**Batch: D - 1 Roll No.: 16010122096**

**Experiment / assignment / tutorial No. 08**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Implementation of MongoDB Concept |

**AIM:** To Implement the Concept of MongoDB

**Problem Definition:**

1. Implement CRUD (Create, Read, Update, Delete) operations in an NodeJS/Express.js application using Mongoose.

2. Implement Student Application form and store the data in MongoDB database.

**Resources used:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

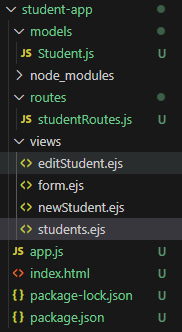
**CO 1:**.Build full stack applications in MongoDB using the MERN technologies.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O’ Reilly 2 nd Edition, 2016.

**Pre Lab/ Prior Concepts:**

**Implementation Details:   
**

Student.js

const mongoose = require('mongoose');

// Define Student schema

const studentSchema = new mongoose.Schema({

name: {

type: String,

required: true

},

email: {

type: String,

required: true

},

age: {

type: Number,

required: true

},

course: {

type: String,

required: true

}

});

module.exports = mongoose.model('Student', studentSchema);

studentRoutes.js

const express = require('express');

const router = express.Router();

const Student = require('../models/Student'); // Import the Student model

// Get all students (Read)

router.get('/students', async (req, res) => {

try {

const students = await Student.find();

res.render('students', { students });

} catch (err) {

res.status(500).send(err.message);

}

});

// Show form to create a new student

router.get('/students/new', (req, res) => {

res.render('newStudent');

});

// Create a new student (Create)

router.post('/students', async (req, res) => {

const student = new Student(req.body);

try {

await student.save();

res.redirect('/students');

} catch (err) {

res.status(500).send(err.message);

}

});

// Show form to edit a student (Update form)

router.get('/students/:id/edit', async (req, res) => {

try {

const student = await Student.findById(req.params.id);

res.render('editStudent', { student });

} catch (err) {

res.status(500).send(err.message);

}

});

// Update a student (Update)

router.put('/students/:id', async (req, res) => {

try {

await Student.findByIdAndUpdate(req.params.id, req.body);

res.redirect('/students');

} catch (err) {

res.status(500).send(err.message);

}

});

// Delete a student (Delete)

router.delete('/students/:id', async (req, res) => {

try {

await Student.findByIdAndDelete(req.params.id);

res.redirect('/students');

} catch (err) {

res.status(500).send(err.message);

}

});

module.exports = router;

app.js

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const methodOverride = require('method-override');

const studentRoutes = require('./routes/studentRoutes'); // Assuming you have this route set up

const app = express();

// Connect to MongoDB (Updated to remove deprecated options)

mongoose.connect('mongodb://localhost:27017/studentdb', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Middleware

app.use(bodyParser.urlencoded({ extended: true }));

app.use(methodOverride('\_method')); // For PUT and DELETE methods

app.set('view engine', 'ejs'); // Assuming you are using EJS as the template engine

// Routes

app.use(studentRoutes);

// Start the server

app.listen(3000, () => {

console.log('Server running on port 3000');

});

form.ejs

<!DOCTYPE html>

<html>

<head>

<title>Student Application Form</title>

</head>

<body>

<h1>Student Application Form</h1>

<form action="/students" method="POST">

<input type="text" name="name" placeholder="Name" required>

<input type="email" name="email" placeholder="Email" required>

<input type="number" name="age" placeholder="Age" required>

<input type="text" name="course" placeholder="Course" required>

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

</form>

<h2>Students List</h2>

<ul>

<% students.forEach(student => { %>

<li><%= student.name %> - <%= student.email %> - <%= student.age %> - <%= student.course %></li>

<% }); %>

</ul>

</body>

</html>

newStudent.ejs

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Add New Student</title>

</head>

<body>

<h1>Add New Student</h1>

<form action="/students" method="POST">

<input type="text" name="name" placeholder="Name" required>

<input type="email" name="email" placeholder="Email" required>

<input type="number" name="age" placeholder="Age" required>

<input type="text" name="course" placeholder="Course" required>

<button type="submit">Add Student</button>

</form>

<a href="/students">Back to Students List</a>

</body>

</html>

students.ejs

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Students List</title>

</head>

<body>

<h1>Students List</h1>

<ul>

<% students.forEach(function(student) { %>

<li><%= student.name %> - <%= student.email %> - <%= student.age %> - <%= student.course %></li>

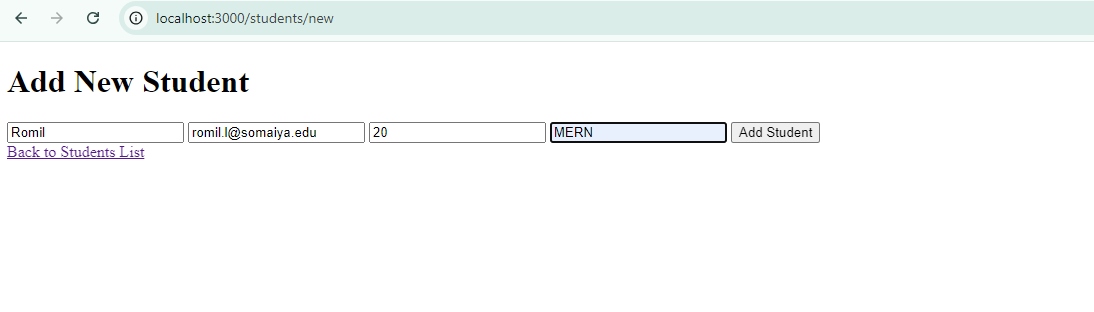
<% }) %>

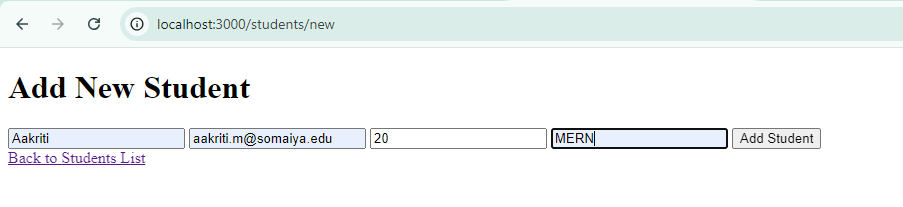
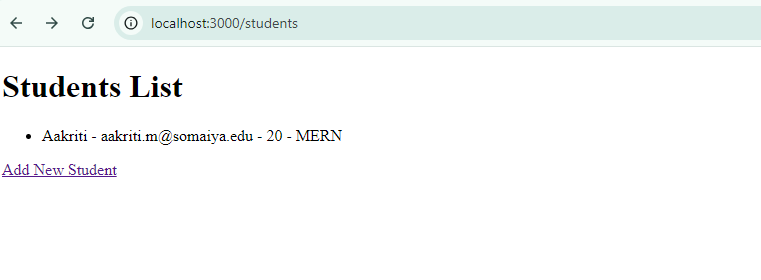
</ul>

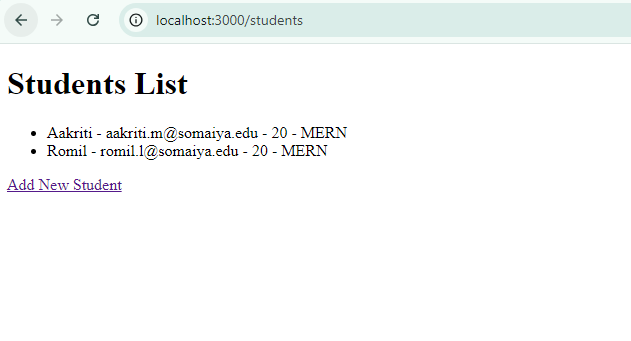
<a href="/students/new">Add New Student</a>

</body>

</html>

****

**  
**

**Conclusion:**Implementing MongoDB enhances data management efficiency, scalability, and flexibility, providing robust solutions for modern application requirements.