|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment No. – 8** | | | | |
| **Date of Performance:** | **9/4/25** | | | |
| **Date of Submission:** | **16/4/25** | | | |
| Program Execution/  formation/  correction/  ethical practices  (06) | Timely  Submission  (01) | Viva  (03) | Experiment  Total (10) | Sign with Date |
|  |  |  |  |  |

**Experiment No. 8**

**Title:** RESTful API Integration using MongoDB

**1.1 Aim:** To implement and integrate RESTful APIs with MongoDB for data storage and retrieval operations.

**1.2 Course Outcome(CO-3):** RESTful API integration using MongoDB

**1.3 Learning Objectives:**

* Understand how to create RESTful APIs using Node,js and Express,js
* Learn how to connect and interact with MongoDB using Mongosoe
* Perform basic CRUD operations (Create, Read, Update, Delete) through APIs
* Gain practical experience in building backend services using for web applications
* Learn how frontend and backend communicate using API’s

**1.4 Requirement:**

**Hardware:**

* A computer or laptop with internet

**Software:**

* Node.js installed
* MongoDB installed (or MongoDB Compass)

**Tools Used:**

* Visual Studio Code (or any code editor)
* Web Browser for testing

**Node Modules:**

* express – to create server and APIs
* mongoose – to connect and work with MongoDB

1.5 Related Theory:

What is a RESTful API?

A RESTful API is an application programming interface that adheres to the principles of REST (Representational State Transfer) architecture. It allows clients to communicate with servers through HTTP methods (GET, POST, PUT, DELETE) and is widely used in web services for data exchange.

**What is MongoDB?**

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It offers scalability and flexibility for handling large volumes of structured or unstructured data.

**Key Concepts:**

* **Node.js**: JavaScript runtime used for building scalable network applications.
* **Express.js**: Lightweight web framework for Node.js to simplify API development.
* **MongoDB**: NoSQL database used to store and retrieve JSON-style documents.
* **Mongoose**: ODM (Object Data Modelling) library for MongoDB and Node.js.

**Tools and Technologies:**

* **Node.js** – JavaScript runtime environment
* **Express.js** – Web framework for Node.js
* **MongoDB** – NoSQL database
* **Mongoose** – ODM (Object Data Modelling) library for MongoDB

**1.6 Procedure:**

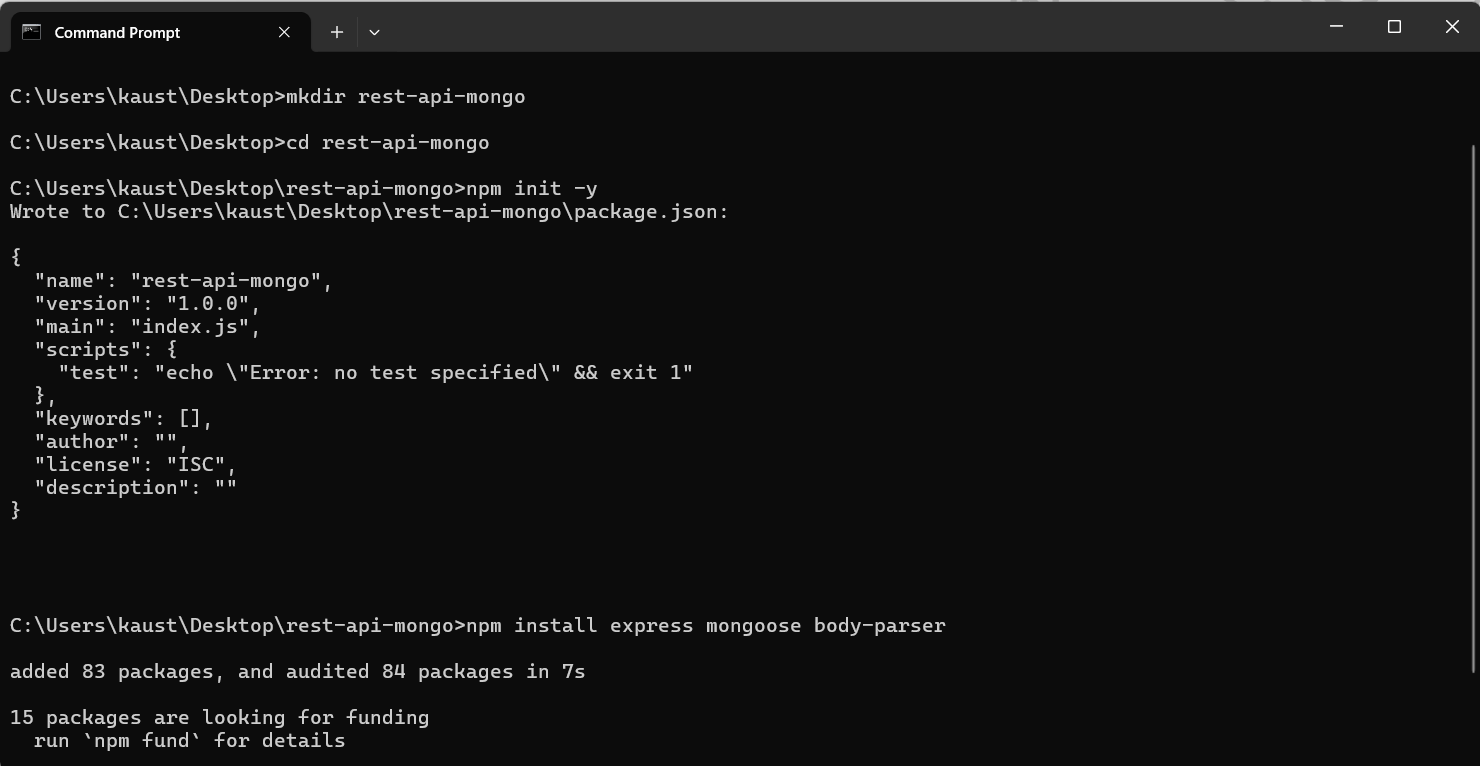
**Step 1: Initialize Project**

mkdir rest-api-mongo

cd rest-api-mongo

npm init -y

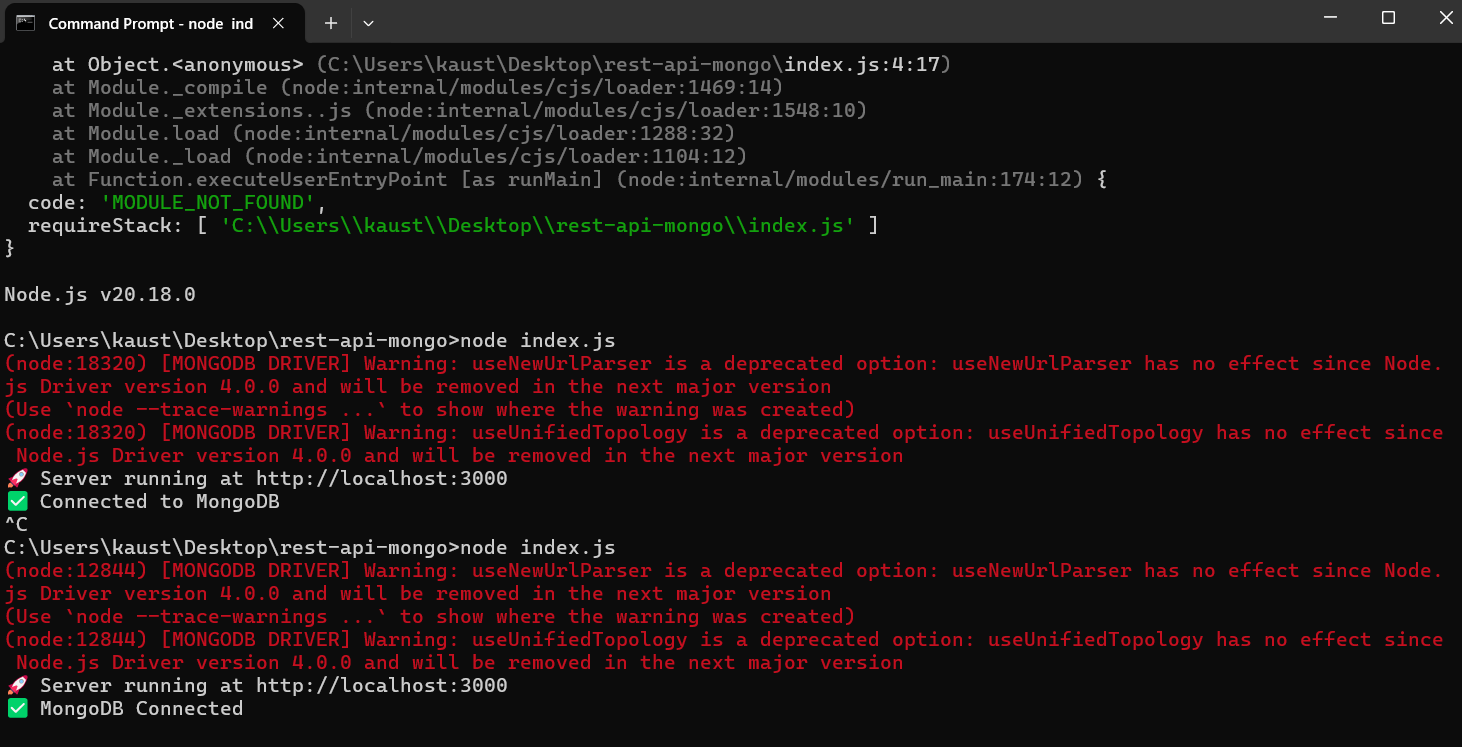
npm install express mongoose body-parser



**Step 2: Create Server File**

**Step 3: Run Server**

node index.js



**Step 4: Test**

Open your browser and visit:

* http://localhost:3000/ → Welcome message
* http://localhost:3000/add → Adds a sample student
* http://localhost:3000/students → Lists all students
* http://localhost:3000/delete/<student\_id> → Deletes a student (replace <student\_id> with actual MongoDB ID)

**1.7 Program and Output:**

**index.js**

const express = require('express');

const mongoose = require('mongoose');

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

const app = express();

// MongoDB Connection

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

useNewUrlParser: true,

useUnifiedTopology: true,

})

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

.catch(err => console.error('MongoDB connection error:', err));

// Home Route

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

res.send(`

<h2>Welcome to Student API</h2>

<ul>

<li><a href="/add">Add Sample Student</a></li>

<li><a href="/students">View All Students</a></li>

<li>To delete: /delete/&lt;id&gt;</li>

</ul>

`);

});

// Add a student (Sample Data)

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

const student = new Student({

name: "Koustubh",

age: 21,

course: "Information Technology"

});

await student.save();

res.send(`<p>Student Added Successfully</p><a href="/">Back</a>`);

});

// View all students

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

const students = await Student.find();

let list = students.map(s => `<li>${s.name} (${s.\_id})</li>`).join('');

res.send(`<h3>All Students</h3><ul>${list}</ul><a href="/">Back</a>`);

});

// Delete a student by ID

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

try {

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

res.send(`<p>Student Deleted</p><a href="/">Back</a>`);

} catch (err) {

res.send(`<p>Error: ${err.message}</p><a href="/">Back</a>`);

}

});

// Start server

app.listen(3000, () => {

console.log('Server running at http://localhost:3000');

});

**models/Student.js**

const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({

    name: String,

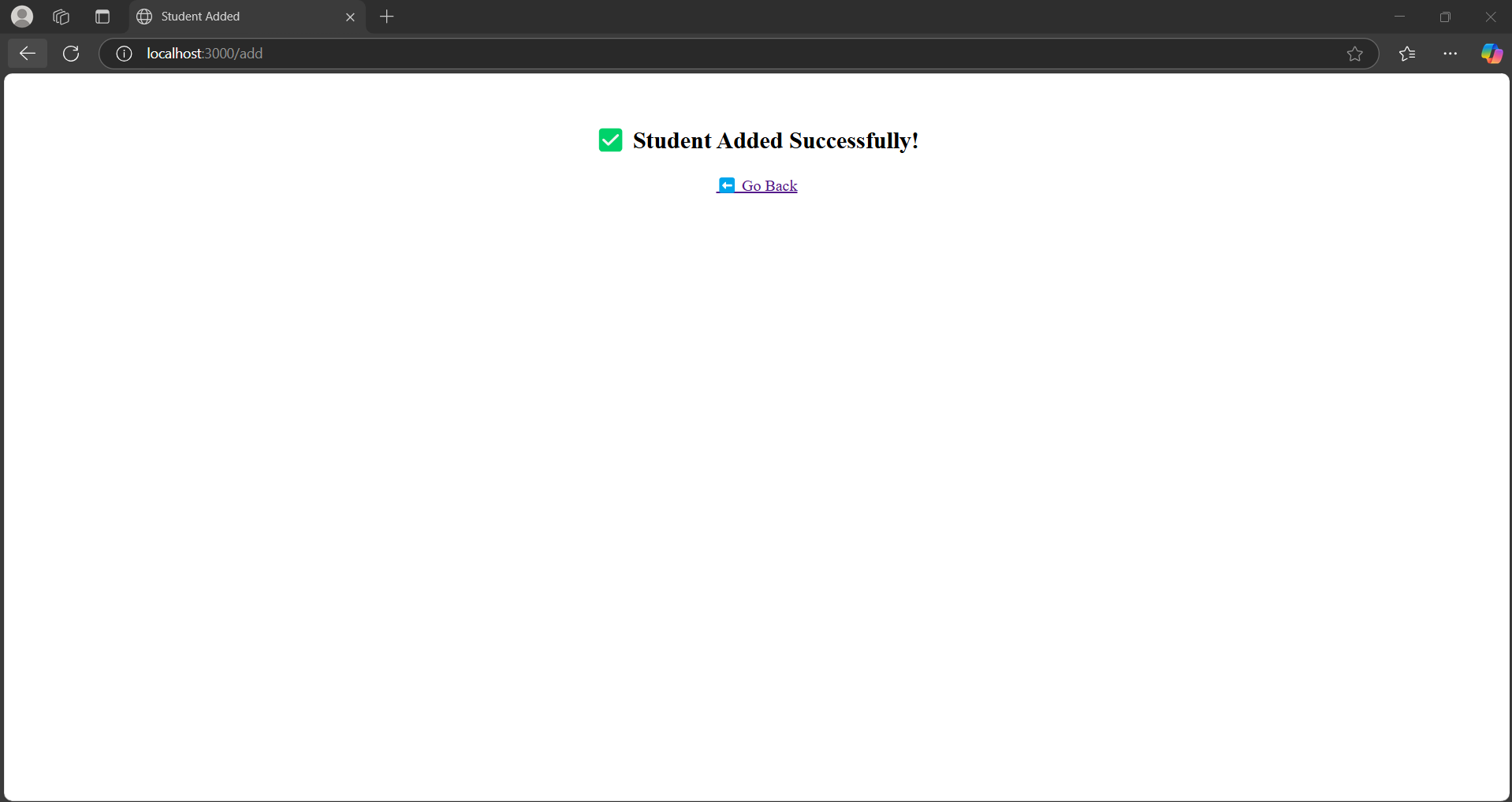
    age: Number,

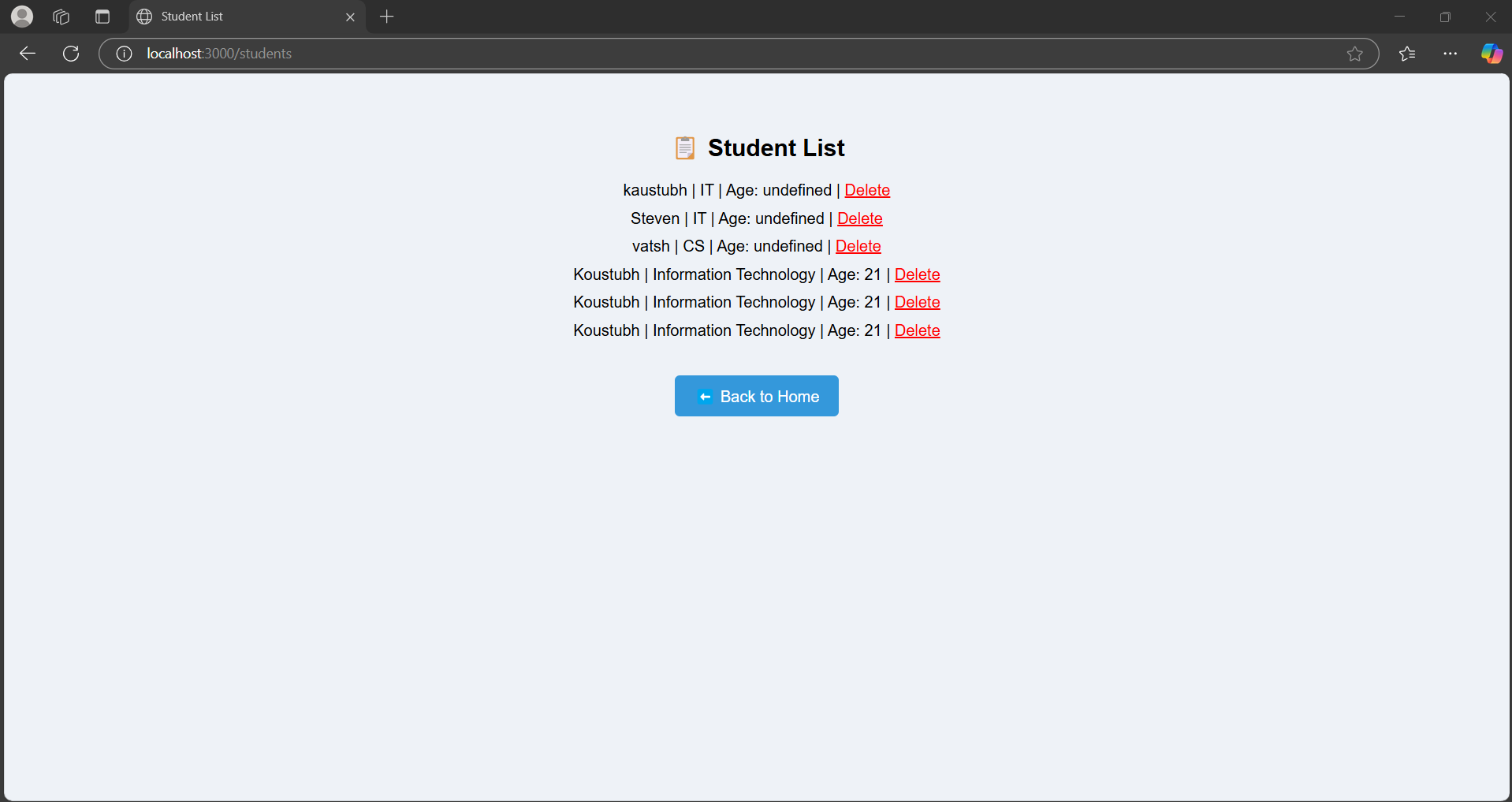
    course: String

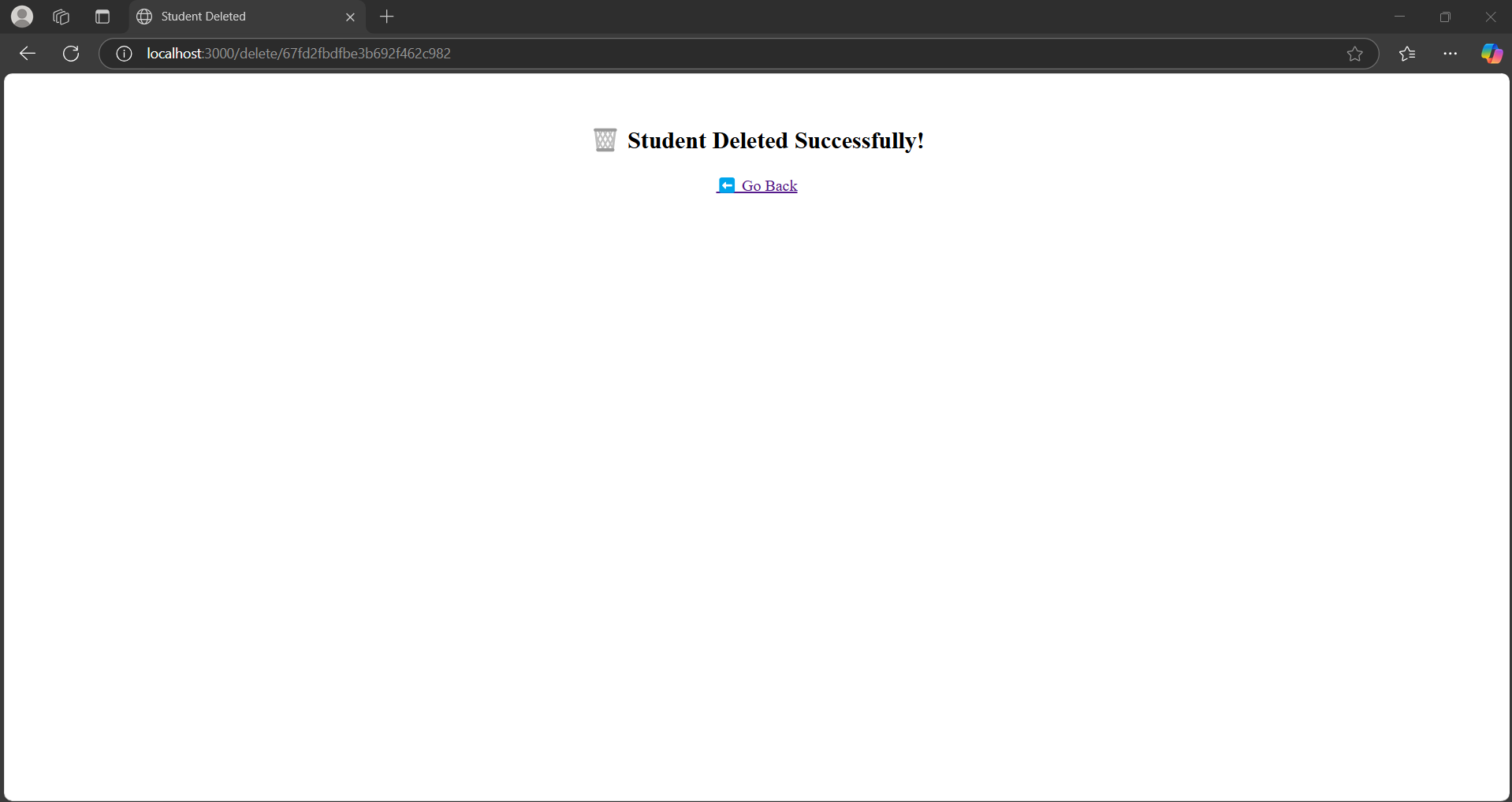
});

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

**Output:**







**1.8 Conclusion:**

This experiment demonstrates how RESTful APIs can be built using Node.js and Express, and how they interact with MongoDB for CRUD operations. This setup forms the foundation of modern web applications where frontend and backend communicate seamlessly through APIs.

MongoDB's schema-less nature and ease of integration with JavaScript make it ideal for developing fast and scalable REST APIs.

**1.9 Questions:**

**1. What is a RESTful API and why is it used in web development?**

**Answer:**

A RESTful API (Representational State Transfer) is a type of web service that follows the REST architectural principles. It uses standard HTTP methods like GET, POST, PUT, and DELETE to perform operations on data. RESTful APIs are widely used in web development because they allow seamless communication between the client (frontend) and the server (backend), enabling the exchange of data in a structured and scalable way.

**2. How is MongoDB integrated into a Node.js application using Mongoose?**

**Answer:**

MongoDB is integrated into a Node.js application using the Mongoose library, which provides a schema-based solution for modeling data. The process involves:

Installing Mongoose using npm.

Connecting to a MongoDB database using mongoose.connect().

Defining a schema and model for the collection (e.g., Student).

Using Mongoose methods such as save(), find(), and findByIdAndDelete() to perform CRUD operations.

**Q3. What are the steps involved in testing a RESTful API built with Node.js and MongoDB?**

**Answer:**

To test a RESTful API built with Node.js and MongoDB, follow these steps:

Start the server using node index.js.

Open a browser or API client like Postman.

Navigate to http://localhost:3000/ to access the home page.

Use http://localhost:3000/add to add a sample student.

Visit http://localhost:3000/students to view all students.

Use http://localhost:3000/delete/<student\_id> to delete a specific student by ID.