**Name:C.PAVANI**

**Roll No:2022MCA16017**

**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 });

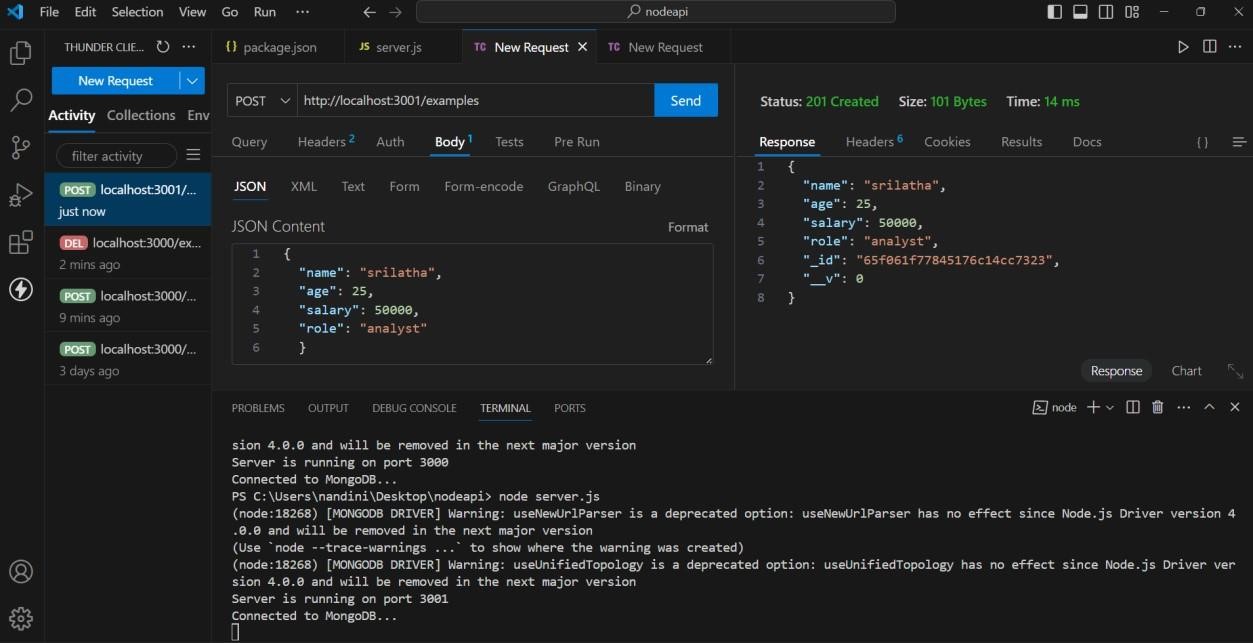
}

});

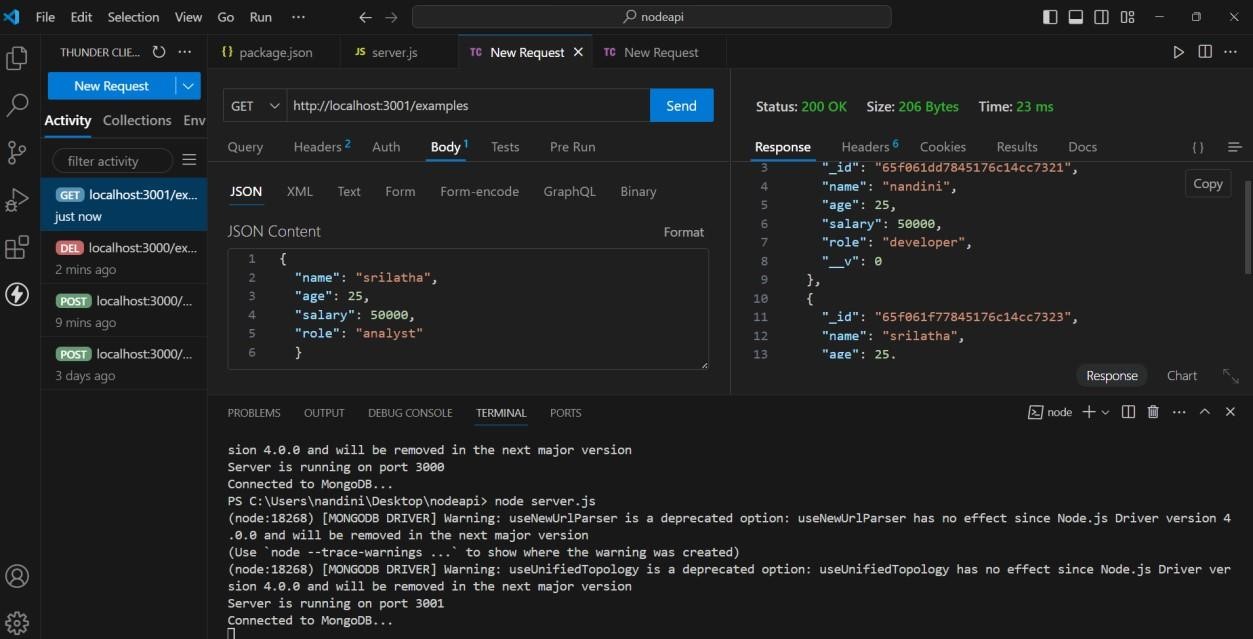
// Start the server app.listen(port, () => { console.log(`Server is running on port ${port}`); });

**OutPut:**

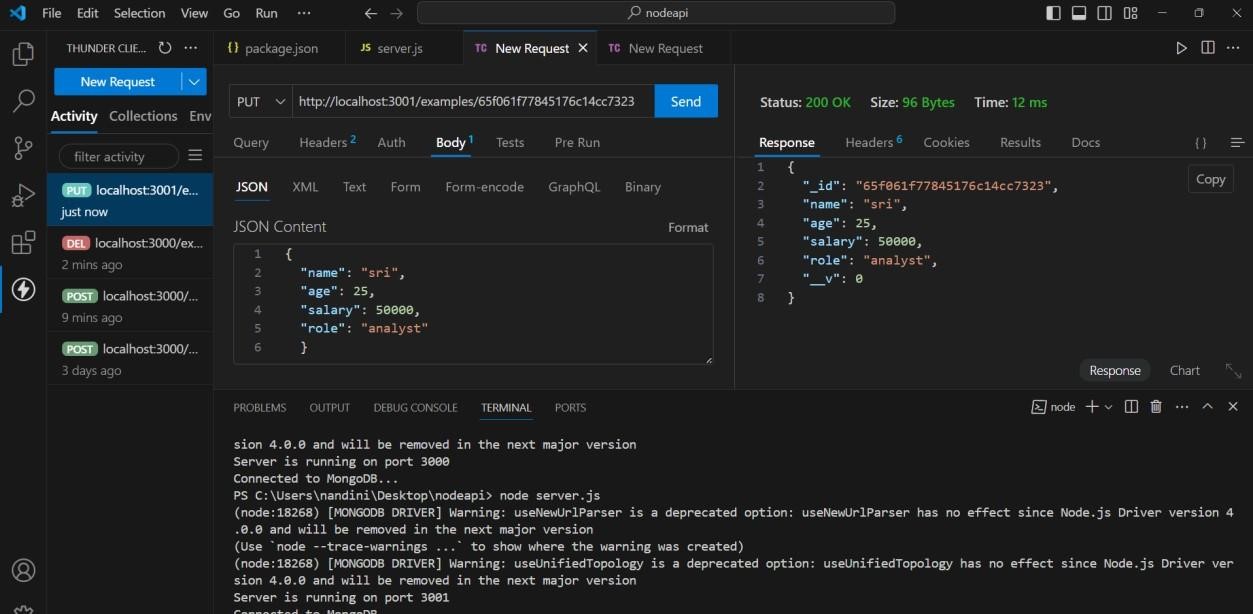
Post method:



Get Method:



Put method:



Delete method:

