

Assignment:3: A RESTful API using express.js and create a database and index in MongoDB

Source code:

```
const express = require('express');
const mongoose =
require('mongoose'); const app =
express(); const port = 3001;

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/myDatabase', { useNewUrlParser:
true, useUnifiedTopology: true }); const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection
error:')); db.once('open', function() { console.log('Connected to
MongoDB...');
});

// Define a schema for your data
const Schema = mongoose.Schema;
const exampleSchema = new
Schema({ name: String,
age: Number, salary: Number,
role: String
});

// Create a model based on the schema const Example
= mongoose.model('Example', exampleSchema); //
Express middleware for parsing JSON bodies
app.use(express.json());

// Route to create a new example
app.post('/examples', async (req, res) => {
try {
const example = await
Example.create(req.body);
res.status(201).json(example); } catch (err) {
res.status(400).json({ message: err.message }); }
});

// Route to retrieve all examples where salary and role have values
greater than 2 app.get('/examples', async (req, res) => {
```

```
    try {        const examples = await
Example.find({
    $and: [
        { salary: { $gt: 2 } }, // Salary greater than 2
        { role: { $gt: 2 } }    // Role greater than 2
    ]
    });
res.json(examples);
    } catch (err) {        res.status(500).json({ message:
err.message });
    }
});

// Route to retrieve a specific example by ID app.get('/examples/:id',
async (req, res) => {    try {        const example = await
Example.findById(req.params.id);        if (!example) {
return res.status(404).json({ message: 'Example not found' });
        }        res.json(example);    } catch (err)
{
        res.status(500).json({ message: err.message
});
    }
});

// Route to update an example by ID app.put('/examples/:id', async
(req, res) => {    try {        const example = await
Example.findByIdAndUpdate(req.params.id, req.body, { new: true });
if (!example) {        return res.status(404).json({ message:
'Example not found' });
    }
    res.json(example);    } catch (err) {
res.status(400).json({ message: err.message });
    }
});

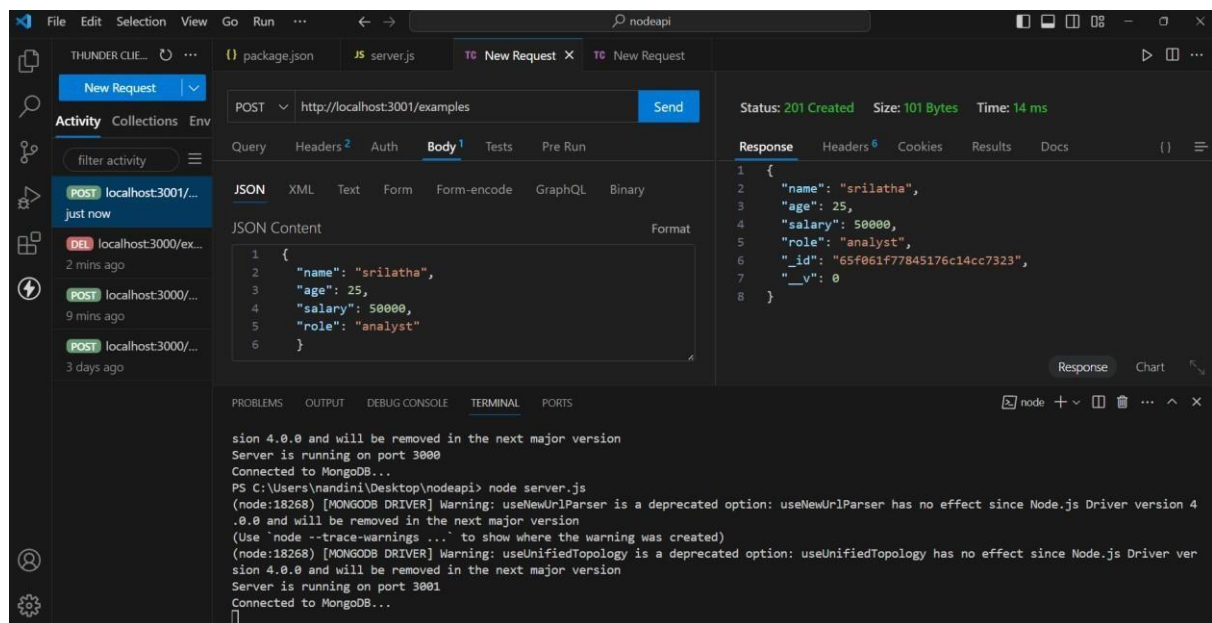
// Route to delete an example by ID
app.delete('/examples/:id', async (req, res) => {    try {        const
example = await Example.findByIdAndDelete(req.params.id);        if
(!example) {        return res.status(404).json({ message: 'Example
not found' });
    }
    res.json({ message: 'Example deleted
successfully' });
});
```

```
    } catch (err) {      res.status(500).json({ message:
err.message });
    }
  });

// Start the server app.listen(port, () => {
console.log(`Server is running on port ${port}`); });
```

OutPut:

Post method:



The screenshot displays the Thunder Client interface. On the left, the 'Activity' panel shows a list of requests, with the most recent one being a POST request to 'localhost:3001/examples'. The main panel shows the details of this request. The 'Query' tab is selected, showing the URL 'http://localhost:3001/examples'. The 'Body' tab is also visible, showing the JSON content:

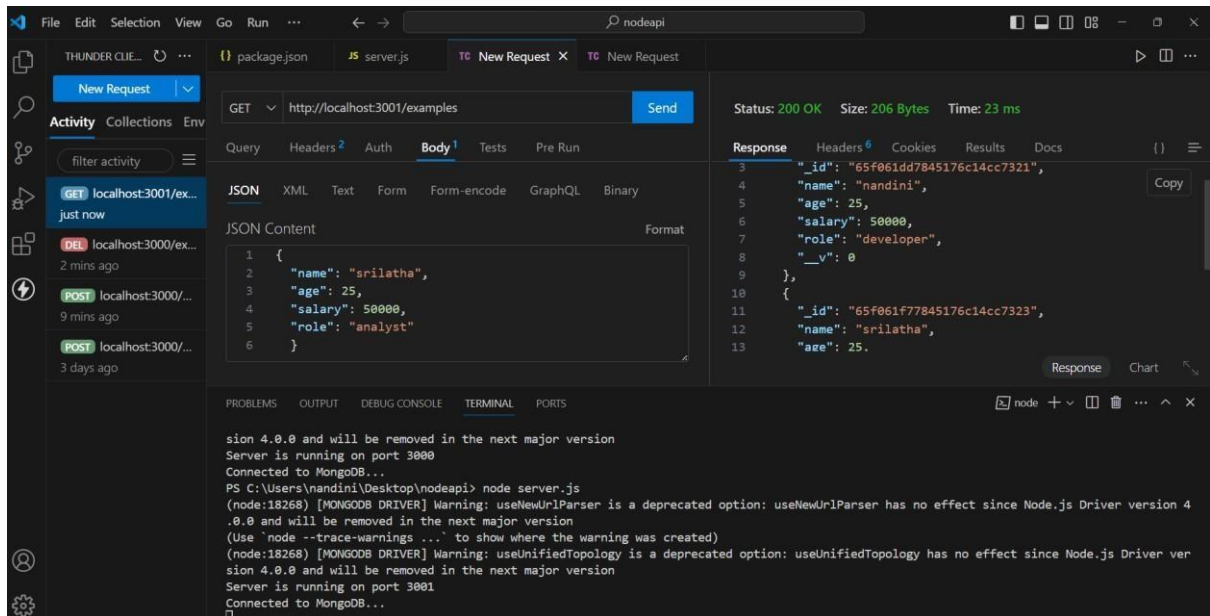
```
{
  "name": "srilatha",
  "age": 25,
  "salary": 50000,
  "role": "analyst"
}
```

. The 'Response' tab shows the status '201 Created', size '101 Bytes', and time '14 ms'. The response body is a JSON object:

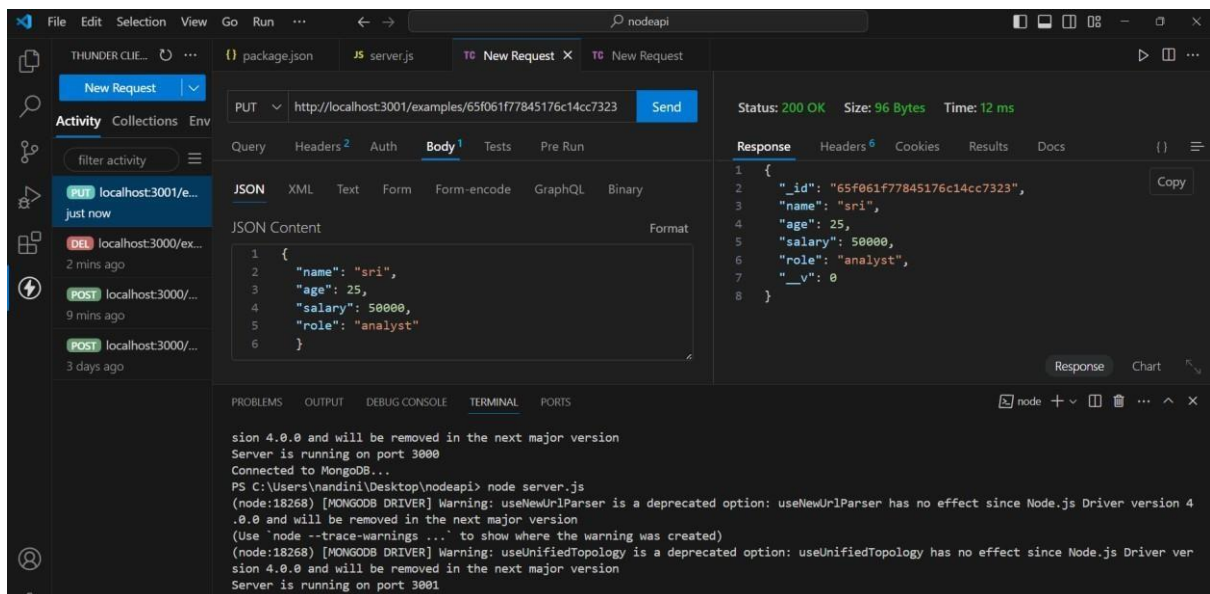
```
{
  "name": "srilatha",
  "age": 25,
  "salary": 50000,
  "role": "analyst",
  "_id": "65f061f77845176c14cc7323",
  "__v": 0
}
```

. The bottom panel shows the terminal output, which includes the command 'node server.js' and the message 'Server is running on port 3000'.

Get Method:



Put method:



Delete method:

The screenshot displays the Thunder Client interface with a REST client request and response. The request is a DELETE method to the URL `http://localhost:3001/examples/65f061f77845176c14cc73`. The response is a 200 OK status with a JSON body containing a success message.

Request Details:

- Method: DELETE
- URL: `http://localhost:3001/examples/65f061f77845176c14cc73`
- Body:

```
{  "name": "sri",  "age": 25,  "salary": 50000,  "role": "analyst"}
```

Response Details:

- Status: 200 OK
- Size: 42 Bytes
- Time: 19 ms
- Body:

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

Terminal Output:

```
sion 4.0.0 and will be removed in the next major version
Server is running on port 3000
Connected to MongoDB...
PS C:\Users\nandini\Desktop\nodeapi> node server.js
(node:18268) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4
.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:18268) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver ver
sion 4.0.0 and will be removed in the next major version
Server is running on port 3001
Connected to MongoDB...
```