Ex. No: 8	Building a REST API with Express, Node, and MongoDB
03.10.2023	

Aim:

To Build a REST API using EJS ,NODE,MongoDB

Algorithm:

1. Import express and mongoose in index.js
2. use json as the format and establish connection using mongoose and atlas
3. once connection is established implement the GET ,PUT POST methods
4. test out different API Methods to ensure proper working

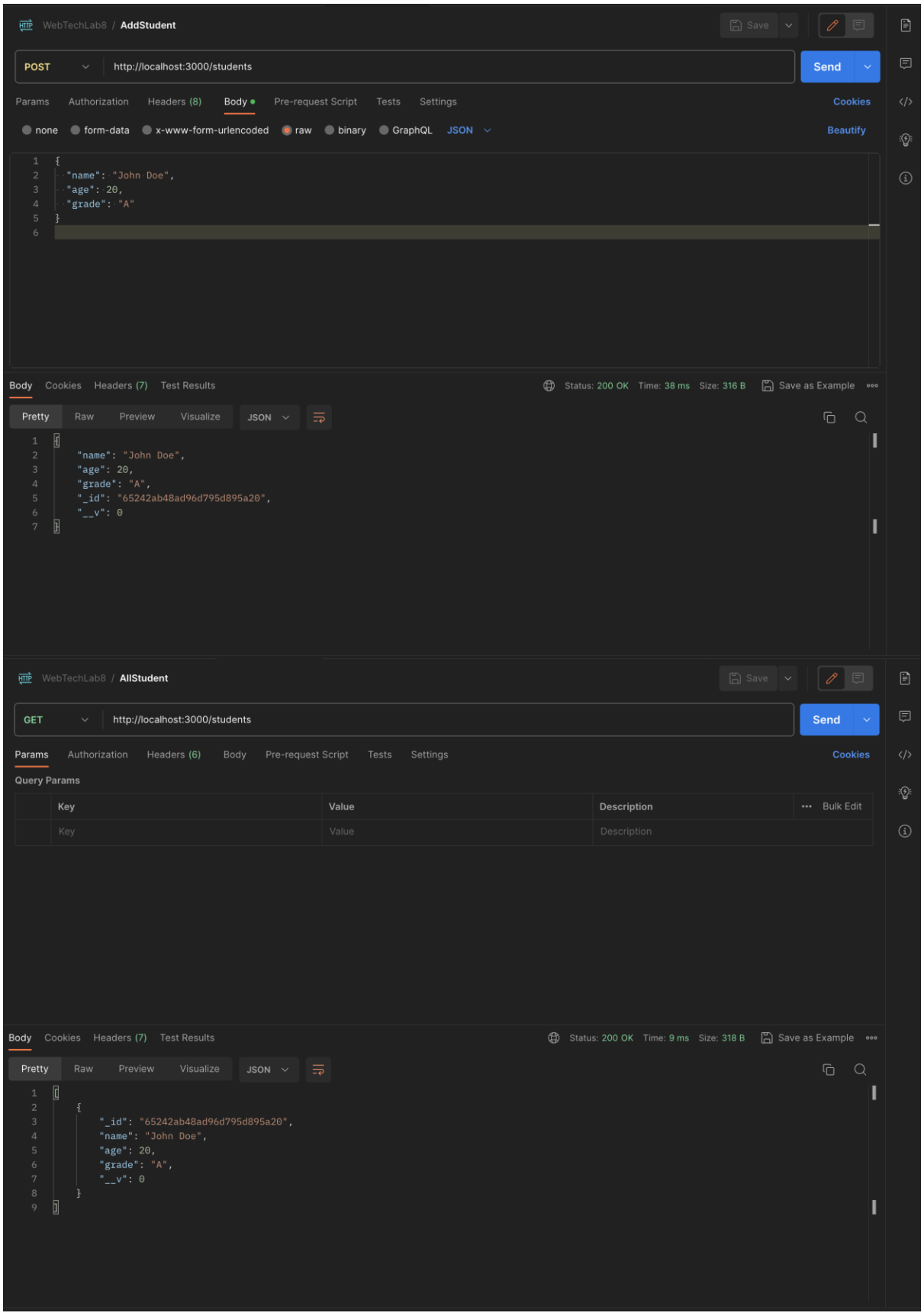
Program:

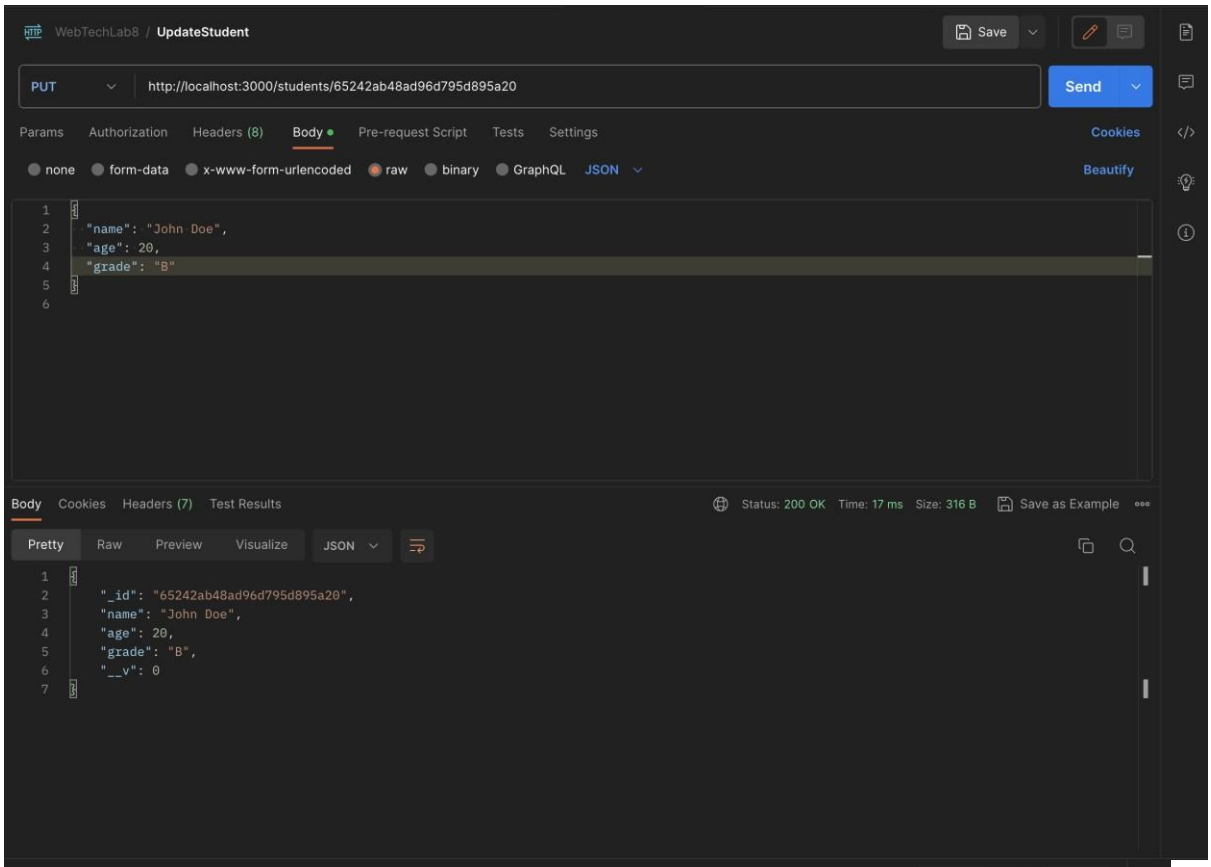
Server.js

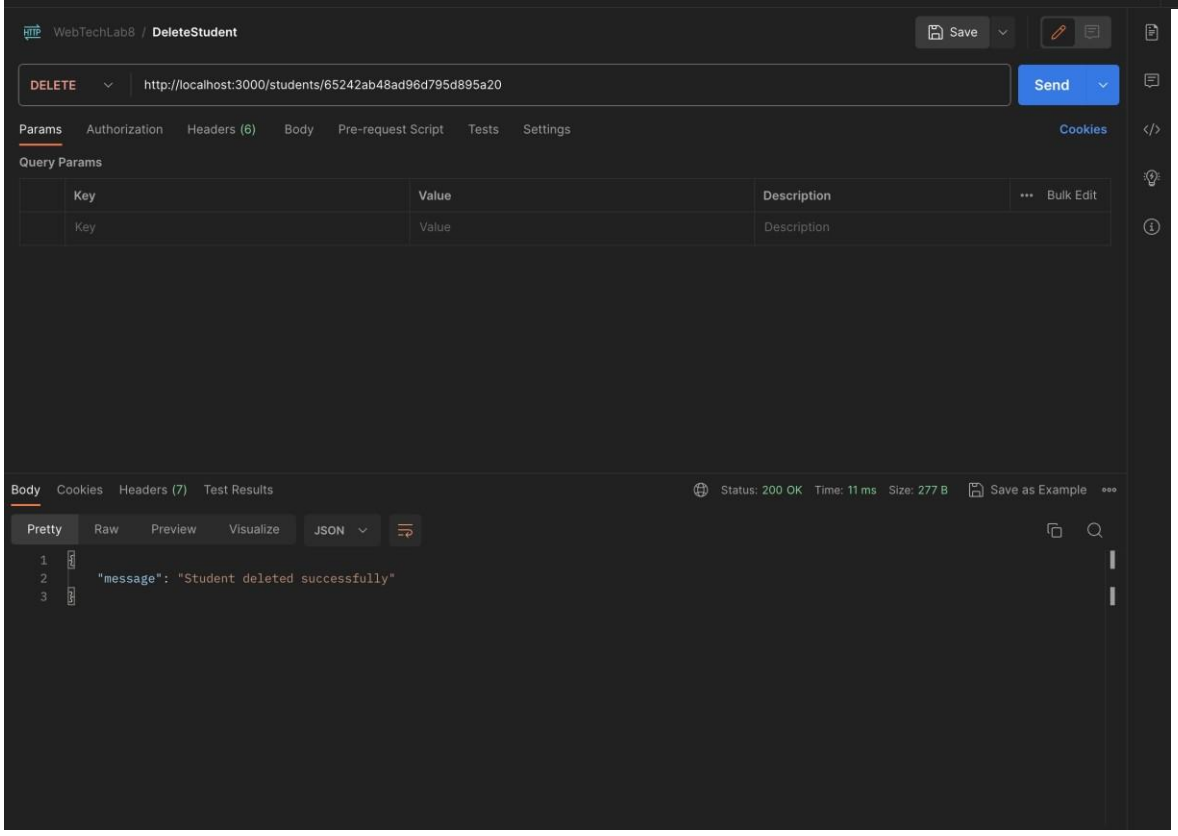
```
const express = require('express')
const app = express()
const mongoose = require('mongoose')

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
Save
POST
http://localhost:3000/students
Send
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
Save
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
[
{
"_id": "65242ab48ad96d795d895a20",
"name": "John Doe",
"age": 20,
"grade": "A",
"__v": 0
}
]





Result: Thus REST API Implemented using EJS,Mongo.