

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

POST

http://localhost:3000/students

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

Beautiful

```
1 {
2   "name": "John Doe",
3   "age": 20,
4   "grade": "A"
5 }
6
```

BodyCookiesHeaders (7)Test Results

Status: 200 OKTime: 38 msSize: 316 BSave as Example

PrettyRawPreviewVisualizeJSON

```
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

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (7)Test Results

Status: 200 OKTime: 9 msSize: 318 BSave as Example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "_id": "65242ab48ad96d795d895a20",
3   "name": "John Doe",
4   "age": 20,
5   "grade": "A",
6   "__v": 0
7 }
8
9
```

The image displays two screenshots of a REST client interface, likely Postman, used for testing a REST API. The top screenshot shows a PUT request to the endpoint `http://localhost:3000/students/65242ab48ad96d795d895a20` with a JSON body containing student information. The bottom screenshot shows a DELETE request to the same endpoint, resulting in a 200 OK status and a success message in the response body.

Top Screenshot: PUT Request

Method: PUT
URL: `http://localhost:3000/students/65242ab48ad96d795d895a20`
Body (JSON):

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

Bottom Screenshot: DELETE Request

Method: DELETE
URL: `http://localhost:3000/students/65242ab48ad96d795d895a20`
Status: 200 OK
Time: 11 ms
Size: 277 B
Body (JSON):

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

Result: Thus REST API Implemented using EJS,Mongo.