

**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 });
    }
});

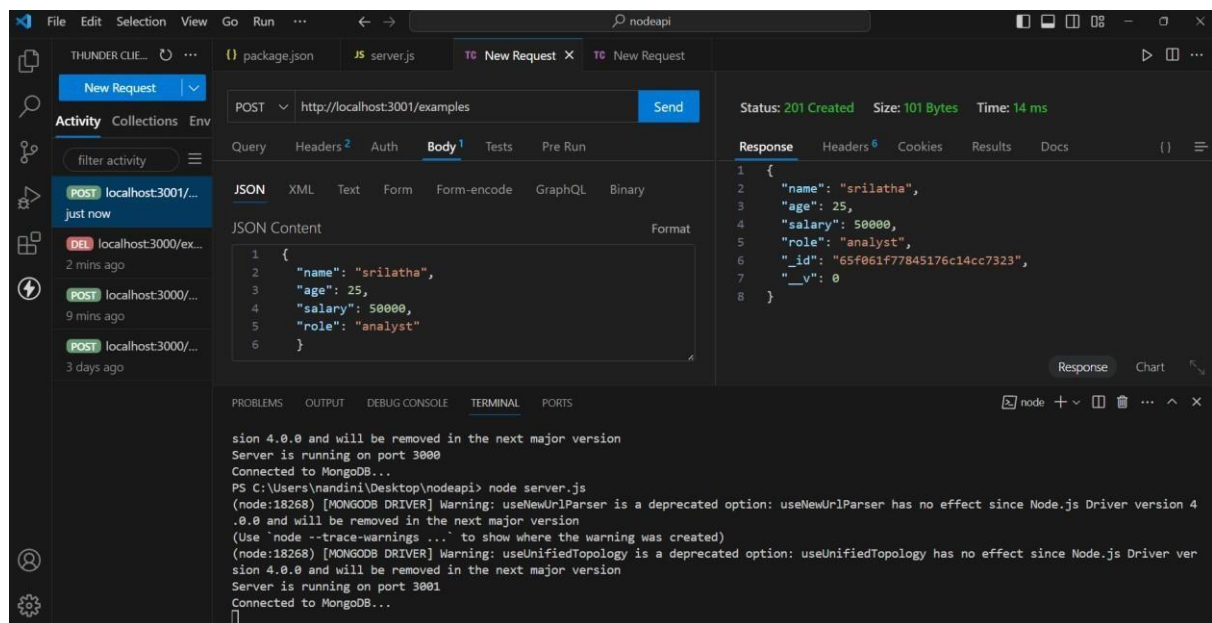
// Route to delete an example by ID
app.delete('/examples/:id', async (req, res) => {    try {        const
example = await Example.findByIdAndDelete(req.params.id);        if
(!example) {        return res.status(404).json({ message: 'Example
not found' });
        }
        res.json({ message: 'Example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



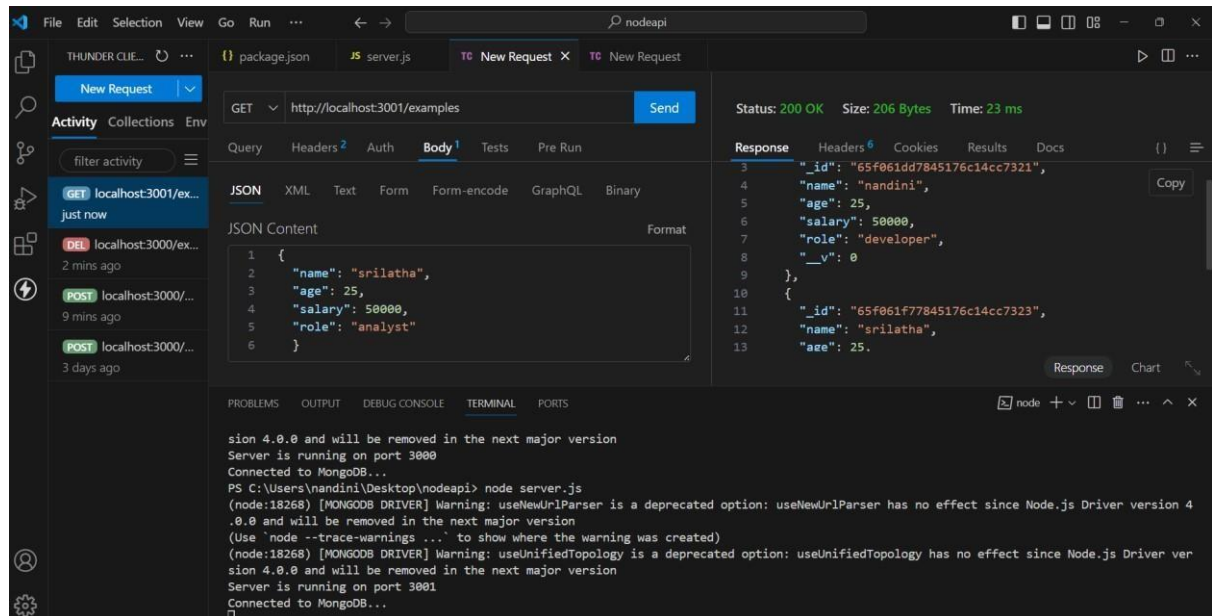
The screenshot displays the VS Code interface with a REST client request and response. The request is a POST to `http://localhost:3001/examples`. The response is a JSON object with the following structure:

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

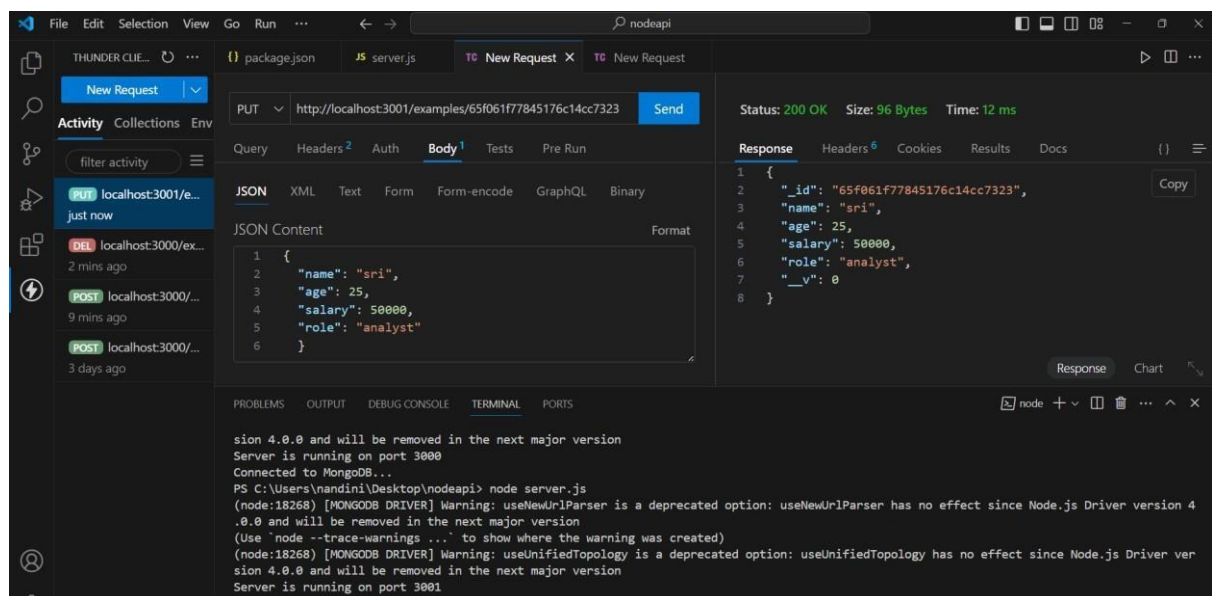
The terminal output shows the server running on port 3000 and the client sending the request. The response is a JSON object with the following structure:

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

### Get Method:



### Put method:



### Delete method:

Name:N.Tejaswi

Rollno:2022MCA16062

The screenshot displays the Thunder Client interface. The top bar includes standard menu items (File, Edit, Selection, View, Go, Run) and a search bar. Below the menu, the 'package.json' and 'server.js' files are visible. The main workspace is divided into several panes:

- Activity:** Shows a list of recent requests with status icons (DELETE, POST) and timestamps.
- Request Editor:** The 'Body' tab is selected, showing a JSON request body: 

```
{  "name": "sri",  "age": 25,  "salary": 50000,  "role": "analyst"}
```
- Response:** Shows the status '200 OK', size '42 Bytes', and time '19 ms'. The response body is: 

```
{  "message": "Example deleted successfully"}
```
- Terminal:** Displays the command prompt output, showing the server running on port 3000 and connected to MongoDB. It also includes deprecation warnings for 'useNewUrlParser' and 'useUnifiedTopology'.