

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

Name: PAMANJI VISALAKSHI

Roll number: 2022MCA16069

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 shows the Thunder Client interface with a POST request to `http://localhost:3001/examples`. The request body is a JSON object:

```

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

```

The response is a JSON object:

```

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

```

The status is 201 Created, size is 101 Bytes, and time is 14 ms. The terminal shows the server running on port 3000 and the client running on port 3001.

Get Method:

The screenshot shows the Thunder Client interface with a GET request to `http://localhost:3001/examples`. The response is a JSON array with two objects. The terminal shows the server running on port 3000.

Request:

```
GET http://localhost:3001/examples
```

Response:

```
[{"_id": "65f061dd7845176c14cc7321", "name": "nandini", "age": 25, "salary": 50000, "role": "developer", "__v": 0}, {"_id": "65f061f77845176c14cc7323", "name": "srilatha", "age": 25, "salary": 50000, "role": "analyst", "__v": 0}]
```

Terminal:

```
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
(node:18268) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server is running on port 3000
Connected to MongoDB...
```

Put method:

The screenshot shows the Thunder Client interface with a PUT request to `http://localhost:3001/examples/65f061f77845176c14cc7323`. The response is a JSON object. The terminal shows the server running on port 3000.

Request:

```
PUT http://localhost:3001/examples/65f061f77845176c14cc7323
```

Response:

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

Terminal:

```
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
(node:18268) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server is running on port 3000
Connected to MongoDB...
```

Delete method:

The screenshot displays the Thunder Client interface with a DELETE request to `http://localhost:3001/examples/65f061f77845176c14cc73`. The request body is a JSON object: `{ "name": "sri", "age": 25, "salary": 50000, "role": "analyst" }`. The response is a 200 OK status with a message: `{ "message": "Example deleted successfully" }`. The terminal at the bottom shows the server running on port 3000 and connected to MongoDB.

Activity: `DEL localhost:3001/ex...` just now

Activity: `DEL localhost:3000/ex...` 2 mins ago

Activity: `POST localhost:3000/...` 9 mins ago

Activity: `POST localhost:3000/...` 3 days ago

Request: `DELETE http://localhost:3001/examples/65f061f77845176c14cc73`

Response: `200 OK` Size: 42 Bytes Time: 19 ms

Response 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...
```