Subject: **20CS2036L – Web Technology Lab** **URK22AI1030**
Lab Exercise: **10. NodeJS Server-side Application with MongoDB Database (Duration: 2 hours)** **BHARATH KUMAR S**

Instructions: Odd no's (Q1), Even no's (Q2)
Note: Apply your creativity to design the templates

Aim:
To develop a NodeJS Server application with HTML forms and MongoDB database to perform CRUD operations

Q1:
Develop a NodeJS Server application to main Employee database with MongoDB.

- The application should have a welcome page with Navigation to Create, Read, Update, and Delete
- Schema includes name, empid, experience, designation, company, salary

Q2:
Develop a NodeJS Server application to main Student database with MongoDB.

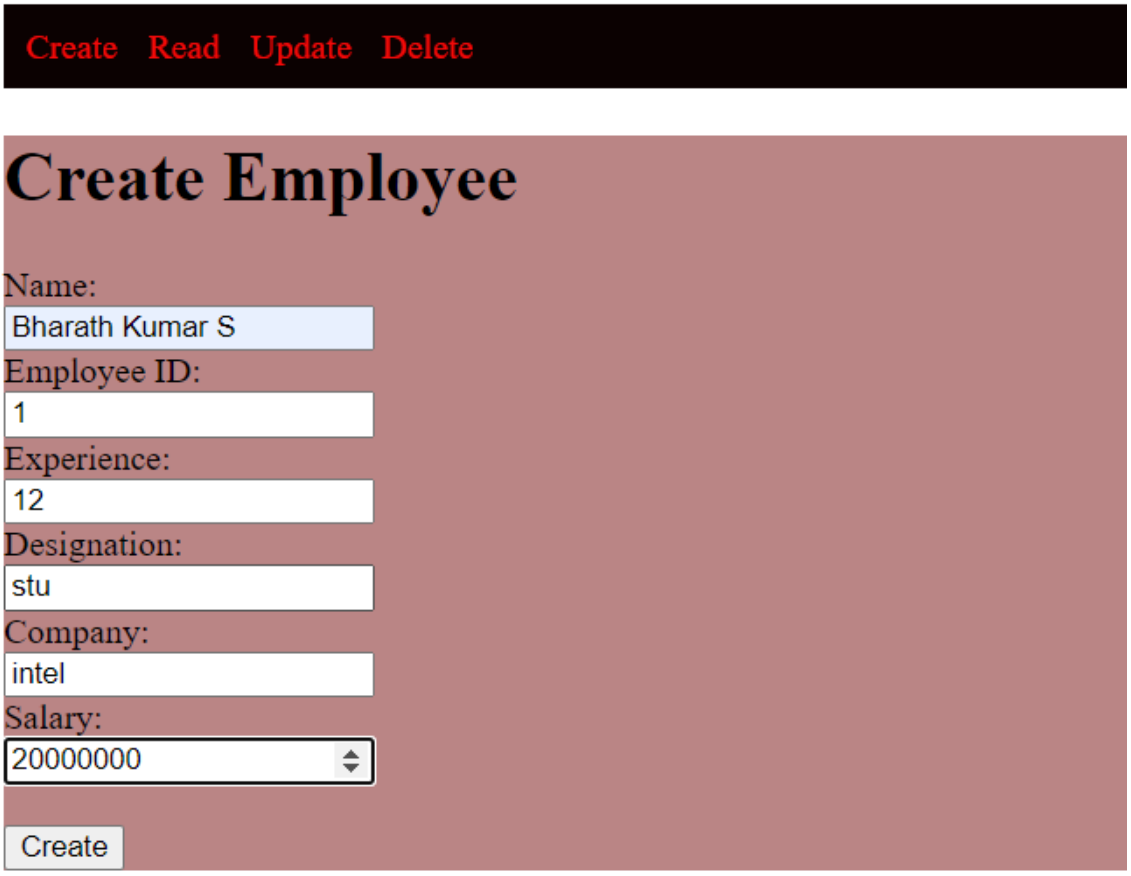
- The application should have a welcome page with Navigation to Create, Read , Update, and Delete
- Schema includes name, regno, age, year, mentor, cgpa

Source Code
Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Employee Database</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="navigation">
    <a href="#" onclick="loadPage('create')">Create</a>
    <a href="#" onclick="loadPage('read')">Read</a>
    <a href="#" onclick="loadPage('update')">Update</a>
    <a href="#" onclick="loadPage('delete')">Delete</a>
  </div>
  <div id="content">
    <!-- Content will be loaded here -->
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Create.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Create Employee</title>
</head>
<body >
   <div style="background-color: rgb(186, 133, 133);">
   <h1>Create Employee</h1>
   <form action="/create" method="POST">
      <label for="name">Name:</label><br>
      <input type="text" id="name" name="name" required><br>
      <label for="empid">Employee ID:</label><br>
      <input type="text" id="empid" name="empid" required><br>
      <label for="experience">Experience:</label><br>
      <input type="number" id="experience" name="experience" required><br>
      <label for="designation">Designation:</label><br>
      <input type="text" id="designation" name="designation" required><br>
      <label for="company">Company:</label><br>
      <input type="text" id="company" name="company" required><br>
      <label for="salary">Salary:</label><br>
      <input type="number" id="salary" name="salary" required><br><br>
      <button type="submit">Create</button>
   </form>
</div>
</body>
</html>
```



Update.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Update Employee</title>
</head>
<body>
   <h1>Update Employee</h1>
   <form action="/update" method="POST">
      <label for="empid">Employee ID:</label><br>
      <input type="text" id="empid" name="empid" required><br>
      <label for="salary">New Salary:</label><br>
      <input type="number" id="salary" name="salary" required><br><br>
      <button type="submit">Update</button>
   </form>
</body>
</html>
```

Delete.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Delete Employee</title>
</head>
<body >
   <div style="background-color: aquamarine;">
   <h1>Delete Employee</h1>
   <form action="/delete" method="POST">
      <label for="empid">Employee ID:</label><br>
      <input type="text" id="empid" name="empid" required><br><br>
      <button type="submit">Delete</button>
   </form>
</div>
</body>
</html>
```

Read.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Read Employee</title>
</head>
<body>
   <h1>Read Employee</h1>
   <form action="/read" method="GET">
      <button type="submit">Read</button>
   </form>

   <div id="content"></div>
</body>
</html>
```

Server.js:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const app = express();

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/employeeDB', { useNewUrlParser: true,
useUnifiedTopology: true });
const db = mongoose.connection;
db.once('open', () => {
  console.log('Connected to MongoDB');
});

// Employee Schema
const employeeSchema = new mongoose.Schema({
  name: String,
  empid: String,
  experience: Number,
  designation: String,
  company: String,
  salary: Number
});

const Employee = mongoose.model('Employee', employeeSchema);

app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('public'));

// Routes
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/public/index.html');
});

app.get('/create', (req, res) => {
  res.sendFile(__dirname + '/public/create.html');
});

app.post('/create', (req, res) => {
  const { name, empid, experience, designation, company, salary } = req.body;
  const newEmployee = new Employee({
    name,
    empid,
    experience,
    designation,
    company,
    salary
  });
  newEmployee.save()
    .then(() => {
      res.send('Employee saved successfully');
    })
    .catch(err => {
```

```
            console.error(err);
            res.send('Error saving employee');
        });
});

// Read route
app.get('/read', (req, res) => {
    Employee.find({})
        .then(employees => {
            res.json(employees);
        })
        .catch(err => {
            console.error(err);
            res.status(500).send('Error reading employees');
        });
});


app.get('/update', (req, res) => {
    // You can send an HTML file for update form here
    res.sendFile(__dirname + '/public/update.html');
});

// Update route
app.post('/update', (req, res) => {
    const { empid, salary } = req.body;
    Employee.findOneAndUpdate({ empid: empid }, { $set: { salary: salary } }, { new: true })
        .then(updatedEmployee => {
            if (!updatedEmployee) {
                res.status(404).send('Employee not found');
            } else {
                res.send('Employee updated successfully');
            }
        })
        .catch(err => {
            console.error(err);
            res.status(500).send('Error updating employee');
        });
});


// Update route



// Delete route
app.get('/delete', (req, res) => {
    // You can send an HTML file for delete form here
    res.sendFile(__dirname + '/public/delete.html');
});

// Delete route
app.post('/delete', (req, res) => {
    const { empid } = req.body;
```

```
    Employee.findOneAndDelete({ empid: empid })
      .then(deletedEmployee => {
        if (!deletedEmployee) {
          res.status(404).send('Employee not found');
        } else {
          res.send('Employee deleted successfully');
        }
      })
      .catch(err => {
        console.error(err);
        res.status(500).send('Error deleting employee');
      });
});
```

```
const PORT = process.env.PORT || 4020;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

After Create :

Create Read Update Delete

Name	Employee ID	Experience	Designation	Company	Salary
Bharath Kumar S	1	12	stu	intel	20000000

After Update::

Create Read Update Delete

Name	Employee ID	Experience	Designation	Company	Salary
Bharath Kumar S	1	12	stu	intel	50000000

After Delete and Before Create any Data:

Create Read Update Delete

Name	Employee ID	Experience	Designation	Company	Salary

Result:

Successfully created a a NodeJS Server application to main Employee database with MongoDB.