**Name :Nikhil**

**UID : 23BCC70030**

**Class : 23BCC-(A)**

**Experiment 4.2**

**Aim:**

To develop a Student Management System using MongoDB as a database and MVC architecture for better organization of code, allowing CRUD (Create, Read, Update, Delete) operations on student records.

**Theory:**

* **MVC Architecture:**
  + **Model:** Handles database interactions (MongoDB collections).
  + **View:** Displays data to users (in this case, we can use EJS templates or JSON responses).
  + **Controller:** Processes input, interacts with the model, and returns the response.
* **MongoDB:**
  + A NoSQL database used to store student information.
  + Stores data in JSON-like documents for easy storage and retrieval.
* **Operations in Student Management System:**
  + Add new student
  + View all students
  + Update student details
  + Delete a student

**Procedure:**

1. Install Node.js, MongoDB, and required npm packages (express, mongoose, body-parser).
2. Create a single Node.js file with MVC structure:
   * Define Student model (Schema)
   * Define controller functions for CRUD
   * Define routes to handle HTTP requests
3. Connect to MongoDB using Mongoose.
4. Start the Express server and test endpoints using Postman or browser.

**Code:**

// ================== Import Packages ==================

const express = require('express');

const mongoose = require('mongoose');

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

// ================== Initialize App ==================

const app = express();

app.use(bodyParser.json());

// ================== MongoDB Connection ==================

mongoose.connect('mongodb://127.0.0.1:27017/studentDB', {

useNewUrlParser: true,

useUnifiedTopology: true

})

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

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

// ================== Model ==================

const studentSchema = new mongoose.Schema({

name: { type: String, required: true },

age: { type: Number, required: true },

grade: { type: String, required: true },

email: { type: String, required: true }

});

const Student = mongoose.model('Student', studentSchema);

// ================== Controller ==================

const studentController = {

// Create a new student

createStudent: async (req, res) => {

try {

const student = new Student(req.body);

const savedStudent = await student.save();

res.status(201).json(savedStudent);

} catch (err) {

res.status(400).json({ message: err.message });

}

},

// Get all students

getAllStudents: async (req, res) => {

try {

const students = await Student.find();

res.status(200).json(students);

} catch (err) {

res.status(500).json({ message: err.message });

}

},

// Get a student by ID

getStudentById: async (req, res) => {

try {

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

if (!student) return res.status(404).json({ message: "Student not found" });

res.status(200).json(student);

} catch (err) {

res.status(500).json({ message: err.message });

}

},

// Update a student

updateStudent: async (req, res) => {

try {

const updatedStudent = await Student.findByIdAndUpdate(

req.params.id,

req.body,

{ new: true }

);

if (!updatedStudent) return res.status(404).json({ message: "Student not found" });

res.status(200).json(updatedStudent);

} catch (err) {

res.status(400).json({ message: err.message });

}

},

// Delete a student

deleteStudent: async (req, res) => {

try {

const deletedStudent = await Student.findByIdAndDelete(req.params.id);

if (!deletedStudent) return res.status(404).json({ message: "Student not found" });

res.status(200).json({ message: "Student deleted successfully" });

} catch (err) {

res.status(500).json({ message: err.message });

}

}

};

// ================== Routes ==================

app.post('/students', studentController.createStudent);

app.get('/students', studentController.getAllStudents);

app.get('/students/:id', studentController.getStudentById);

app.put('/students/:id', studentController.updateStudent);

app.delete('/students/:id', studentController.deleteStudent);

// ================== Start Server ==================

const PORT = 3000;

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**Learning Outcomes:**

1. Understand the MVC architecture and its implementation in Node.js.
2. Learn to perform CRUD operations with MongoDB using Mongoose.
3. Gain experience in connecting Node.js with MongoDB and handling data.
4. Understand how to structure controllers and routes for clean code.
5. Learn to test API endpoints using Postman or browser.