# Subject: 20CS2036L – Web Technology Lab URK22AI1048

**Lab Exercise: 10. NodeJS Server-side Application with HARIHARAN K MongoDB Database (Duration: 2 hours)**

**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>Student Management</title>

<script src="text.js"></script>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<div class="navbar">

<a href="/">Home</a> |

<a href="/create">Create</a> |

<a href="/read">Read</a> |

<a href="/update">Update</a> |

<a href="/delete">Delete</a>

</div>

<p>Welcome to the Home Page</p>

</body>

</html>

## Create.html

<!-- create.html -->

<html>

<head>

<title>Create Student</title>

</head>

<body>

<!-- Common navigation HTML (optional here since it's served by the server) -->

<h2>Create Student</h2>

<form action="/create" method="post">

Name: <input type="text" name="name" required /><br /><br />

Registration Number: <input type="number" name="regno" required /><br /><br />

Age: <input type="number" name="age" required /><br /><br />

Year: <input type="number" name="year" required /><br /><br />

Mentor: <input type="text" name="mentor" required /><br /><br />

CGPA: <input type="number" name="cgpa" required /><br /><br />

<input type="submit" value="Submit" />

</form>

</body>

</html>



## Update.html:

<!-- update.html -->

<html>

<head>

<title>Update Student</title>

</head>

<body>

<h2>Update Student</h2>

<form action="/update" method="post">

Registration Number (to identify student):

<input type="number" name="regno" required /><br /><br />

New Name: <input type="text" name="name" /><br /><br />

New Age: <input type="number" name="age" /><br /><br />

New Year: <input type="number" name="year" /><br /><br />

New Mentor: <input type="text" name="mentor" /><br /><br />

New CGPA: <input type="number" name="cgpa" /><br /><br />

<input type="submit" value="Update" />

</form>

<script>

// This script will be used later for alerting if no student is found

</script>

</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 Student</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<div class="container">

<h2>Delete Student</h2>

<form action="/delete" method="post">

<label for="regno">Registration Number (to identify student):</label>

<input type="number" name="regno" required><br><br>

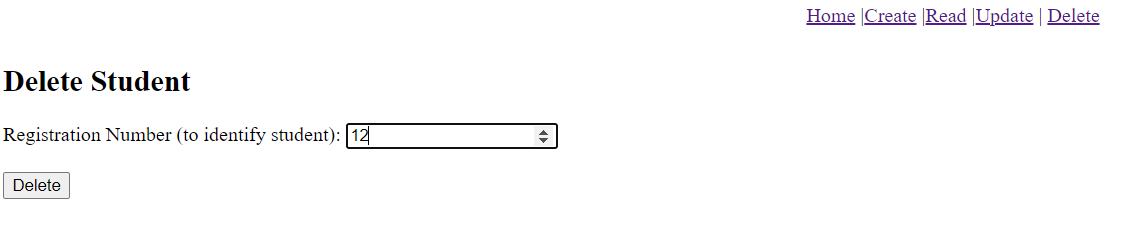
<input type="submit" value="Delete">

</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</title>

</head>

<body>

<h1>Read</h1>

<form action="/read" method="GET">

<button type="submit">Read</button>

</form>

<div id="content"></div>

</body>

</html>

## test.js:

const http = require("http");

const fs = require("fs");

const url = require("url");

const path = require("path");

const mongoose = require("mongoose");

const queryString = require("querystring");

mongoose

.connect("mongodb://localhost:27017/ex10", { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log("MongoDB Connected"))

.catch((err) => console.error("MongoDB Connection Error:", err));

const studentSchema = new mongoose.Schema({

name: String,

regno: Number,

age: Number,

year: Number,

mentor: String,

cgpa: Number

}, { collection: 'students' });

const Student = mongoose.model("students", studentSchema);

const navbar = () =>

"<div style='background-color:black; padding: 10px 0; text-align: center;'><a href='/'>Home</a> |<a href='/create'>Create</a> |<a href='/read'>Read</a> |<a href='/update'>Update</a> | <a href='/delete'>Delete</a></div>";

const server = http.createServer((req, res) => {

const { pathname } = url.parse(req.url, true);

switch (pathname) {

case "/":

res.writeHead(200, { "Content-Type": "text/html" });

res.end(

<html><body>${navbar()}<p>Welcome to the Home Page</p></body></html>

);

break;

case "/create":

if (req.method === "GET") {

serveFormPage(res, "create.html");

} else if (req.method === "POST") {

collectRequestData(req, (data) => {

Student.create(data)

.then(() => {

res.writeHead(302, { Location: "/read" });

res.end();

})

.catch((err) => {

console.error("Error creating student:", err);

res.writeHead(500);

res.end("Error creating student");

});

});

}

break;

case "/read":

if (req.method === "GET") {

Student.find()

.then(function (students) {

res.writeHead(200, { "Content-Type": "text/html" });

let content = ${navbar()}<div style='text-align: center;'>;

content += "<h2>Student Records</h2>";

content +=

"<table style='border: 1px solid blue; cellspacing: 1px; width: 80%;'>";

content +=

"<tr><th>Name</th><th>Registration Number</th><th>Age</th><th>Year</th><th>Mentor</th><th>CGPA</th></tr>";

students.forEach((student) => {

content += "<tr>";

content += <td>${student.name}</td>;

content += <td>${student.regno}</td>;

content += <td>${student.age}</td>;

content += <td>${student.year}</td>;

content += <td>${student.mentor}</td>;

content += <td>${student.cgpa}</td>;

content += "</tr>";

});

content += "</table></div>";

res.end(content);

})

.catch((err) => {

console.error("Error fetching students:", err);

res.writeHead(500);

res.end("Error fetching students");

});

}

break;

case "/update":

if (req.method === "GET") {

serveFormPage(res, "update.html");

} else if (req.method === "POST") {

collectRequestData(req, (data) => {

const { regno, ...updateData } = data;

Student.findOneAndUpdate({ regno: regno }, updateData)

.then(() => {

res.writeHead(302, { Location: "/read" });

res.end();

})

.catch((err) => {

console.error("Error updating student:", err);

res.writeHead(500);

res.end("Error updating student");

});

});

}

break;

case "/delete":

if (req.method === "GET") {

serveFormPage(res, "delete.html");

} else if (req.method === "POST") {

collectRequestData(req, (data) => {

Student.findOneAndDelete({ regno: data.regno })

.then(() => {

res.writeHead(302, { Location: "/read" });

res.end();

})

.catch((err) => {

console.error("Error deleting student:", err);

res.writeHead(500);

res.end("Error deleting student");

});

});

}

break;

default:

res.writeHead(404);

res.end("<html><body><p>Not Found</p></body></html>");

}

});

function serveFormPage(res, pageName) {

const filePath = path.join(\_\_dirname, pageName);

fs.readFile(filePath, (err, data) => {

if (err) {

console.error(Error reading ${filePath}:, err);

res.writeHead(500);

res.end("Server Error: Unable to read form page.");

return;

}

res.writeHead(200, { "Content-Type": "text/html" });

res.end(<html><body>${navbar()}${data}</body></html>);

});

}

function collectRequestData(request, callback) {

let data = "";

request.on("data", (chunk) => (data += chunk));

request.on("end", () => {

callback(queryString.parse(data));

});

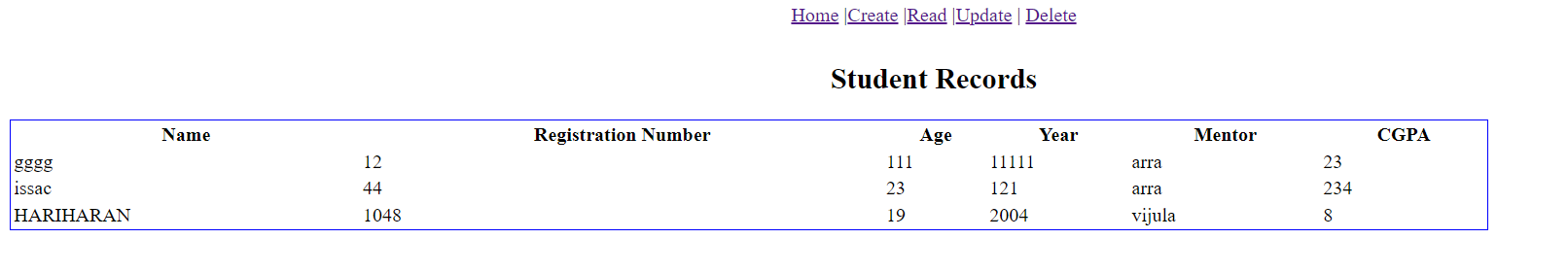
}

server.listen(9200, () => {

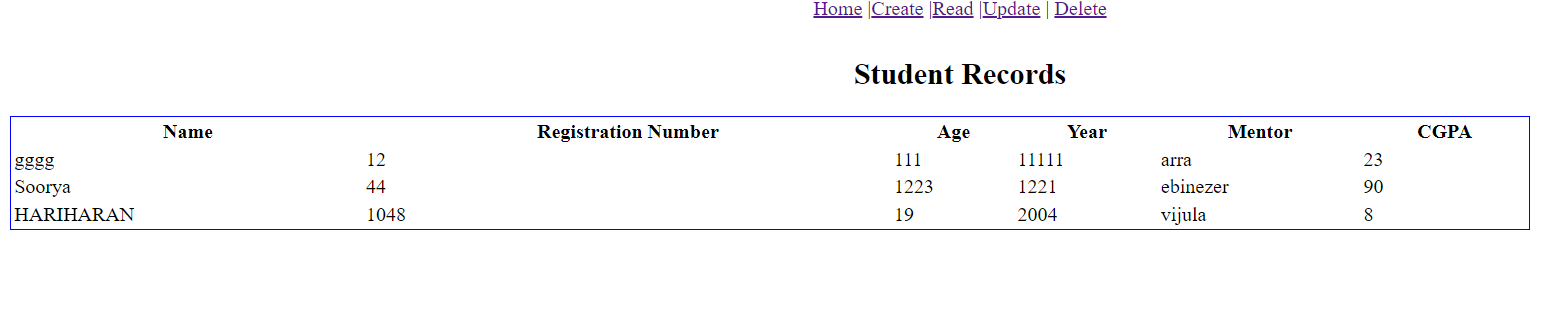
console.log("Server running on http://localhost:9200");

});

After Create :



After Update:



After Delete:

A close-up of a student registration

Description automatically generated

# Result:

Successfully created a NodeJS Server application to main Student

database with MongoDB.