# EX.NO:3 DATE:

**STUDENT REGISTRATIONS SYSTEM USING NODE.JS, EXPRESS, AND MONGODB (ATLAS)**

**AIM:**

To create a web application that allows users to Add, Update, and Delete student details

using an HTML form, storing the data in MongoDB Atlas.

**PROCEDURE:**

**STEP 1**: Start the process and create a main folder (LAB-3).

**STEP 2**: Inside Ex3, create a subfolder named views to hold frontend EJS templates (e.g., form.ejs).

**STEP 3**: Create the MongoDB Atlas cluster and connect it using a connection URI inside server.js via Mongoose.

**STEP 4**: Set up the Express server, configure middleware, and connect Mongoose to MongoDB Atlas.

**STEP 5**: Create a Mongoose schema and model for storing student details (name, email, age).

**STEP 6:** Design a form using EJS to add and display all student data (form.ejs).

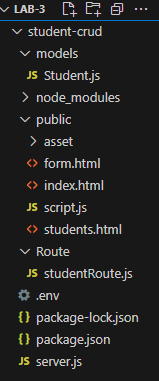
**STEP 7:** Define routes in server.js to handle adding, updating, and deleting students.

**STEP 8:** Apply logic in form.ejs to loop through student data and attach update/delete functionality.

**STEP 9:** Run the server using node server.js and open [http://localhost:3000](http://localhost:3000/) to test.

**STEP 10**: Once working, verify database entries in MongoDB Atlas and stop the server using Ctrl + C.

# DESIGN :

****

1. STUDENT CRUD/
   * Root directory of the project.
2. iews/ folder
   * Contains EJS template (form.ejs) to dynamically display and interact with student data.
   * Used to render student form and list using Express.
3. public/ folder
   * Contains style.css file to style the student form and list layout.
   * Enhances visual design and user experience.
4. app.js
   * Main Node.js server script.
   * Handles database connection, CRUD operations, and EJS rendering.
   * Connects to MongoDB Atlas using Mongoose.

# CODING:

## Form.html:

<!DOCTYPE html>

<html>

<head>

<title>Student Form</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">

</head>

<body class="container py-5">

<h1 class="mb-4">Register Student</h1>

<form action="/students" method="POST" class="mb-3">

<div class="mb-3">

<input name="name" class="form-control" placeholder="Name" required>

</div>

<div class="mb-3">

<input name="email" type="email" class="form-control" placeholder="Email" required>

</div>

<div class="mb-3">

<input name="course" class="form-control" placeholder="Course" required>

</div>

<div class="mb-3">

<input name="age" type="number" class="form-control" placeholder="Age" required>

</div>

<button type="submit" class="btn btn-primary">Register</button>

</form>

<a href="/students.html" class="btn btn-secondary">View All Students</a>

</body>

</html>

## Student.html

<!DOCTYPE html>

<html>

<head>

<title>Student List</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">

</head>

<body class="container py-5">

<h1 class="mb-4">All Students</h1>

<table class="table table-bordered">

<thead class="table-dark">

<tr>

<th>Name</th>

<th>Email</th>

<th>Course</th>

<th>Age</th>

<th>Actions</th>

</tr>

</thead>

<tbody id="studentTable"></tbody> </table>

<h2 class="mt-5">Update Student</h2>

<form id="updateForm" class="mb-3">

<input type="hidden" id="updateId">

<div class="mb-2"><input class="form-control" id="updateName" placeholder="Name" required></div>

<div class="mb-2"><input class="form-control" id="updateEmail" placeholder="Email" required></div>

<div class="mb-2"><input class="form-control" id="updateCourse" placeholder="Course" required></div>

<div class="mb-2"><input class="form-control" id="updateAge" placeholder="Age" required></div>

<button class="btn btn-success">Update</button>

</form>

<a href="/form.html" class="btn btn-secondary">Back to Registration</a>

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

</body>

</html>

## Script.js

fetch('/students')

.then(res => res.json())

.then(data => {

const table = document.getElementById('studentTable');

data.forEach(s => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${s.name}</td>

<td>${s.email}</td>

<td>${s.course}</td>

<td>${s.age}</td>

<td>

<button onclick="edit('${s.\_id}', '${s.name}', '${s.email}', '${s.course}', ${s.age})">Edit</button>

<button onclick="remove('${s.\_id}')">Delete</button>

</td>`;

table.appendChild(row);

});

});

function edit(id, name, email, course, age) {

document.getElementById('updateId').value = id;

document.getElementById('updateName').value = name;

document.getElementById('updateEmail').value = email;

document.getElementById('updateCourse').value = course;

document.getElementById('updateAge').value = age;

}

function remove(id) {

fetch(`/students/${id}`, { method: 'DELETE' })

.then(() => location.reload());

}

document.getElementById('updateForm').addEventListener('submit', e => {

e.preventDefault();

const id = document.getElementById('updateId').value;

const updated = {

name: document.getElementById('updateName').value,

email: document.getElementById('updateEmail').value,

course: document.getElementById('updateCourse').value,

age: document.getElementById('updateAge').value

};

fetch(`/students/${id}`, {

method: 'PUT',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(updated)

}).then(() => location.reload()); });

## Server.js

const express = require('express');

const mongoose = require('mongoose');

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

const path = require('path');

require('dotenv').config();

const app = express();

const PORT = 3000;

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

app.use(bodyParser.json());

app.use(express.static(path.join(\_\_dirname, 'public')));

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

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

.catch(err => console.error(err));

const Student = require('./models/Student');

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

try {

const student = new Student(req.body);

await student.save();

res.redirect('/students.html');

} catch (err) {

res.status(400).send(err);

}

});

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

const students = await Student.find();

res.json(students);

});

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

try {

const updated = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true });

res.json(updated);

} catch (err) {

res.status(400).send(err); }

});

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

try {

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

res.json({ message: 'Deleted' });

} catch (err) {

res.status(400).send(err); }

});

app.get('/', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

app.listen(PORT, () => {

console.log(`Server running on [http://localhost:${PORT}`](http://localhost:$%7bPORT%7d%60)); });

## STUDENT.js

const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({

name: String,

email: String,

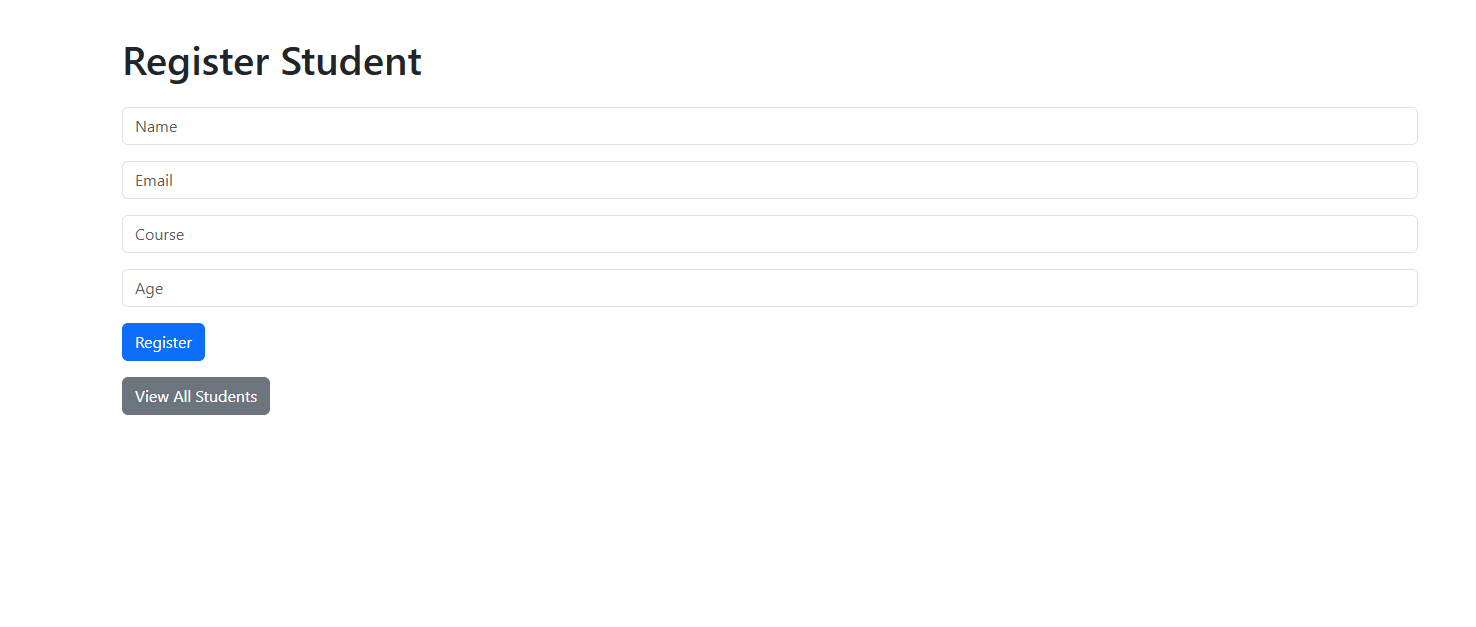
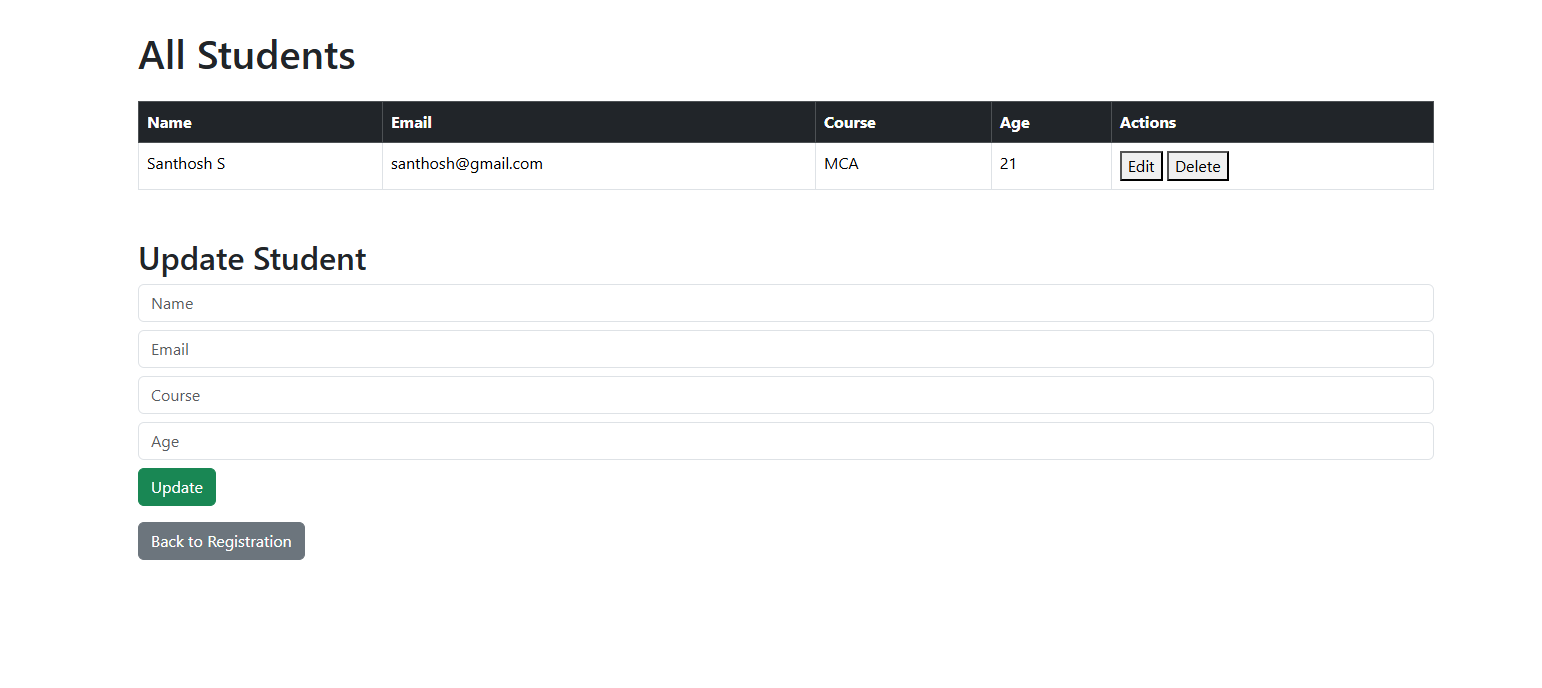
course: String,

age: Number

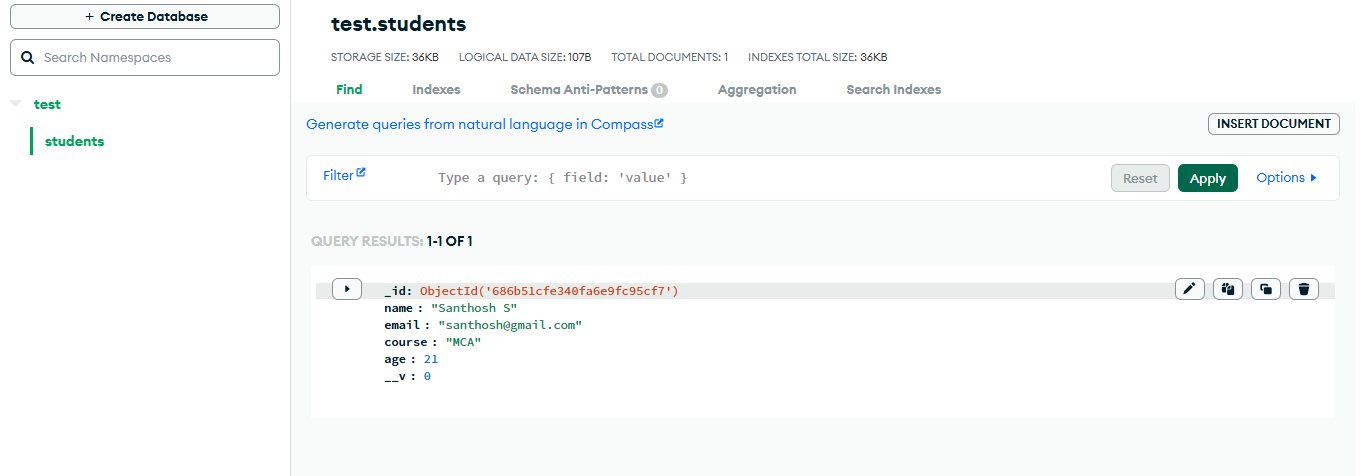
});

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

# OUTPUT:

****

****

****

|  |  |
| --- | --- |
| COE (30) |  |
| OBSERVATION(10) |  |
| RECORD (10) |  |
| VIVA (10) |  |
| TOTAL (60) |  |

**RESULT:**

Successfully performed all CRUD operations. Student records were created, updated, deleted, and retrieved from the MongoDB Atlas database using a Node.js + Express server.