EXP-4.2:

AIM:
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 clear

CODE-

1. Setup Project and Install Dependencies:

Create a folder and initialize Node.js project:

```
mkdir student-management
cd student-management
npm init -y
npm install express mongoose body-parser
```

2. Project Structure:

Create folders and files:

```
student-management/
│
├── models/
│   └── student.js
├── controllers/
│   └── studentController.js
├── routes/
│   └── studentRoutes.js
├── app.js
└── package.json
```

3. Define Student Model (models/student.js):

Create 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;
```

4. IImplement Controller (controllers/studentController.js):

```
const Student = require('../models/student');
// Create a new student
exports.createStudent = async (req, res) => {
  try {
    const { name, age, course } = req.body;
    const student = new Student({ name, age, course });
    await student.save();
    res.status(201).json(student);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
// Get all students
exports.getAllStudents = async (req, res) => {
  try {
    const students = await Student.find();
    res.json(students);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

```
// Get student by ID
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 (error) {
    res.status(500).json({ message: error.message });
  }
};
// Update student by ID
exports.updateStudent = async (req, res) => {
  try {
    const { name, age, course } = req.body;
    const student = await Student.findByIdAndUpdate(
      req.params.id,
      { name, age, course },
      { new: true }
    );
    if (!student) return res.status(404).json({ message: 'Student not found' });
    res.json(student);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
// Delete student by ID
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 (error) {
    res.status(500).json({ message: error.message });}};
```

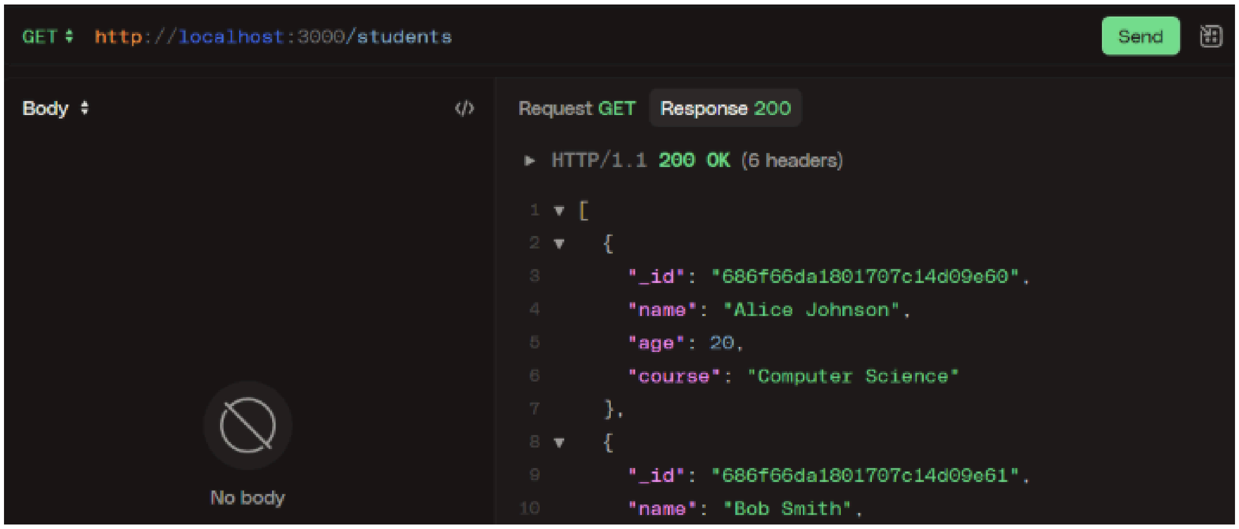
5. Setup Project and Install Dependencies:

```
const express = require('express');
const router = express.Router();
const studentController = require('../controllers/studentController');
router.post('/', studentController.createStudent);
router.get('/', studentController.getAllStudents);
router.get('/:id', studentController.getStudentById);
router.put('/:id', studentController.updateStudent);
router.delete('/:id', studentController.deleteStudent);
module.exports = router;
```

6. Create Main App File (app.js):

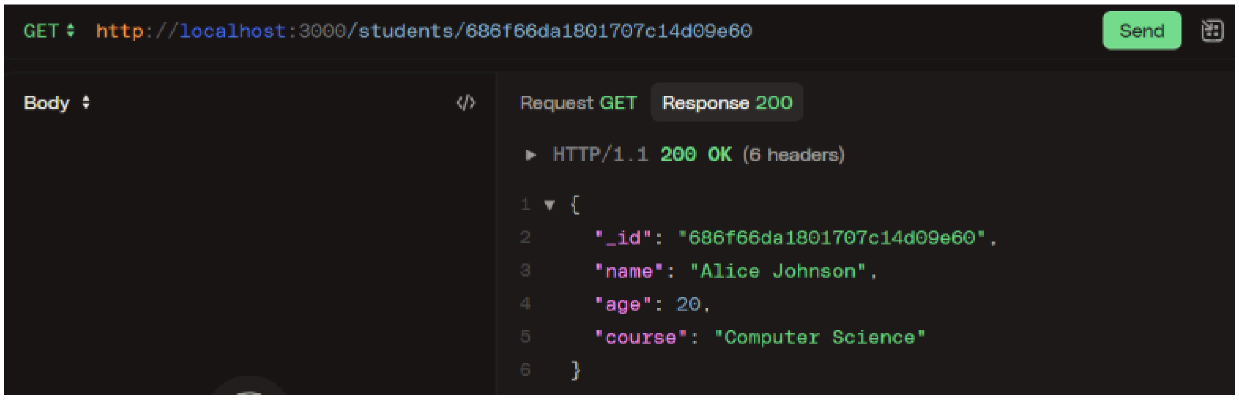
```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const studentRoutes = require('./routes/studentRoutes');
const app = express();
const PORT = process.env.PORT || 3000;
// Middleware
app.use(bodyParser.json());
// Connect to MongoDB (replace <your_mongodb_uri> with your connection
string)
mongoose.connect('<your_mongodb_uri>', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('Connected to MongoDB'))
  .catch((err) => console.error('MongoDB connection error:', err));
// Routes
app.use('/students', studentRoutes);
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

OUTPUTS:



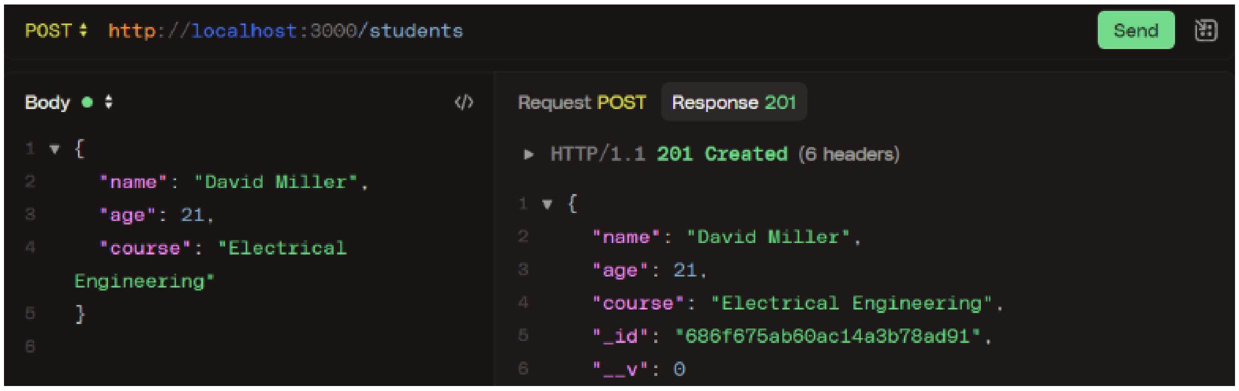
```
GET http://localhost:3000/students
```

```
HTTP/1.1 200 OK (6 headers)
[
  {
    "_id": "686f66da1801707c14d09e60",
    "name": "Alice Johnson",
    "age": 20,
    "course": "Computer Science"
  },
  {
    "_id": "686f66da1801707c14d09e61",
    "name": "Bob Smith",
```



```
GET http://localhost:3000/students/686f66da1801707c14d09e60
```

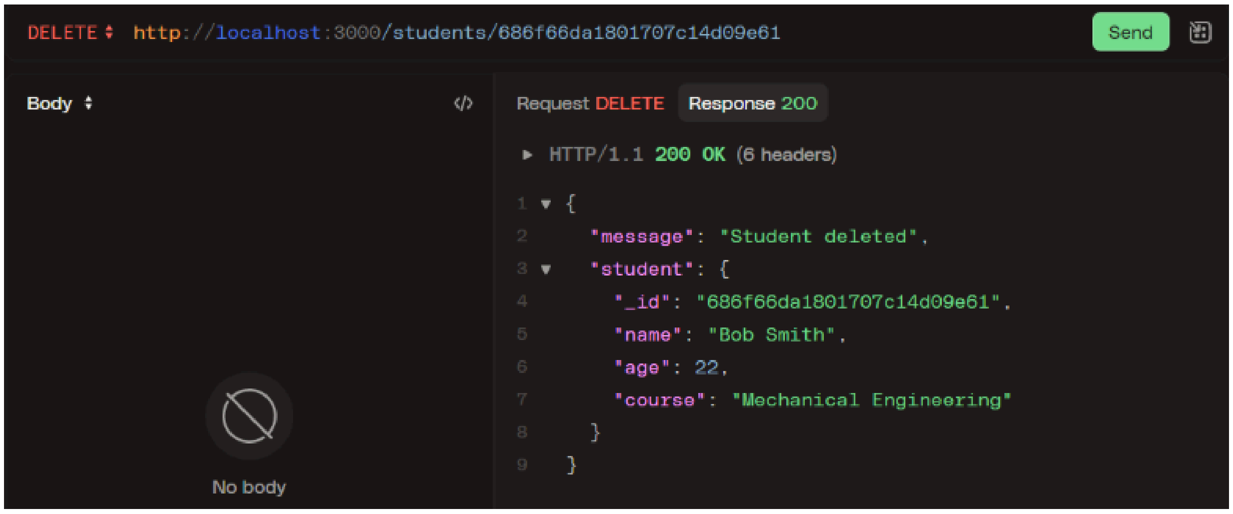
```
HTTP/1.1 200 OK (6 headers)
{
  "_id": "686f66da1801707c14d09e60",
  "name": "Alice Johnson",
  "age": 20,
  "course": "Computer Science"
}
```



```
POST http://localhost:3000/students
```

```
{
  "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
```



```
DELETE http://localhost:3000/students/686f66da1801707c14d09e61
```

```
HTTP/1.1 200 OK (6 headers)
{
  "message": "Student deleted",
  "student": {
    "_id": "686f66da1801707c14d09e61",
    "name": "Bob Smith",
    "age": 22,
    "course": "Mechanical Engineering"
  }
}
```