

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);

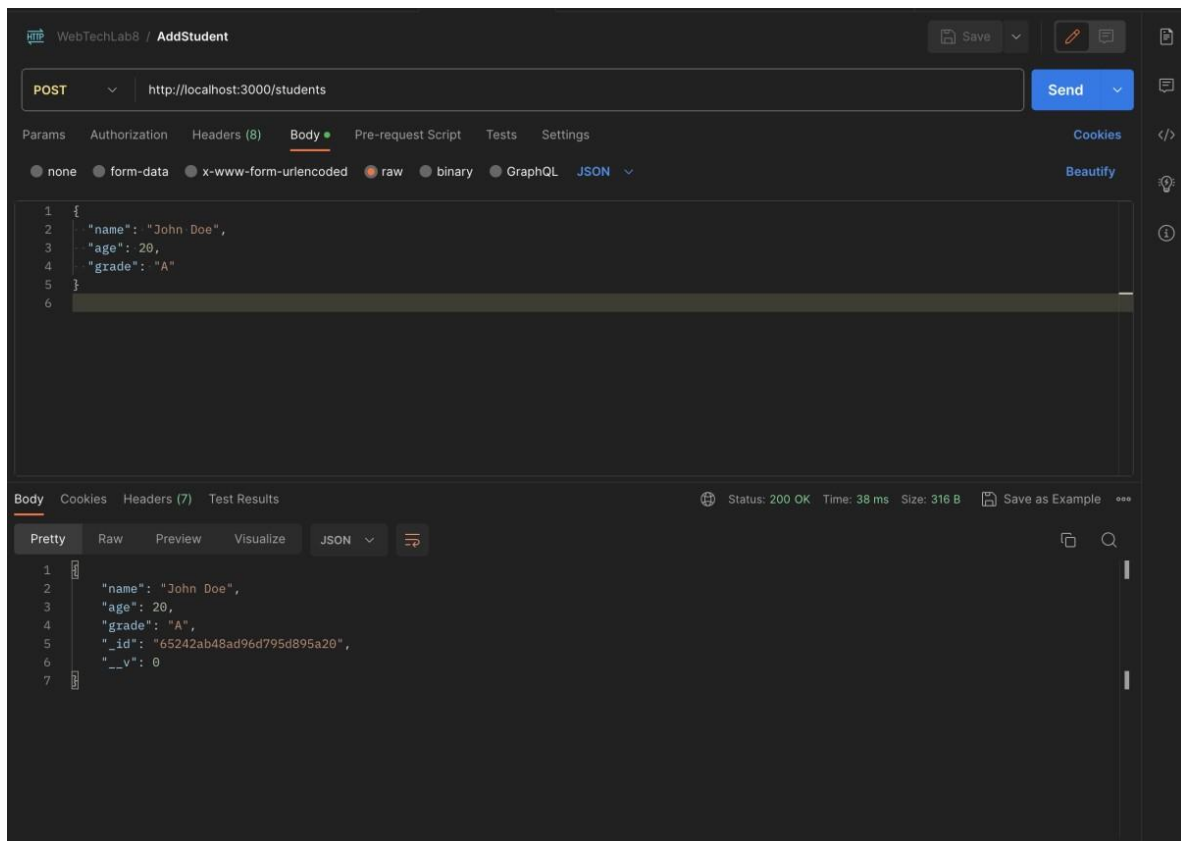
    if (!deletedStudent) {
      return res.status(404).json({ error: 'Student not found' });
    }
  }
});
```

```
}

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

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

Output:



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/students
- Body Type:** raw
- Request Body:**

```
1 {
2   "name": "John Doe",
3   "age": 20,
4   "grade": "A"
5 }
```
- Status:** 200 OK
- Time:** 38 ms
- Size:** 316 B
- Response Body (JSON):**

```
1 {
2   "name": "John Doe",
3   "age": 20,
4   "grade": "A",
5   "_id": "65242ab48ad96d795d895a20",
6   "__v": 0
7 }
```

WebTechLab8 / AllStudent

GET

http://localhost:3000/students

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

Status: 200 OK Time: 9 ms Size: 318 B Save as Example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": "65242ab48ad96d795d895a20",
3   "name": "John Doe",
4   "age": 20,
5   "grade": "A",
6   "__v": 0
7 }
```

WebTechLab8 / UpdateStudent

PUT

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

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "name": "John Doe",
3   "age": 20,
4   "grade": "B"
5 }
```

Body

Cookies

Headers (7)

Test Results

Status: 200 OK Time: 17 ms Size: 316 B Save as Example

Pretty

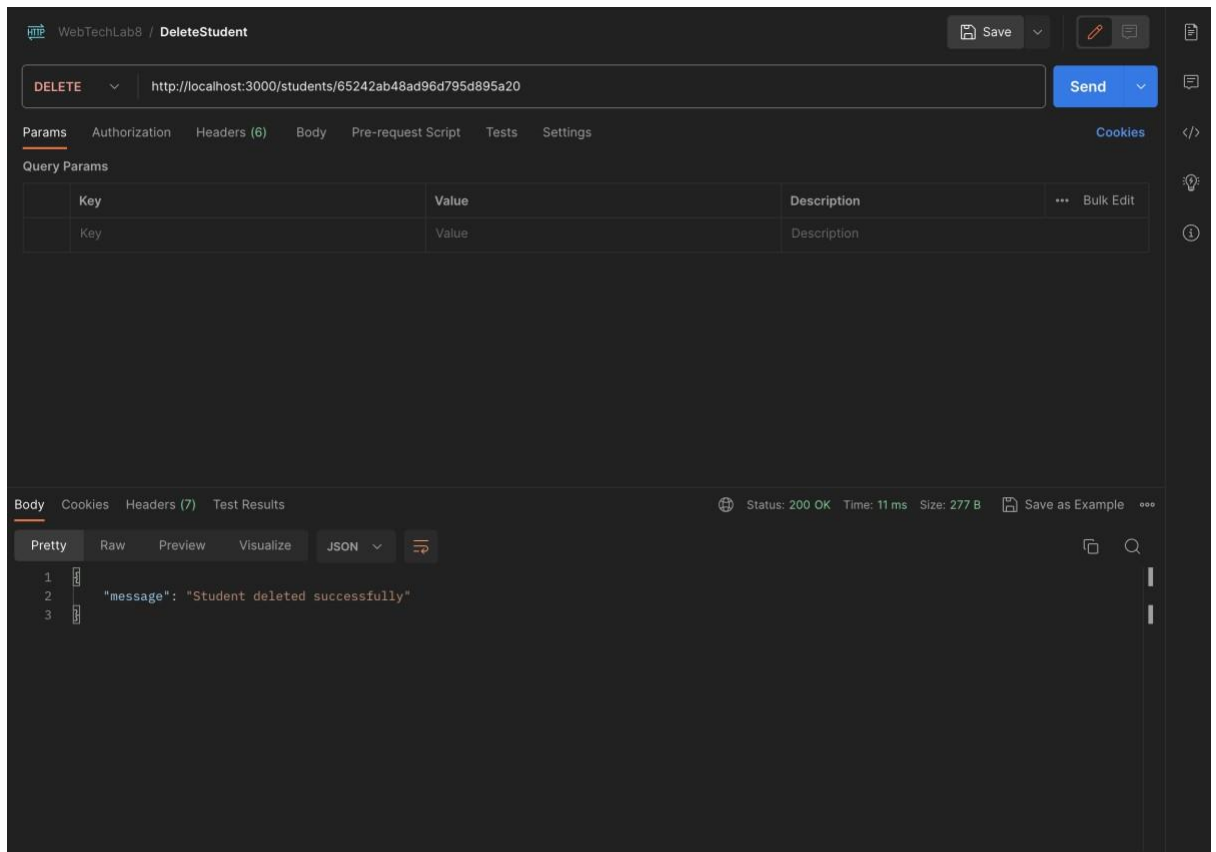
Raw

Preview

Visualize

JSON

```
1 {
2   "id": "65242ab48ad96d795d895a20",
3   "name": "John Doe",
4   "age": 20,
5   "grade": "B",
6   "__v": 0
7 }
```



Git hub link: <https://github.com/vatsav123/webtech-all-labs.git>

Result:

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

