, a JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out. The JavaScript code can produce an error message before any information is actually transmitted to the server.

- **Mongo dB**

MongoDB is a cross-platform and open-source document-oriented database, a kind of NoSQL database. As a NoSQL database, MongoDB shuns the relational database's table-based structure to adapt JSON-like documents that have dynamic schemas which it calls JSON. This makes data integration for certain types of applications faster and easier. MongoDB is built for scalability, high availability and performance from a single server deployment to large and complex multi-site infrastructures.

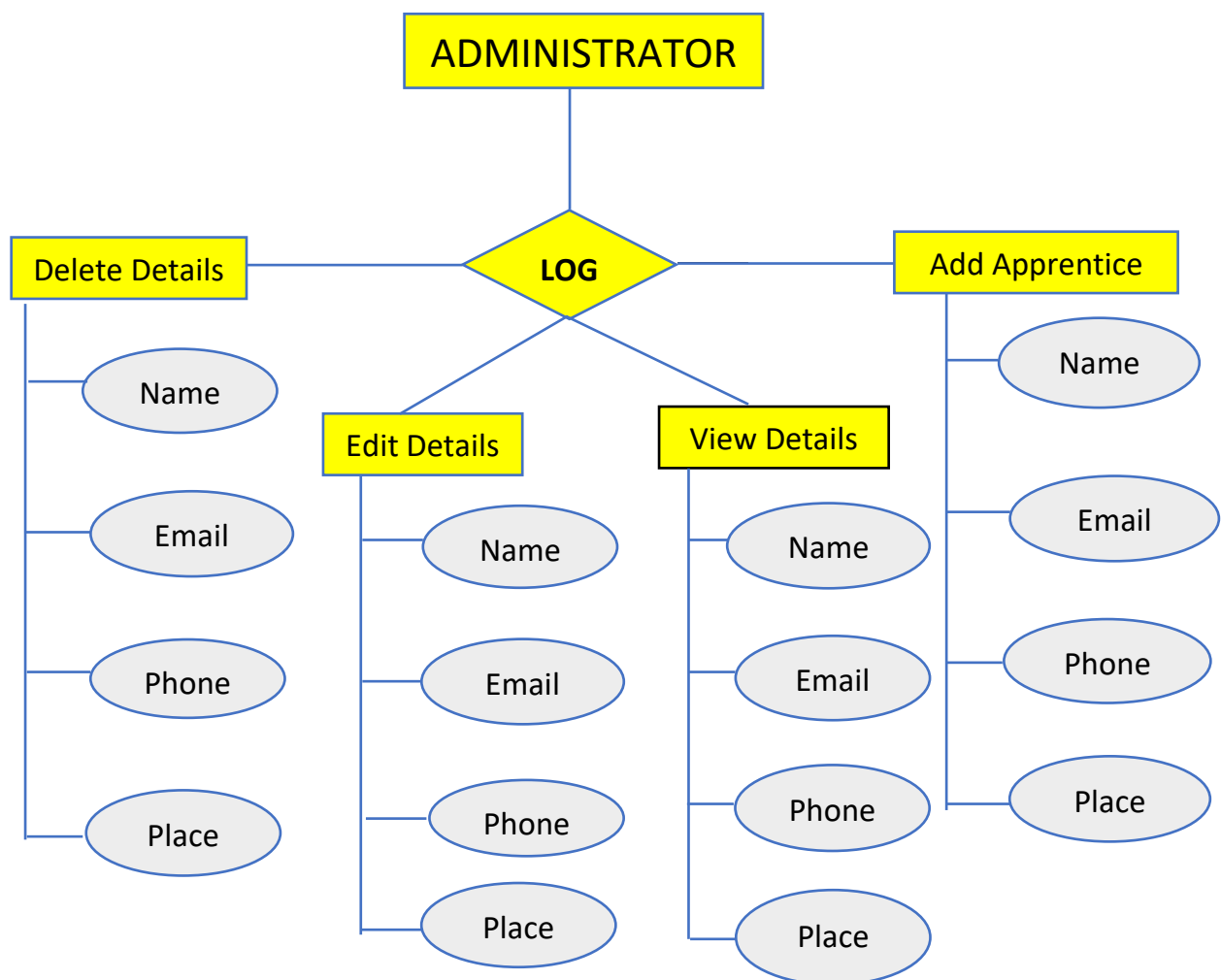
- **Nodejs**

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,^[6] unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

3. SYSTEM DESIGN

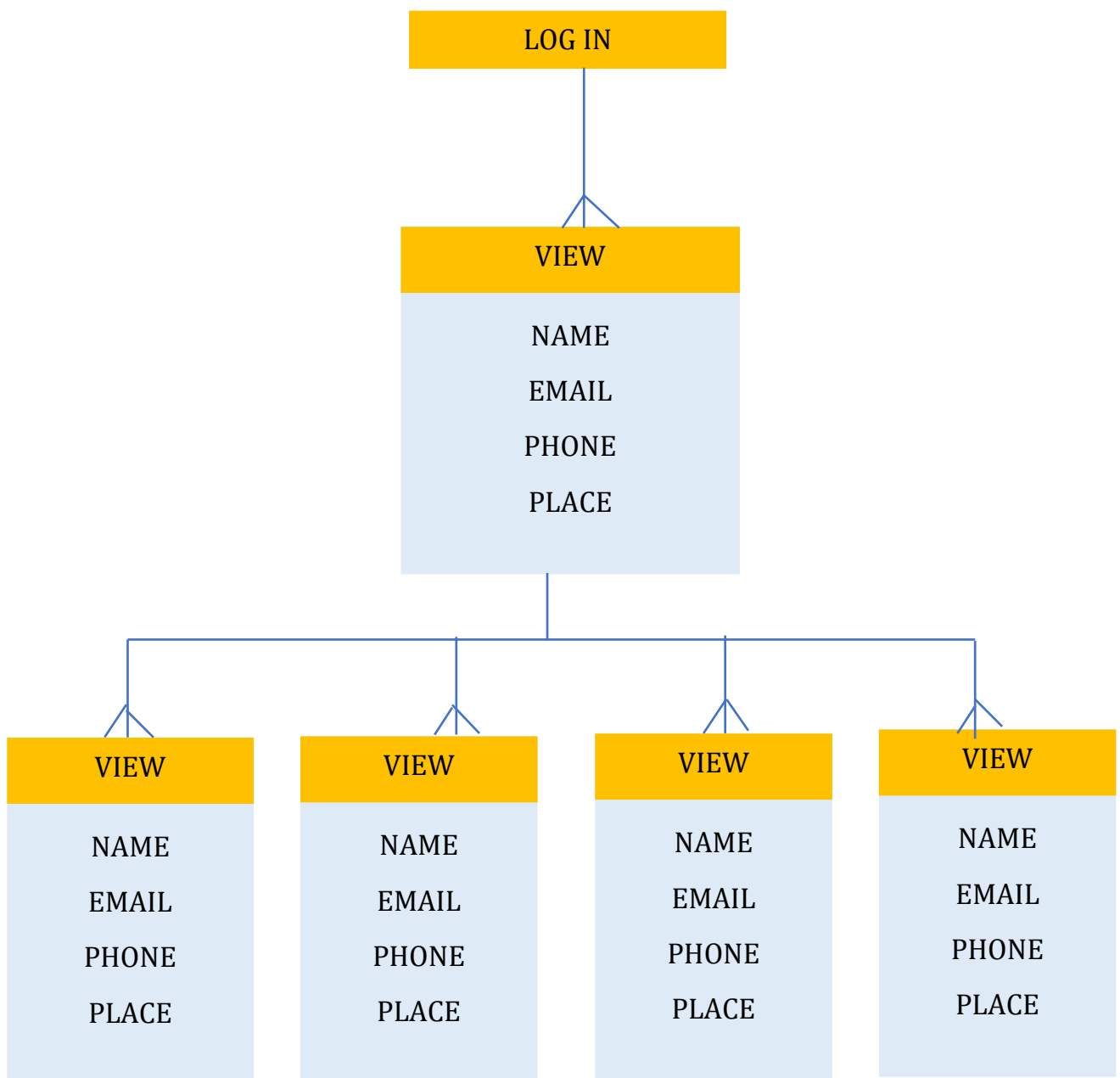
3.1. ER DIAGRAM

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database.



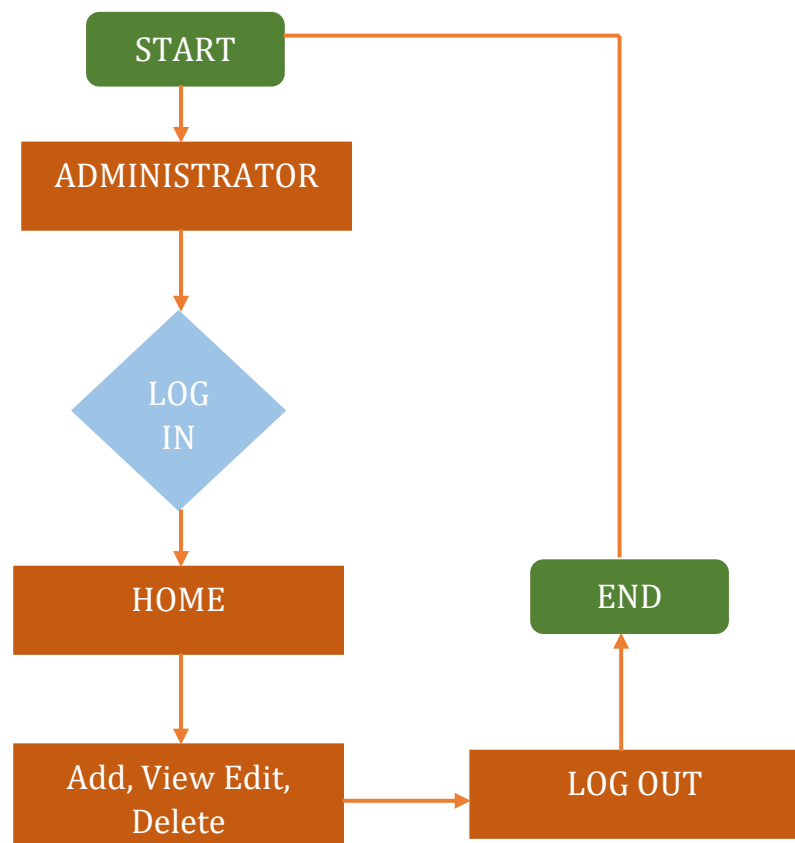
3.2. CLASS DIAGRAM

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



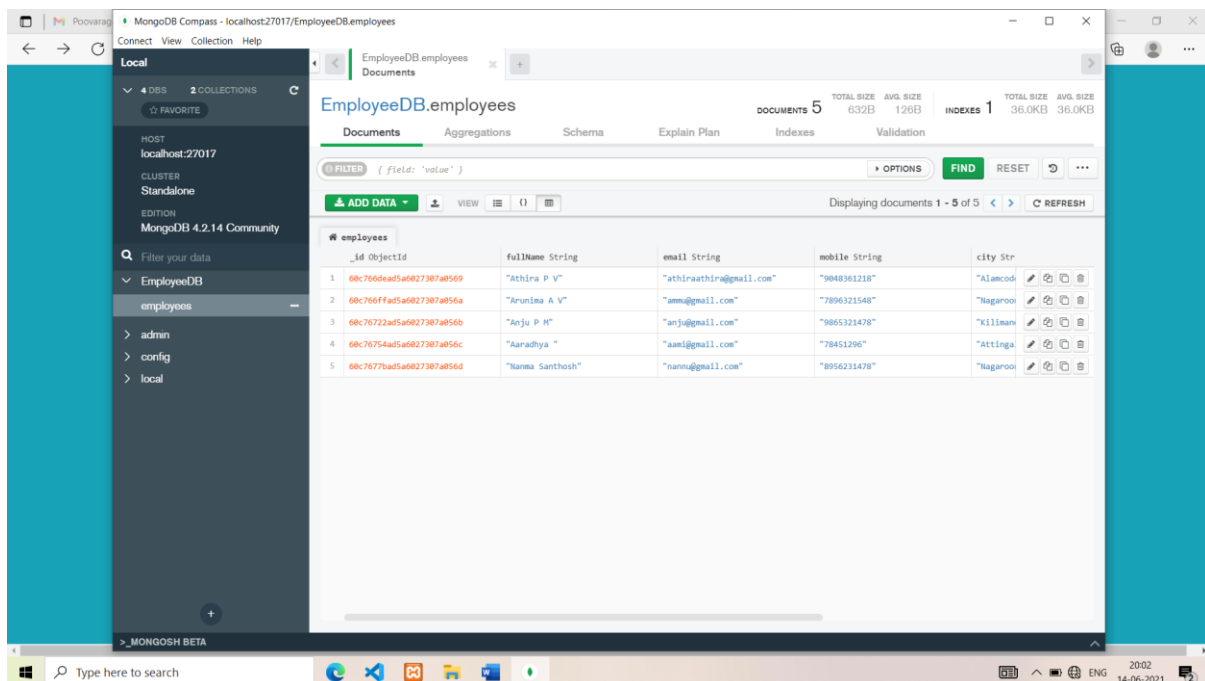
3.3. FLOW CHART

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields.



4. APPENDICES

4.1. DATABASE TABLE



The screenshot displays the MongoDB Compass interface for the 'EmployeeDB.employees' collection. The left sidebar shows the database structure, including the 'employees' collection. The main panel shows a table of 5 documents. The table has columns: _id (ObjectId), fullName (String), email (String), mobile (String), and city (String). The documents are listed with their _id, index, and corresponding values for each field. The interface also shows a filter bar, a 'FIND' button, and a 'REFRESH' button. The status bar at the bottom indicates the time is 20:02 on 14-06-2021.

#	employees	_id ObjectId	fullName String	email String	mobile String	city Str
1		68c766de5a6827387a9569	"Athira P V"	"athiraathira@gmail.com"	"9848361218"	"Alankodi"
2		68c766ffad5a6827387a956a	"Arunima A V"	"amm@gmail.com"	"7896321548"	"Nagarooi"
3		68c76722ad5a6827387a956b	"Anju P H"	"anju@gmail.com"	"9865321478"	"Kiliman"
4		68c76754ad5a6827387a956c	"Aaradhya "	"aam@gmail.com"	"78451296"	"Attinga"
5		68c7677bad5a6827387a956d	"Nanna Santhosh"	"nannu@gmail.com"	"8956231478"	"Nagarooi"

4.2. SOURCE CODE

- list.hbs

```
<h3><a class="btn btn-secondary" href="/employee"><i class="fa fa-plus"></i> Create New</a> Employee Data Collection</h3>
<table class="table table-striped">
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Phone</th>
      <th>Place</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    {{#each list}}
      <tr>
        <td>{{fullName}}</td>
        <td>{{this.email}}</td>
        <td>{{this.mobile}}</td>
        <td>{{this.city}}</td>
        <td>
          <a href="/employee/{{this._id}}"><i class="fa fa-pencil fa-lg" aria-hidden="true"></i></a>
          <a href="/employee/delete/{{this._id}}" onclick="return confirm('Are you sure to delete this record ?');"><i class="fa fa-trash fa-lg" aria-hidden="true"></i></a>
        </td>
      </tr>
    {{/each}}
  </tbody>
</table>
```

- **addOrEdit.hbs**

```
<h3>{{viewTitle}}</h3>

<form action="/employee" method="POST" autocomplete="off">
  <input type="hidden" name="_id" value="{{employee._id}}">
  <div class="form-group">
    <label>Name</label>
    <input type="text" class="form-
control" name="fullName" placeholder="Full Name" value="{{employee.fullName}}"
  >
    <div class="text-danger">
      {{employee.fullNameError}}</div>
  </div>
  <div class="form-group">
    <label>Email</label>
    <input type="text" class="form-
control" name="email" placeholder="Email" value="{{employee.email}}">
    <div class="text-danger">
      {{employee.emailError}}</div>
  </div>
  <div class="form-row">
    <div class="form-group col-md-6">
      <label>Phone</label>
      <input type="text" class="form-
control" name="mobile" placeholder="Mobile" value="{{employee.mobile}}">
    </div>
    <div class="form-group col-md-6">
      <label>Place</label>
      <input type="text" class="form-
control" name="city" placeholder="City" value="{{employee.city}}">
    </div>
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-info"><i class="fa fa-
database"></i> Submit</button>
    <a class="btn btn-secondary" href="/employee/list"><i class="fa fa-
list-alt"></i> View All</a>
  </div>
</form>
```

- **mainLayout.hbs**

```
<!DOCTYPE html>

<html>

<head>
  <title>Node.js express mongDB CRUD</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPM0"
    crossorigin="anonymous">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
</head>

<body class="bg-info">
  <div class="row">
    <div class="col-md-6 offset-md-3" style="background-color:rgb(236, 248, 123);margin-top: 25px;padding:20px;">
      {{{body}}}
    </div>
  </div>

</body>

</html>
```

- **employeeController.js**

```
const express = require('express');
var router = express.Router();
const mongoose = require('mongoose');
const Employee = mongoose.model('Employee');

router.get('/', (req, res) => {
  res.render("employee/addOrEdit", {
    viewTitle: "Insert employee Data"
  });
});

router.post('/', (req, res) => {
  if (req.body._id == '')
    insertRecord(req, res);
  else
    updateRecord(req, res);
});

function insertRecord(req, res) {
  var employee = new Employee();
  employee.fullName = req.body.fullName;
  employee.email = req.body.email;
  employee.mobile = req.body.mobile;
  employee.city = req.body.city;
  employee.save((err, doc) => {
    if (!err)
      res.redirect('employee/list');

    else {
      if (err.name == 'ValidationError') {
        handleValidationError(err, req.body);
        res.render("employee/addOrEdit", {
          viewTitle: "Insert employee Data",
          employee: req.body
        });
      }
      else
        console.log('Error during record insertion : ' + err);
    }
  });
}

function updateRecord(req, res) {
```

```

    Employee.findOneAndUpdate({ _id: req.body._id }, req.body, { new: true },
(err, doc) => {
    if (!err) { res.redirect('employee/list'); }
    else {
        if (err.name == 'ValidationError') {
            handleValidationError(err, req.body);
            res.render("employee/addOrEdit", {
                viewTitle: 'Update employee Data',
                employee: req.body
            });
        }
        else
            console.log('Error during record update : ' + err);
    }
});
}

```

```

router.get('/list', (req, res) => {
    Employee.find((err, docs) => {
        if (!err) {
            res.render("employee/list", {
                list: docs
            });
        }
        else {
            console.log('Error in retrieving employee list :' + err);
        }
    }).lean();
});

```

```

function handleValidationError(err, body) {
    for (field in err.errors) {
        switch (err.errors[field].path) {
            case 'fullName':
                body['fullNameError'] = err.errors[field].message;
                break;
            case 'email':
                body['emailError'] = err.errors[field].message;
                break;
            default:
                break;
        }
    }
}

```

```

router.get('/:id', (req, res) => {
    Employee.findById(req.params.id, (err, doc) => {
        if (!err) {

```

```

res.render("employee/addOrEdit", {
    viewTitle: "Update employee Data",
    employee: doc
});
}
}).lean();
});

router.get('/delete/:id', (req, res) => {
    Employee.findByIdAndRemove(req.params.id, (err, doc) => {
        if (!err) {
            res.redirect('/employee/list');
        }
        else { console.log('Error in employee delete :' + err); }
    });
});

module.exports = router;

```

- **db.js**

```

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/EmployeeDB', { useNewUrlParser: true }, (err) => {
    if (!err) { console.log('MongoDB Connection Succeeded.') }
    else { console.log('Error in DB connection : ' + err) }
});

require('./employee.model');

```


- **server.js**

```
require('./models/db');

const express = require('express');
const path = require('path');
const exphbs = require('express-handlebars');
const bodyParser = require('body-parser');

const employeeController = require('./controllers/employeeController');

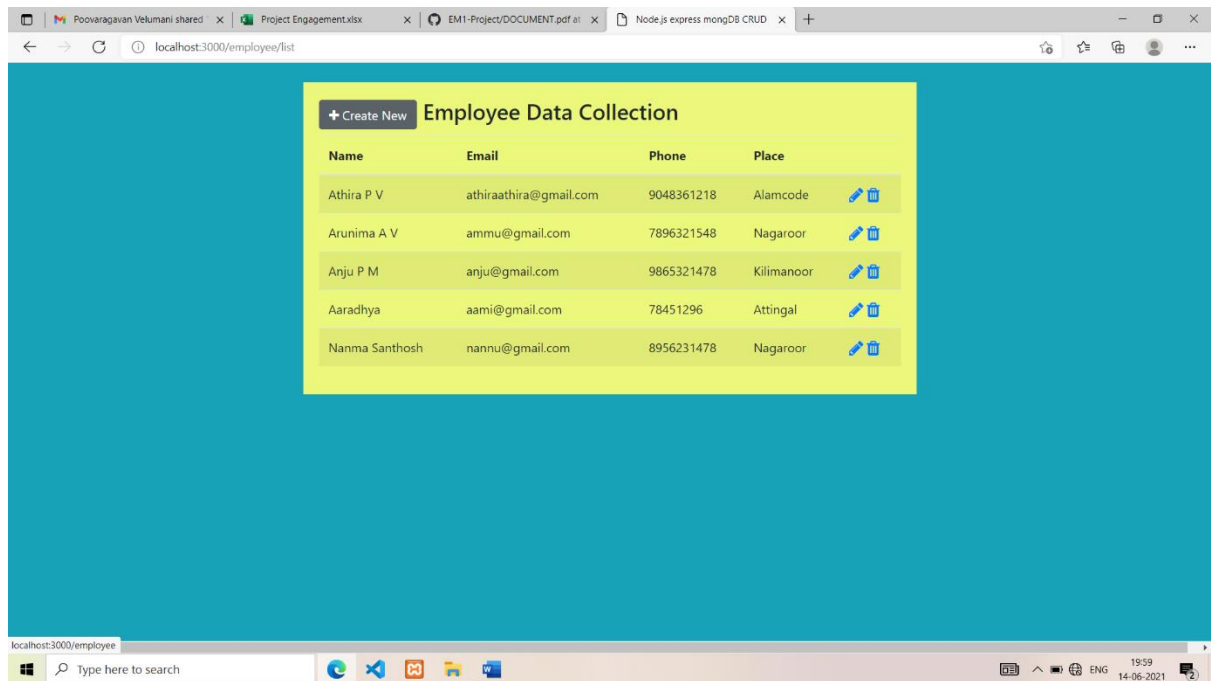
var app = express();
app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(bodyParser.json());
app.set('views', path.join(__dirname, '/views/'));
app.engine('hbs', exphbs({ extname: 'hbs', defaultLayout: 'mainLayout', layout
sDir: __dirname + '/views/layouts/' }));
app.set('view engine', 'hbs');

app.listen(3000, () => {
  console.log('Express server started at port : 3000');
});

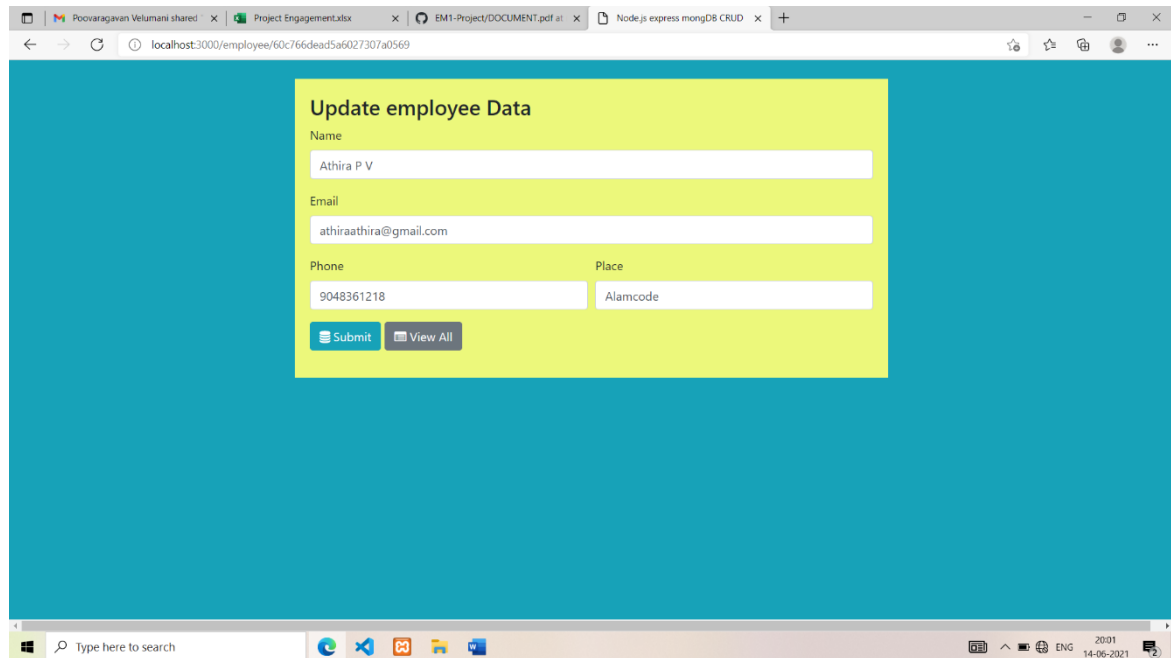
app.use('/employee', employeeController);
```

4.3. SCREENSHOTS

- **ADD/INSERT EMPLOYEE DATA**



- **EDIT EMPLOYEE DATA**

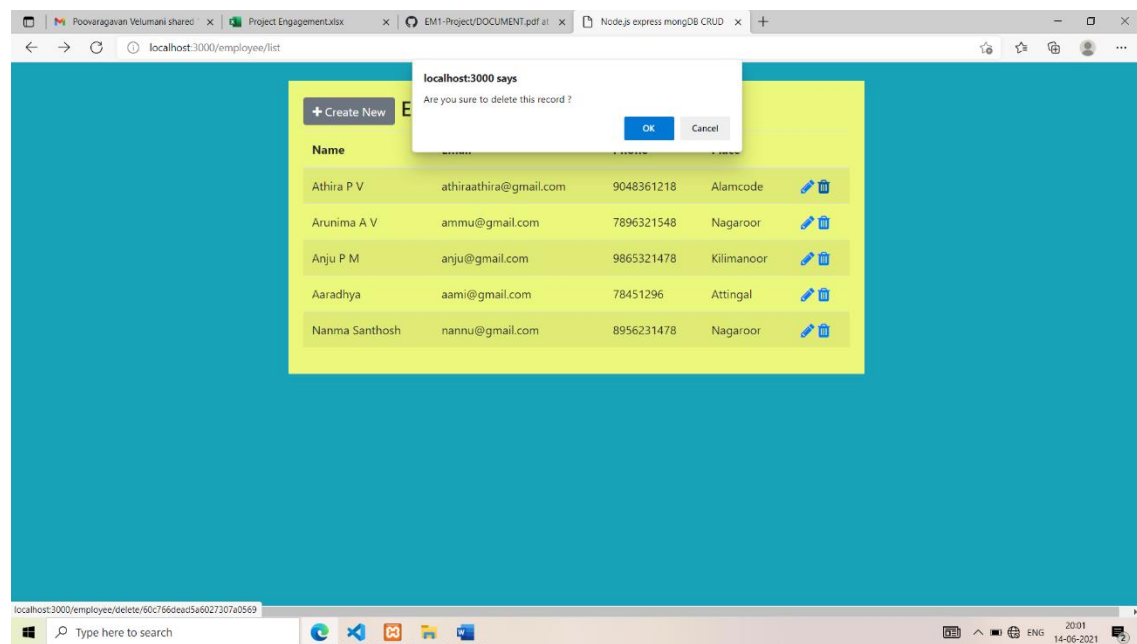


The screenshot shows a web browser window with the address bar displaying 'localhost:3000/employee/60c766dead5a6027307a0569'. The browser has several tabs open, including 'Poooragavan Velumani shared', 'Project Engagement.xlsx', 'EM1-Project/DOCUMENT.pdf a', and 'Node.js express mongDB CRUD'. The main content area features a yellow form titled 'Update employee Data'. The form contains the following fields and values:

Field	Value
Name	Athira P V
Email	athiraathira@gmail.com
Phone	9048361218
Place	Alamcode

At the bottom of the form, there are two buttons: 'Submit' and 'View All'.

- **DELETE EMPLOYEE DATA**



5. CONCLUSION

Employee Data Collection System project used to manage the entire data of the employees such as personal details, education details, work experience etc. ... Through this project company can manage all data online. The admin can manage and control the all activities in an easier and quicker way.

6. REFERENCE

- [Node.js MongoDB Get Started \(w3schools.com\)](https://www.w3schools.com/mongodb/mongodb_tutorial.asp)
- [Node.js MongoDB Tutorial with Examples \(guru99.com\)](https://guru99.com/mongodb-tutorial.html)