

NAME	Konika Khurana
UID	23BCS12925
CLASS	622-A

➤ NodeJs PRACTISE 5.2

- CODE

Server.js (Main server):

```
// server.js
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/?retryWrites=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);

// Start Server
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

GET

http://localhost:3000/students/

Params

Authorization

Headers (8)

Body ●

Scripts

Tests

Settings

Body

Cookies

Headers (7)

Test Results

🔄

200 OK


{ } JSON

▶ Preview

🖼 Visualize

▼

```
1  [
2    {
3      "_id": "68f10b674db6c3112b168771",
4      "name": "Charlie",
5      "age": 16,
6      "course": "MCE",
7      "createdAt": "2025-10-16T15:12:39.144Z",
8      "__v": 0
9    },
10   {
11     "_id": "68f10b904db6c3112b168773",
12     "name": "BloodyMary",
13     "age": 25,
14     "course": "Fashion Designing",
15     "createdAt": "2025-10-16T15:13:20.946Z",
16     "__v": 0
17   },
18   {
19     "_id": "68f10bc04db6c3112b168775",
20     "name": "Anabella",
21     "age": 19,
22     "course": "LLB",
23     "createdAt": "2025-10-16T15:14:08.179Z",
24     "__v": 0
25   }
26 ]
```

 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

▼

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

HTTP <http://localhost:3000/students/add>

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

Params Authorization Headers (8) **Body** Scripts Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ☐

```
1 {
2   "name": "The Nun",
3   "age": 40,
4   "course": "Nursing"
5 }
```

Body Cookies Headers (7) Test Results **201 Created**

{} JSON ☐ Preview Visualize ☐

```
1 {
2   "name": "The Nun",
3   "age": 40,
4   "course": "Nursing",
5   "_id": "68f10fcfc6d2c2368d6dc0b1",
6   "createdAt": "2025-10-16T15:31:27.703Z",
7   "__v": 0
8 }
```

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

API's: **GET** <http://localhost:3000/students/>

GET [http://localhost:3000/students/\(id\)](http://localhost:3000/students/(id))

DELETE [http://localhost:3000/students/\(id\)](http://localhost:3000/students/(id))

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

