Ex. No: 8	**Building a REST API with Express, Node, and MongoDB**
9.10.2023	

Aim:

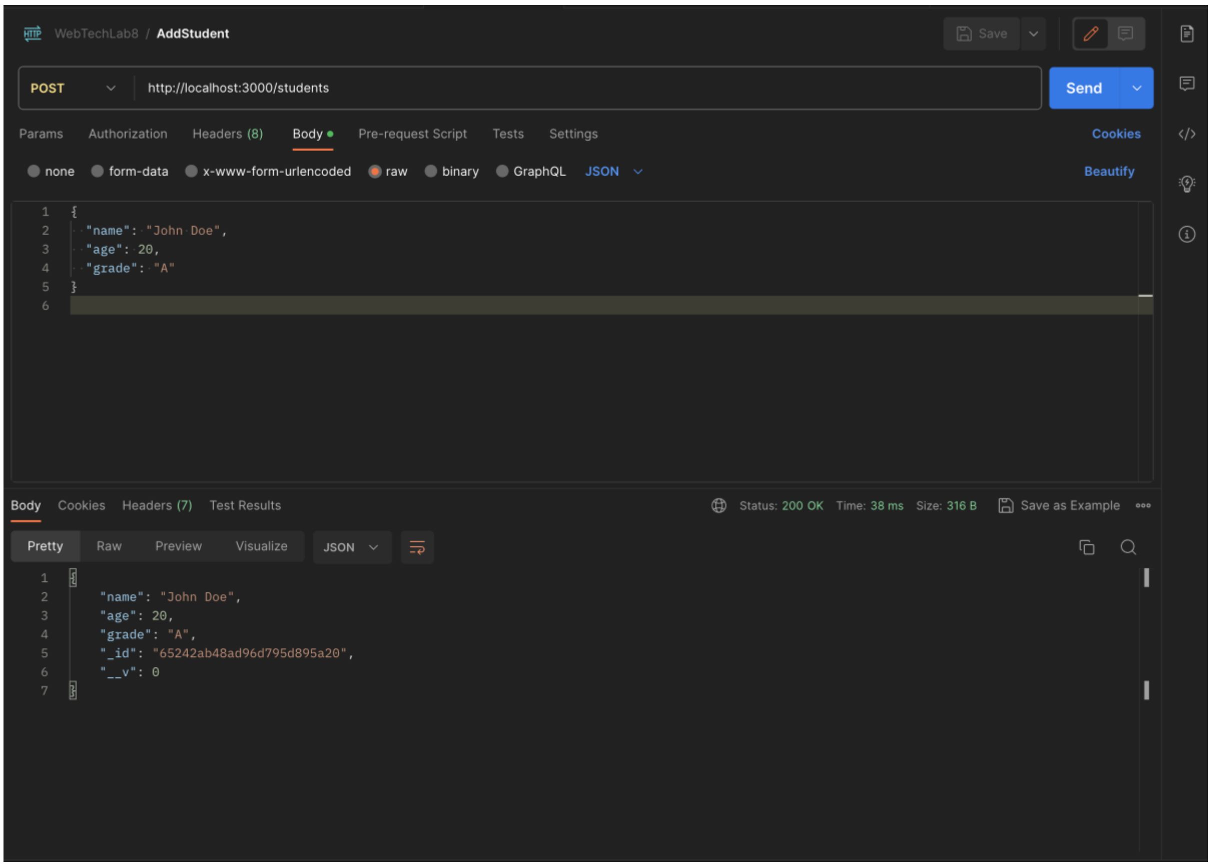
To Build a REST API with Express, Node, and MongoDB

Program:

Server.js

```
const express = require('express')
const app = express()
const mongoose = require('mongoose')

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true })
const db = mongoose.connection
db.on('error', (error) => console.error(error))
db.once('open', () => console.log('Connected to Database'))

app.use(express.json())

const studentSchema = new mongoose.Schema({
  name: String,
  age: Number,
  grade: String,
});

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

app.get('/students', async (req, res) => {
  try {
    const students = await Student.find();
    res.json(students);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.post('/students', async (req, res) => {
    const { name, age, grade } = req.body;

    try {
        const newStudent = new Student({ name, age, grade });
        await newStudent.save();
        res.json(newStudent);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.put('/students/:id', async (req, res) => {
    const { id } = req.params;
    const { name, age, grade } = req.body;

    try {
        const updatedStudent = await Student.findByIdAndUpdate(
            id,
            { name, age, grade },
            { new: true } // Return the updated document
        );

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

        res.json(updatedStudent);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.delete('/students/:id', async (req, res) => {
    const { id } = req.params;

    try {
        const deletedStudent = await Student.findByIdAndDelete(id);

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

    res.json({ message: 'Student deleted successfully' });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.listen(3000, () => console.log('Server Started'))
```

Output:



WebTechLab8 / AddStudent

POST http://localhost:3000/students

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

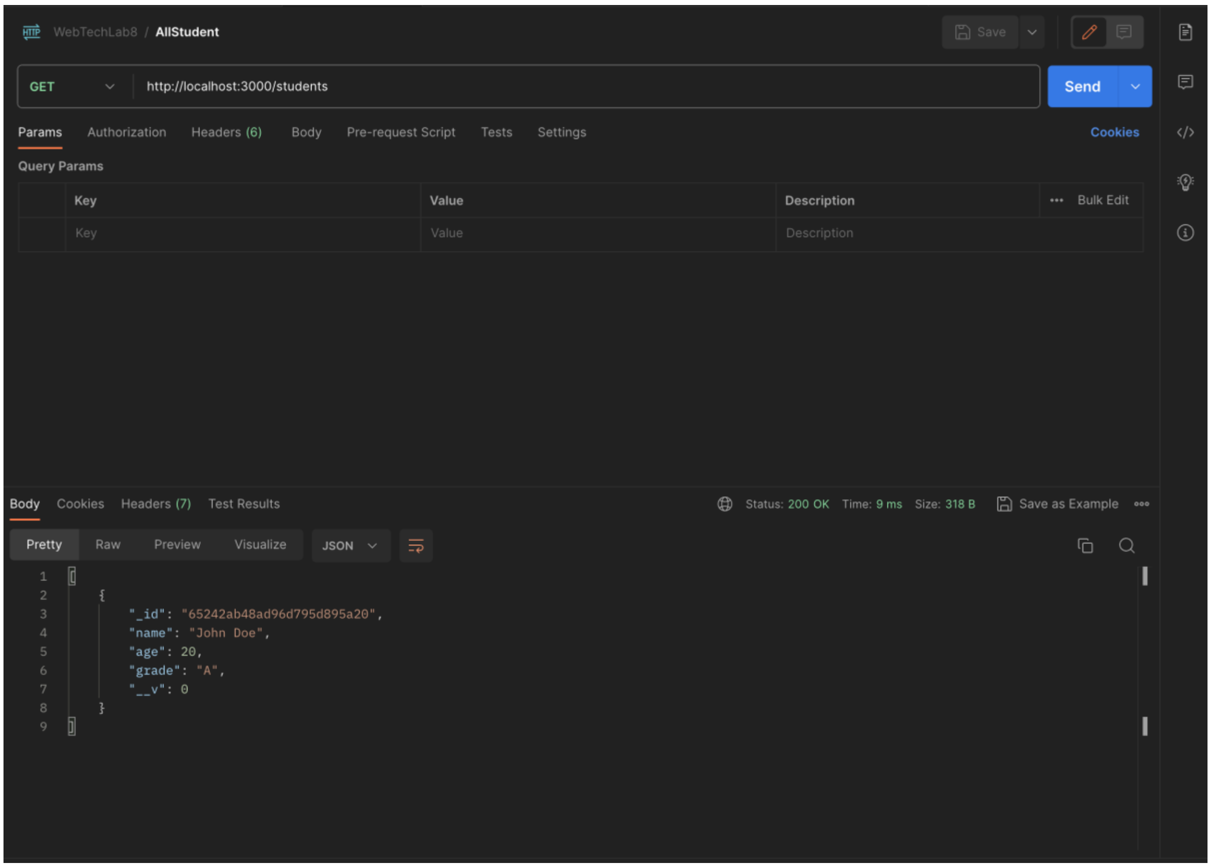
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON | Beautify

```
{
  "name": "John Doe",
  "age": 20,
  "grade": "A"
}
```

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 38 ms Size: 316 B — Save as Example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "name": "John Doe",
    "age": 20,
    "grade": "A",
    "_id": "65242ab48ad96d795d895a20",
    "__v": 0
}
```

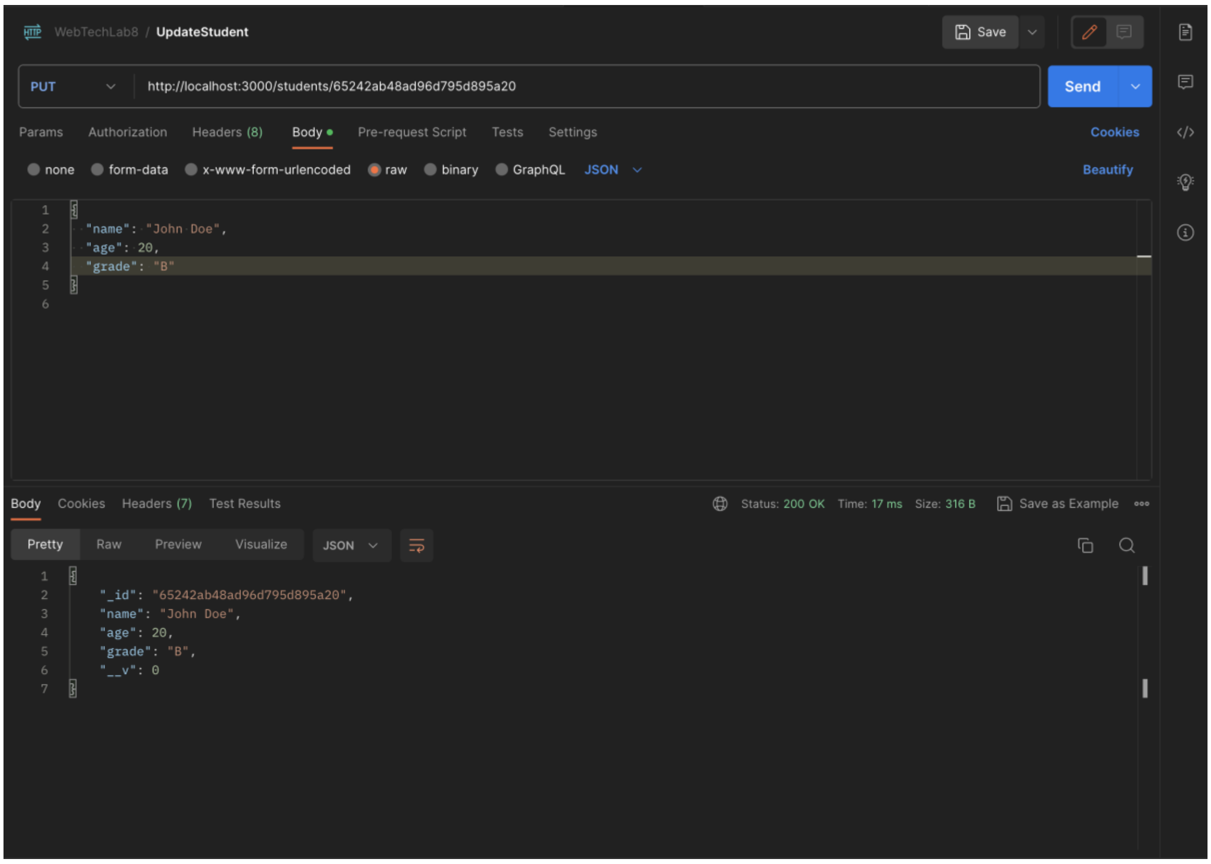

WebTechLab8 / AllStudent

GET http://localhost:3000/students

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results — Status: 200 OK Time: 9 ms Size: 318 B

```
[
    {
        "_id": "65242ab48ad96d795d895a20",
        "name": "John Doe",
        "age": 20,
        "grade": "A",
        "__v": 0
    }
]
```

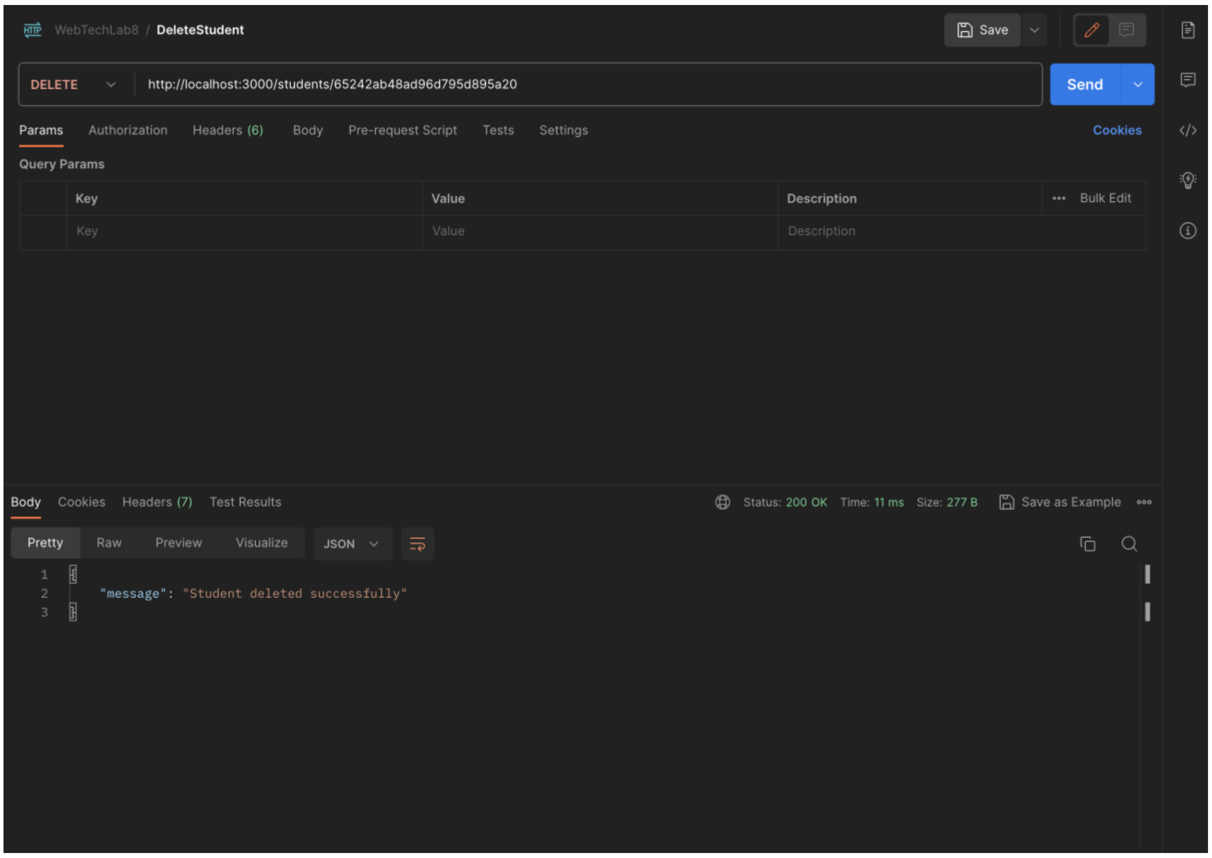

WebTechLab8 / UpdateStudent

PUT http://localhost:3000/students/65242ab48ad96d795d895a20

```
{
  "name": "John Doe",
  "age": 20,
  "grade": "B"
}
```

Body Cookies Headers (7) Test Results — Status: 200 OK Time: 17 ms Size: 316 B

```
{
    "_id": "65242ab48ad96d795d895a20",
    "name": "John Doe",
    "age": 20,
    "grade": "B",
    "__v": 0
}
```



http://localhost:3000/students/65242ab48ad96d795d895a20

Status: 200 OK Time: 11 ms Size: 277 B

```
{
    "message": "Student deleted successfully"
}
```

Result:

Successfully built a REST API with Express, Node, and MongoDB