PROGRESSIVE PROJECT REPORT

EMPLOYEE DATA MANAGEMENT SYSTEM

Submitted by

SOORYA S DAS

ADIT/TVM/19/015

ADIT (2019-2021)

National Skill Training Institute for Women, Trivandrum

ABSTRACT

The project title is Employee Data Management System. It is a computer system that helps manage the information related to Employee who worked the shop earlier. Proper data collection and management are absolutely essential for ensuring that your company avoids data breach issues and the resulting loss of employee trust. Furthermore, effective employee data management is beneficial for your business period.

CONTENTS

6. CONCLUSION

7. REFERENCE

1. INTRODUCTION

1.1Objective/ Project Overview

The main objective of this project to give the information about Employee who worked the shop earlier. The Employee management have rights to add the Employee details and edit, update and delete the Employee details.

1.2 Project Description

This project is about developing the website for Employee Data Management System. Programming languages include JavaScript with Nodejs and MongodB are used for developing the website. The NodeJS is the The Company administrator can add, update, delete, view the Employee details.

1.3 SCOPE OF WORK

Data is essentially the plain facts and statistics collected during the operations of a business. They can be used to measure/record a wide range of business activities - both internal and external. While the data itself may not be very informative, it is the basis for all reporting and as such is crucial in business. Employee Data Management system helps to give a brief idea about the Employees details. It is helps to the Company Management easily identify their employee details. This website is to become a user-friendly and reliable for all users.

2. SOFTWARE DEVELOPMENT ENVIRONMENT

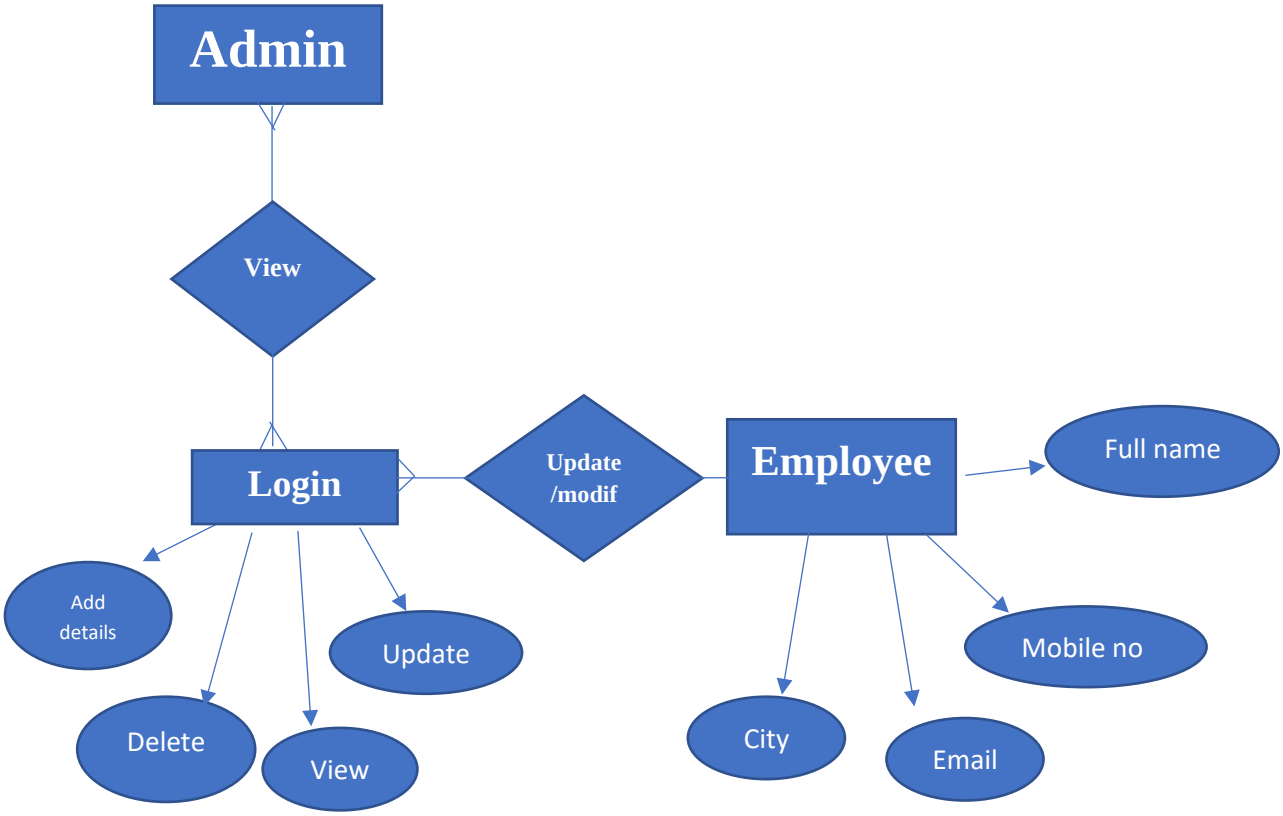
We using Nodejs with JavaScript and Mongo dB Database for developing us website. Node.js is an open-source server environment.Node.js allows you to run JavaScript on the server. Node.js is free. Node.js can generate dynamic page content. Node.js can create, open, read, write, delete, and close files on the server. Node.js can collect form data. Node.js can add, delete, modify data in your database. Node. js is a runtime environment that allows software developers to launch both the frontend and backend of web apps using JavaScript. Although JS underpins all the processes for app assembly, as a backend development environment, Node.

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables.

3. SYSTEM DESIGN

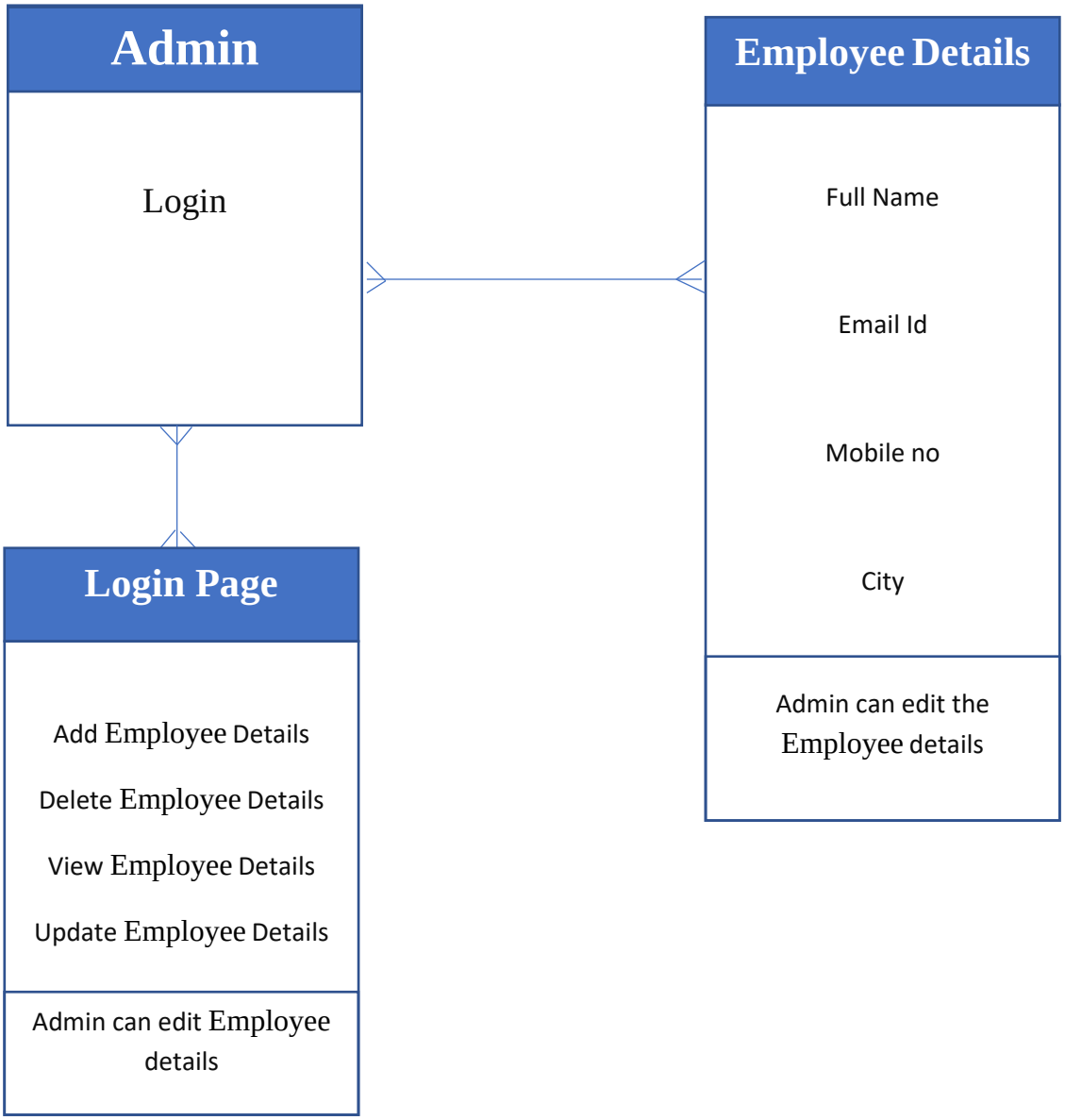
3.1 ER DIAGRAM

An Entity Relationship Diagram is a visual representation of different entities within a system and how they relate to each other



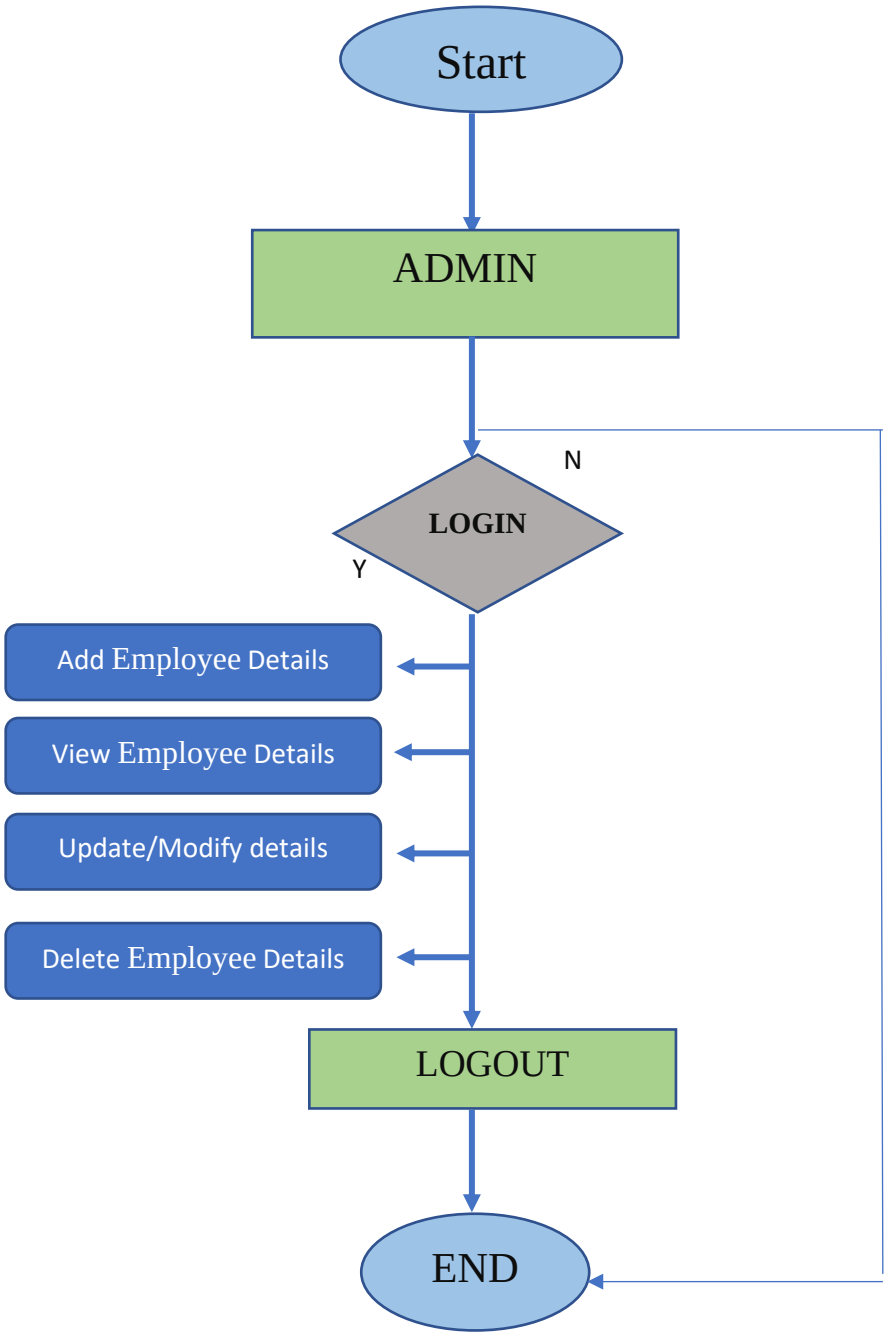
3.2 CLASS DIAGRAM

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.



3.3 FLOW CHART

A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.



4. SYSTEM REQUIREMENTS

4.1 SOFTWARE SPECIFICATION

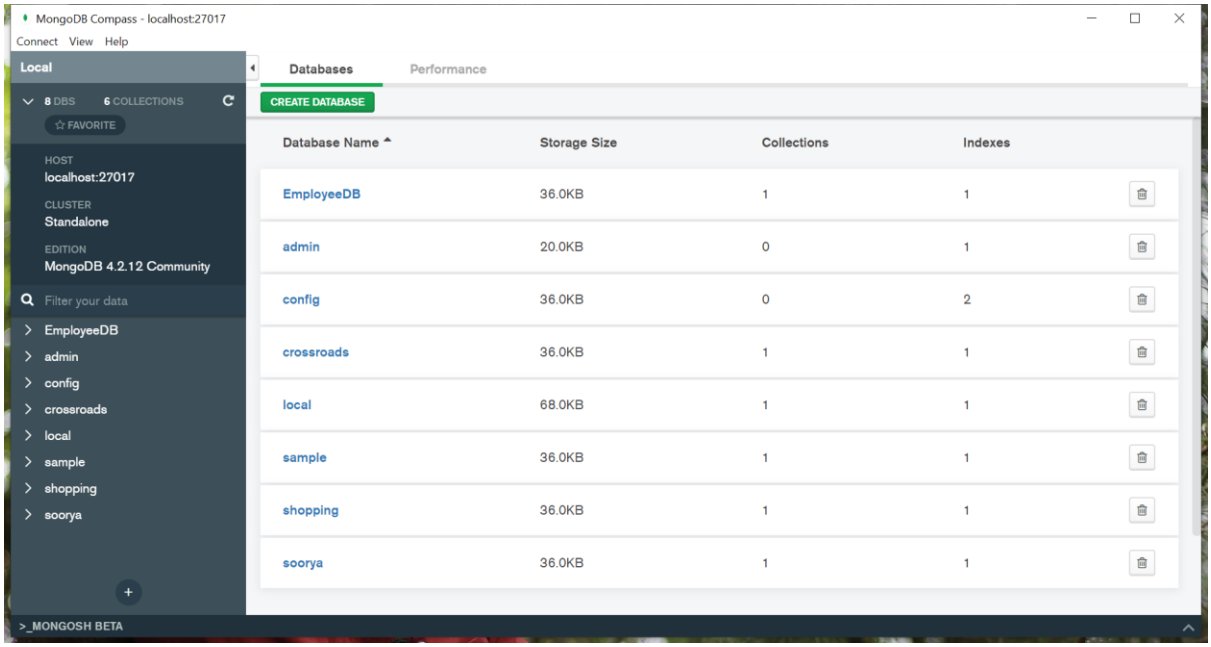
Operating System	:	Windows 7
Front End	:	JavaScript
Back End	:	Nodejs, MongoDB
Code Editor	:	Visual Code
Server	:	Web Browser: Google Chrome

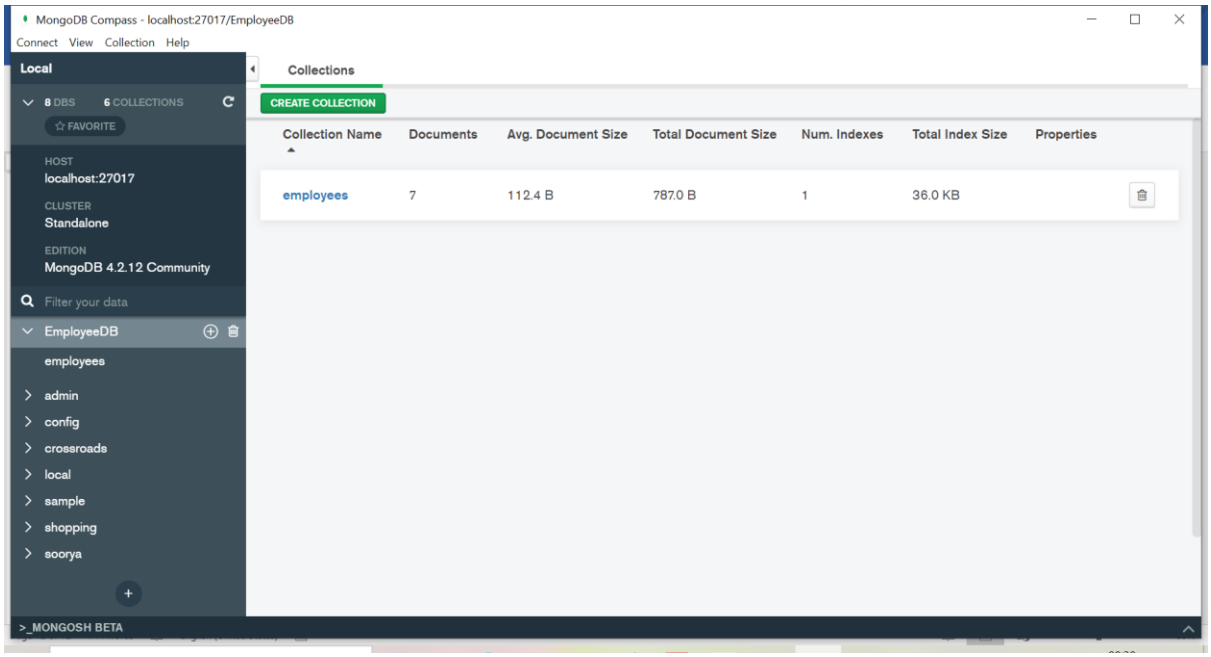
4.2 HARDWARE SPECIFICATION

RAM	:	1 GB or above
Processor	:	1 GHz or more
Hard Drive	:	32 GB or above
Network Connectivity	:	LAN or Wi-Fi

1. APPENDICES

5.1. DATABASE TABLES





5.2. SOURCE CODE

1. 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>Full 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>Mobile</label>
         <input type="text" class="form-control" name="mobile" placeholder="Mobile"
value="{{employee.mobile}}">
      </div>
      <div class="form-group col-md-6">
         <label>City</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>
```

2. list.hbs

```
<h3><a class="btn btn-secondary" href="/employee"><i class="fa fa-plus"></i> Create New</a>
Employee List</h3>
<table class="table table-striped">
  <thead>
    <tr>
      <th>Full Name</th>
      <th>Email</th>
      <th>Mobile</th>
      <th>City</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>
```

3. employee controller.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"
   });
});

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",
               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',
                    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",
                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;
```

4. 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');
```

5. main-layout.js

```
<!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/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
      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: #fff;margin-top:
25px;padding:20px;">
         {{{body}}}
      </div>
   </div>

</body>

</html>
```

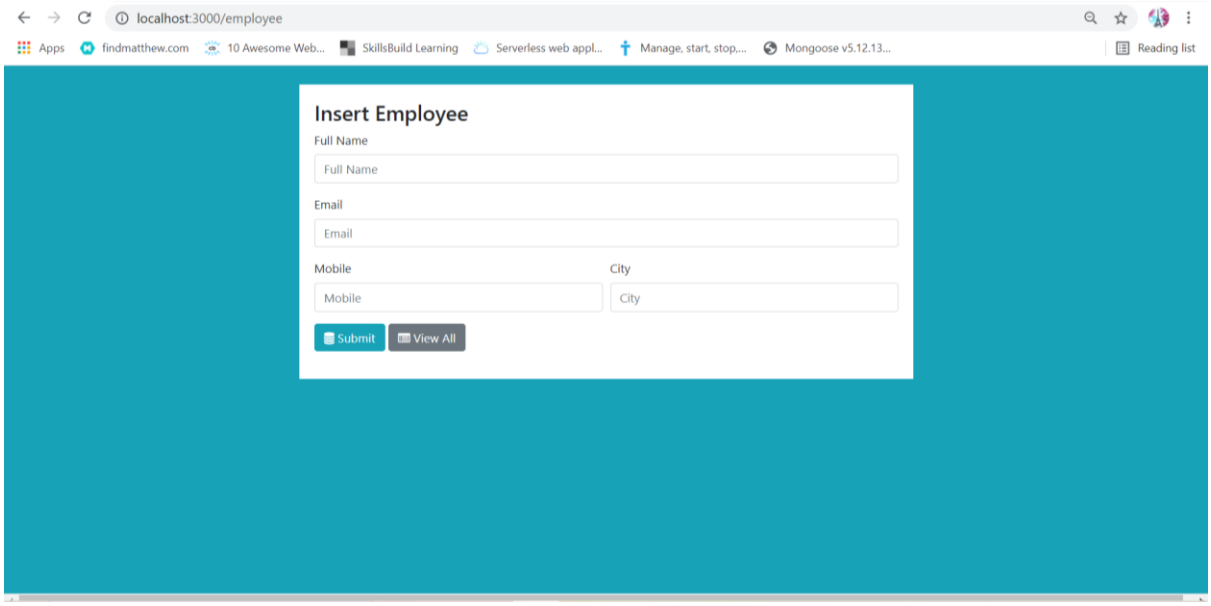
6. server.js

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



const express = require('express');

const path = require('path');

const exphbs = require('express-handlebars');
```

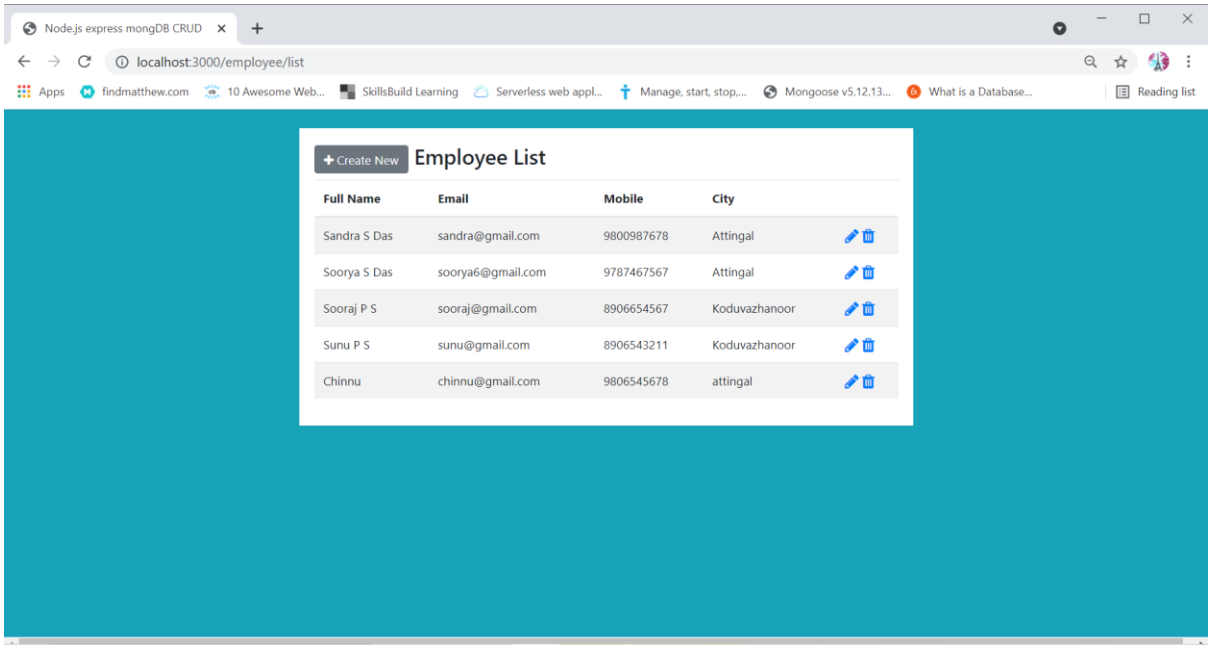
```
const bodyparser = require('body-parser');


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', layoutsDir:
__dirname + '/views/layouts/' }));

app.set('view engine', 'hbs');


app.listen(3000, () => {

    console.log('Express server started at port : 3000');

});


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

5.3. SCREENSHOTS

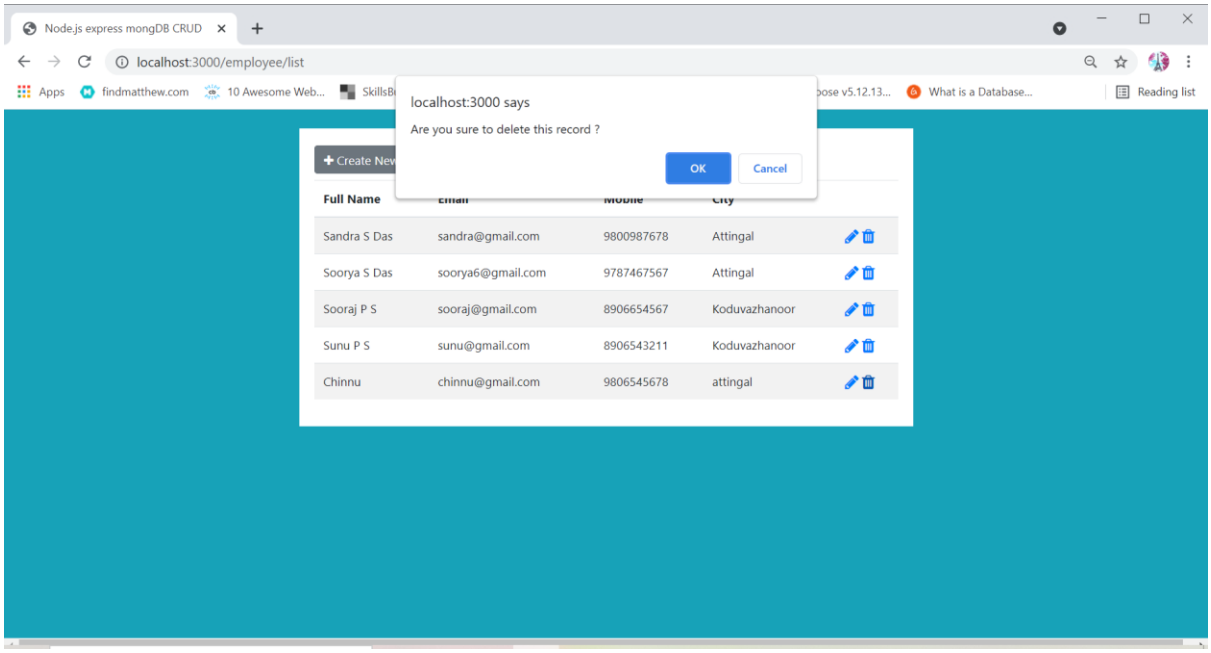
1. Insert page



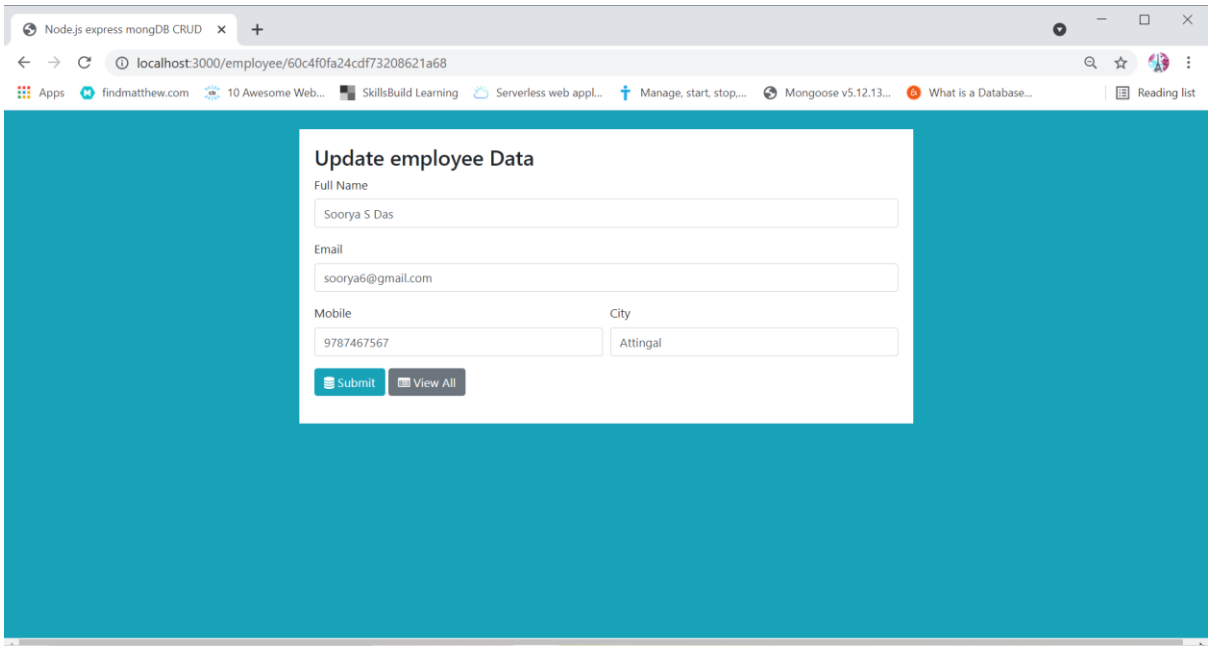
2. Detailed List



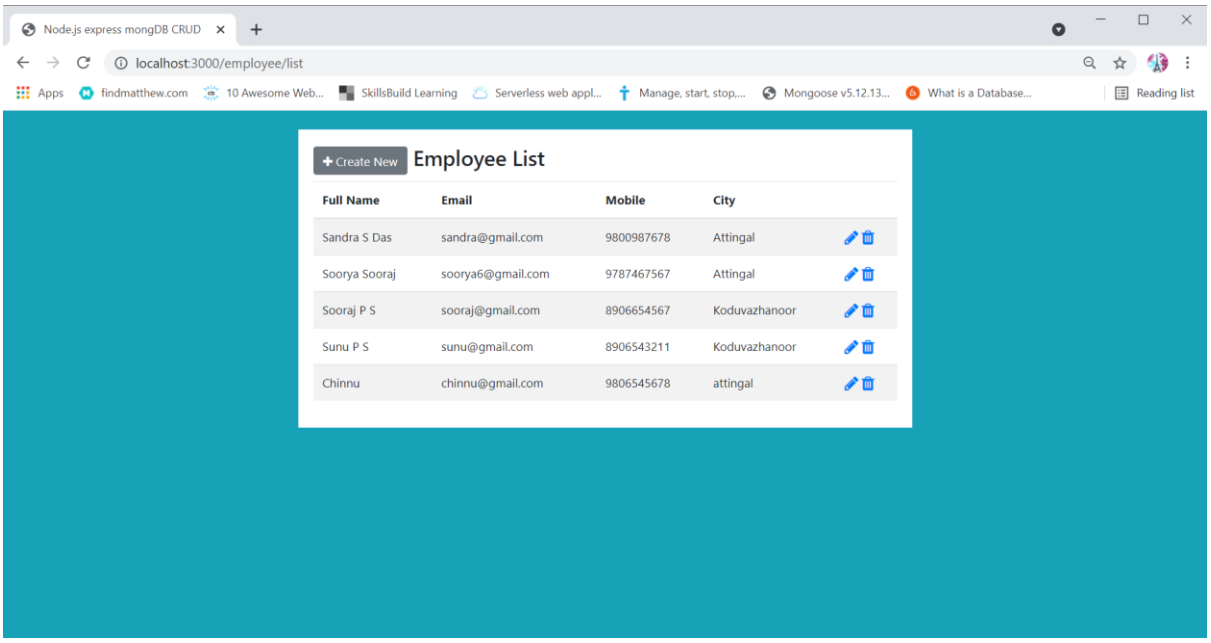
3. Delete data

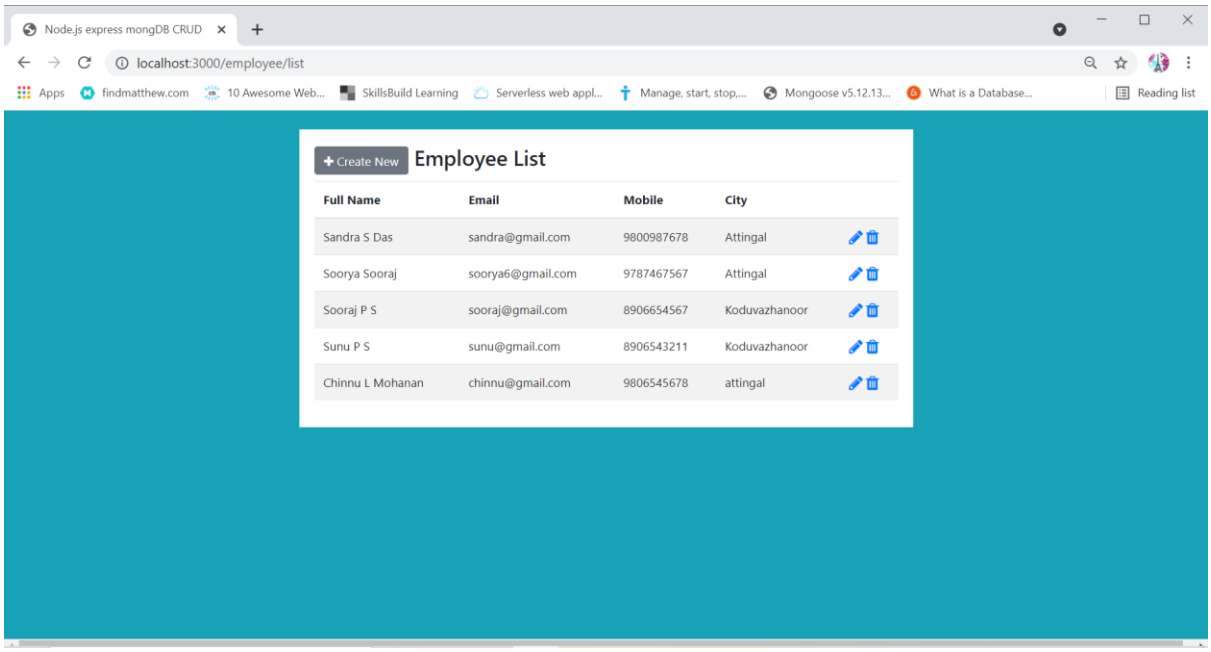


4. Update data

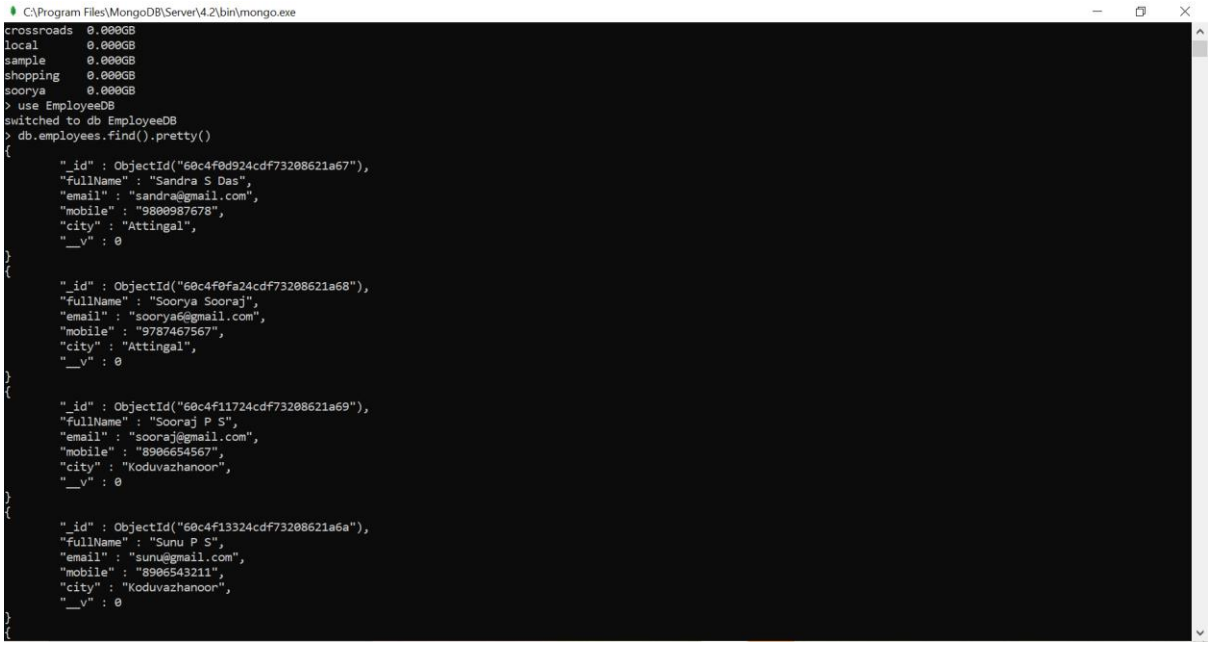


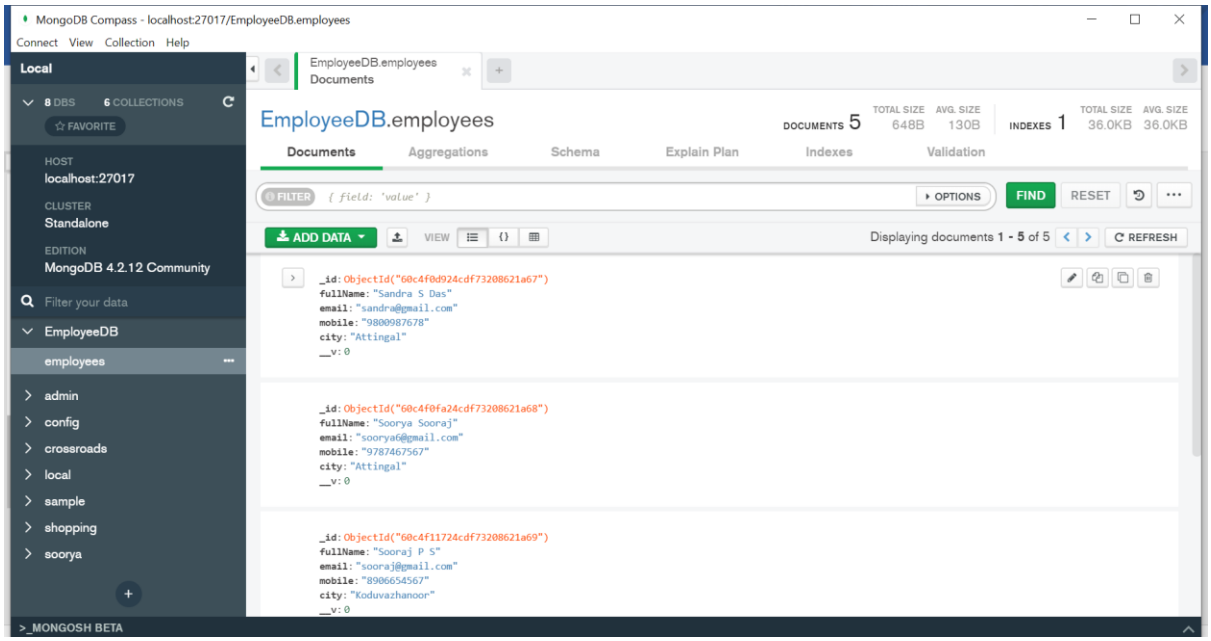
5. Updated data





5. Database


C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
crossroads 0.000GB
local 0.000GB
sample 0.000GB
shopping 0.000GB
soorya 0.000GB
> use EmployeeDB
switched to db EmployeeDB
> db.employees.find().pretty()
{
"_id" : ObjectId("60c4f0d924cdf73208621a67"),
"fullName" : "Sandra S Das",
"email" : "sandra@gmail.com",
"mobile" : "9800987678",
"city" : "Attingal",
"__v" : 0
}
{
"_id" : ObjectId("60c4f0fa24cdf73208621a68"),
"fullName" : "Soorya Sooraj",
"email" : "soorya6@gmail.com",
"mobile" : "9787467567",
"city" : "Attingal",
"__v" : 0
}
{
"_id" : ObjectId("60c4f11724cdf73208621a69"),
"fullName" : "Sooraj P S",
"email" : "sooraj@gmail.com",
"mobile" : "8906654567",
"city" : "Koduvazhanoor",
"__v" : 0
}
{
"_id" : ObjectId("60c4f13324cdf73208621a6a"),
"fullName" : "Sunu P S",
"email" : "sunu@gmail.com",
"mobile" : "8906543211",
"city" : "Koduvazhanoor",
"__v" : 0
}
{


MongoDB Compass - localhost:27017/EmployeeDB.employees
Connect View Collection Help
Local
8 DBS 6 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.2.12 Community
Filter your data
EmployeeDB
employees
admin
config
crossroads
local
sample
shopping
soorya
EmployeeDB.employees
Documents
EmployeeDB.employees
DOCUMENTS 5 TOTAL SIZE 648B AVG. SIZE 130B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 5 of 5
REFRESH
_id: ObjectId("60c4f0d924cdf73208621a67")
fullName: "Sandra S Das"
email: "sandra@gmail.com"
mobile: "9800987678"
city: "Attingal"
__v: 0
_id: ObjectId("60c4f0fa24cdf73208621a68")
fullName: "Soorya Sooraj"
email: "soorya6@gmail.com"
mobile: "9787467567"
city: "Attingal"
__v: 0
_id: ObjectId("60c4f11724cdf73208621a69")
fullName: "Sooraj P S"
email: "sooraj@gmail.com"
mobile: "8906654567"
city: "Koduvazhanoor"
__v: 0
>_MONGOSH BETA

6. CONCLUSION

The purpose of this project is to build a website for Employee Data Management System. It is a computer system that helps manage the information related to Employees who worked the shop earlier. Proper data collection and management are absolutely essential for ensuring that your company avoids data breach issues and the resulting loss of Employee trust. Furthermore, effective Employee data management is beneficial for your business period.

7. REFERENCE

1) https://www.w3schools.com

2) https://way2tutorial.com

3) https://www.tutorialrepublic.com