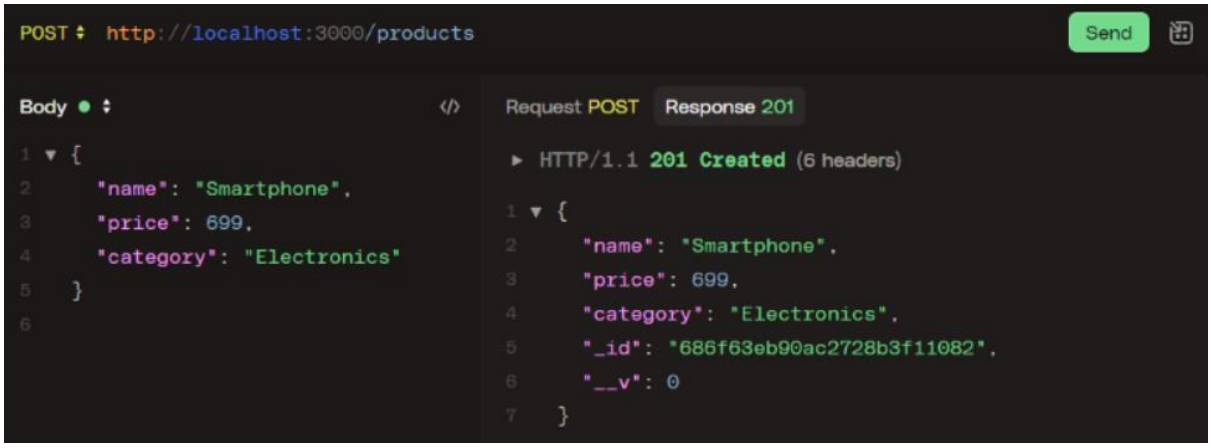


EXPERIMENT 5.2

Title

Student Management System Using MongoDB and MVC Architecture

Objective

Learn how to design and build a Node.js application using the Model-View-Controller (MVC) architecture to manage student data stored in MongoDB. This task helps you understand how to separate concerns, structure backend logic clearly, and interact with a database using Mongoose.

Task Description

Create a student management system using Node.js, Express.js, and MongoDB (with Mongoose). Define a Student model with properties like name, age, and course. Implement a controller to handle CRUD operations (create, read, update, delete) on student data. Set up routes to connect client requests to the appropriate controller methods. Use Mongoose to handle all database interactions. Organize your codebase into separate folders for models, controllers, and routes to follow MVC principles clearly.

CODE:

// ====== app.js ====== const express =

require('express'); const mongoose =

require('mongoose'); const studentRoutes =

require('./routes/studentRoutes'); const app = express();

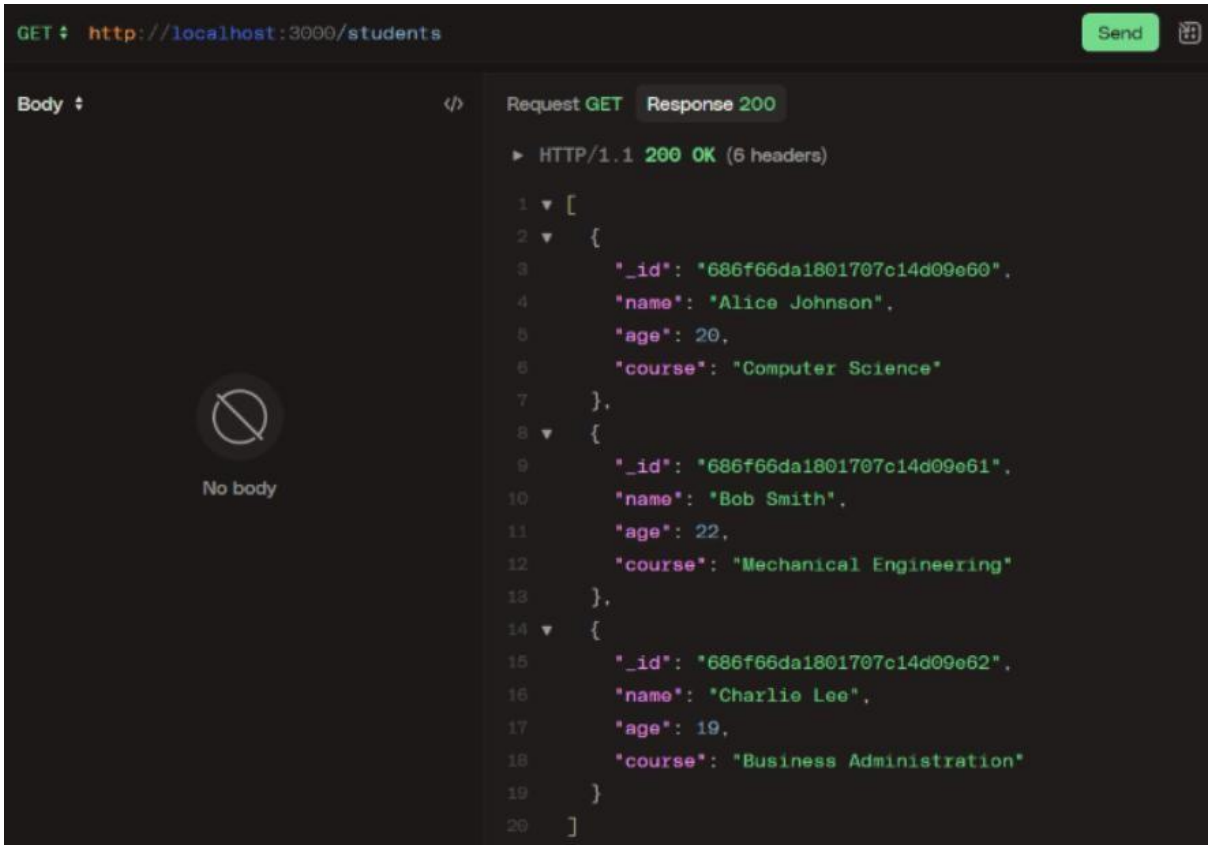
app.use(express.json());

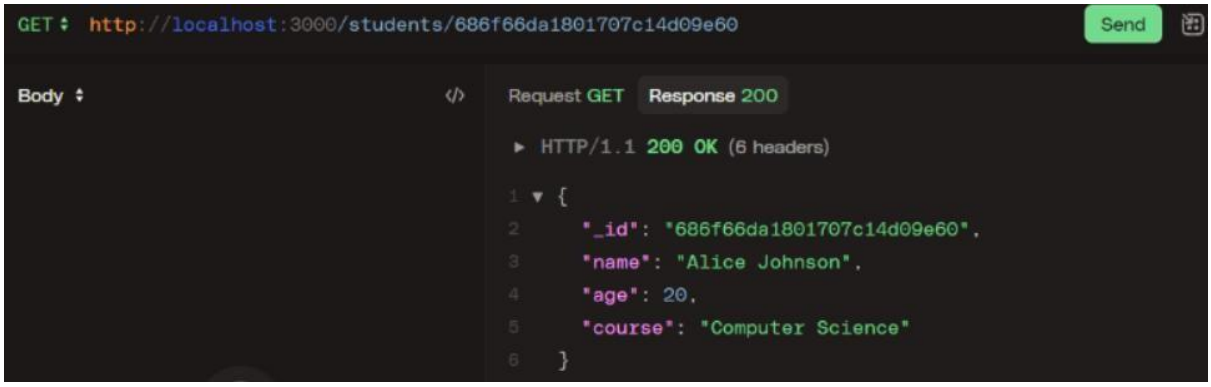
```
mongoose.connect('mongodb://localhost:27017/studentDB', {
useNewUrlParser: true,   useUnifiedTopology: true
});

app.use('/students', studentRoutes);

app.listen(3000, () => {   console.log('Server
running on port 3000');
});

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

const studentSchema = new mongoose.Schema({
  name: { type: String, required: true },
age: { type: Number, required: true },
course: { type: String, required: true }
});

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



// ====== controllers/studentController.js ====== const
Student = require('../models/Student');

exports.createStudent = async (req, res) => {
```

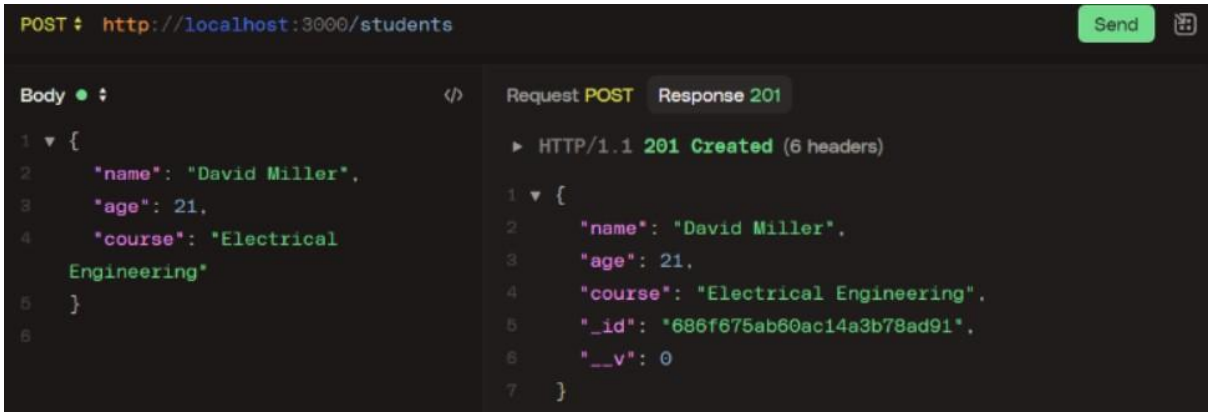
```
  try {
    const student = new Student(req.body);
await student.save();
res.status(201).json(student);
  } catch (err) {     res.status(400).json({ error:
err.message });
  }
};

exports.getAllStudents = async (req, res) => {
  try {
    const students = await Student.find();
res.json(students);   } catch (err) {
res.status(500).json({ error: err.message });
  }
};

exports.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.json(student);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.updateStudent = async (req, res) => {
```

```
  try {
    const student = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true });
if (!student) return res.status(404).json({ message: 'Student not found' });
res.json(student);   } catch (err) {     res.status(400).json({ error: err.message });
  }
};

exports.deleteStudent = async (req, res) => {
  try {
    const student = await Student.findByIdAndDelete(req.params.id);     if
(!student) return res.status(404).json({ message: 'Student not found' });
res.json({ message: 'Student deleted successfully' });
  } catch (err) {     res.status(500).json({ error:
err.message });
  }
};
```

OUTPUT :


GET http://localhost:3000/students
Send
Body
Request GET
Response 200
HTTP/1.1 200 OK (6 headers)
No body
[
{
"_id": "686f66da1801707c14d09e60",
"name": "Alice Johnson",
"age": 20,
"course": "Computer Science"
},
{
"_id": "686f66da1801707c14d09e61",
"name": "Bob Smith",
"age": 22,
"course": "Mechanical Engineering"
},
{
"_id": "686f66da1801707c14d09e62",
"name": "Charlie Lee",
"age": 19,
"course": "Business Administration"
}
]


GET http://localhost:3000/students/686f66da1801707c14d09e60
Send
Body
Request GET
Response 200
HTTP/1.1 200 OK (6 headers)
{
"_id": "686f66da1801707c14d09e60",
"name": "Alice Johnson",
"age": 20,
"course": "Computer Science"
}


POST http://localhost:3000/students
Send
Body
Request POST
Response 201
{
"name": "David Miller",
"age": 21,
"course": "Electrical Engineering"
}
HTTP/1.1 201 Created (6 headers)
{
"name": "David Miller",
"age": 21,
"course": "Electrical Engineering",
"_id": "686f675ab60ac14a3b78ad91",
"__v": 0
}