NAME	Taimur Kamran		
UID	23BCS10578		
CLASS	622-A		

- ➤ NodeJs PRACTISE 5.1
- CODE

Server.js (Main server):

```
const express = require('express');
const mongoose = require('mongoose');
const studentRoutes = require('./routes/studentRoutes');
const app = express();
app.use(express.json());
// MongoDB Connection
mongoose.connect('mongodb+srv://alpha:alpha2025@cluster0.gttfsb5.mongodb.net/?ret
ryWrites=true&w=majority&appName=Cluster0')
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('MongoDB connection error:', err.message));
// Routes
app.use('/students', studentRoutes);
const PORT = 3000;
app.listen(PORT, () => {
 console.log(`Server running on http://localhost:${PORT}`);
});
```

/models/Student.js:

```
// models/Student.js
const mongoose = require('mongoose');
const StudentSchema = new mongoose.Schema({
  name: {
   type: String,
    required: true,
   trim: true
 },
  age: {
   type: Number,
   required: true,
   min: 1
  },
 course: {
   type: String,
   required: true,
   trim: true
  },
 createdAt: {
   type: Date,
   default: Date.now
});
module.exports = mongoose.model('Student', StudentSchema);
```

/controller/studentController.js :

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

// Create new student
exports.createStudent = async (req, res) => {
   try {
     const { name, age, course } = req.body;
     const student = new Student({ name, age, course });
     const savedStudent = await student.save();
     res.status(201).json(savedStudent);
} 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 single 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 updatedStudent = await Student.findByIdAndUpdate(
      req.params.id,
      { name, age, course },
      { new: true }
    );
    if (!updatedStudent) return res.status(404).json({ message: 'Student not
found' });
   res.json(updatedStudent);
 } catch (error) {
    res.status(400).json({ message: error.message });
};
// Delete student by ID
exports.deleteStudent = async (req, res) => {
  try {
   const student = await Student.findById(req.params.id);
```

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

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

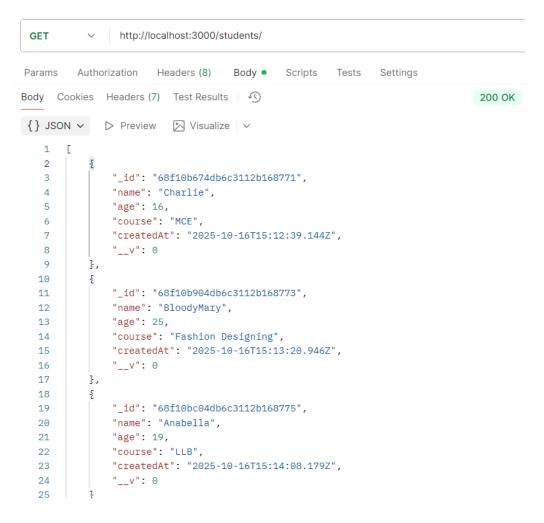
res.json({
    message: 'Student deleted successfully',
    deletedStudent: {
        id: student._id,
        name: student.name,
        age: student.age,
        course: student.course
    }
    });
} catch (error) {
    res.status(500).json({ message: error.message });
}
```

/routes/studentRoutes.js:

```
  // routes/studentRoutes.js
   const express = require('express');
   const router = express.Router();
   const studentController = require('../controllers/studentController');

   // CRUD Routes
   router.post('/add', studentController.createStudent);
   router.get('/', studentController.getAllStudents);
   router.get('/:id', studentController.getStudentById);
   router.put('/:id', studentController.updateStudent);
   router.delete('/:id', studentController.deleteStudent);
   module.exports = router;
```

OUTPUT



http://localhost:3000/students/68f10b904db6c3112b168773

```
GET
                                                                                                                     http://localhost:3000/students/68f10b904db6c3112b168773
   Params
                                                                 Authorization
                                                                                                                                                               Headers (8)
                                                                                                                                                                                                                                                            Body •
                                                                                                                                                                                                                                                                                                                        Scripts
                                                                                                                                                                                                                                                                                                                                                                                   Tests
                                                                                                                                                                                                                                                                                                                                                                                                                                         Settings
Body Cookies Headers (7) Test Results
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      200 OK
     {} JSON ~
                                                                                                  > Preview

    ∀ Visualize  
    ✓ Visualize  

    ✓ Visualize  
    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

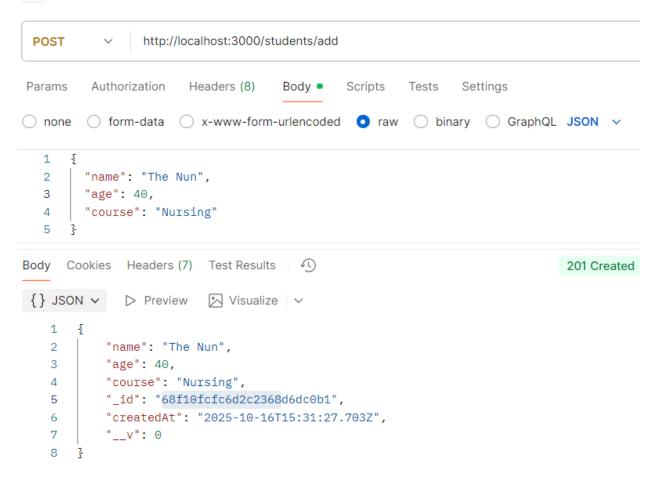
    ✓ Visualize  

    ✓ Visualize  

    ✓ Visualize  

     
                          1
                                                     £
                           2
                                                                                "_id": "68f10b904db6c3112b168773",
                           3
                                                                               "name": "BloodyMary",
                          4
                                                                                 "age": 25,
                           5
                                                                              "course": "Fashion Designing",
                                                                               "createdAt": "2025-10-16T15:13:20.946Z",
                           6
                                                                                "__v": 0
                           7
                           8
```

http://localhost:3000/students/add



Output Link (Postman): http://localhost:3000/students/

API's: GET http://localhost:3000/students/

GEThttp://localhost:3000/students/(id)

DELETE http://localhost:3000/students/(id)

POST http://localhost:3000/students/add