

Subject code & Name	:	CSE304 & Full Stack Development	Practical	:	18	Academic Year	:	2024-25
---------------------	---	---------------------------------	-----------	---	----	---------------	---	---------

## Task – 18

### Aim

Create Rest API CRUD using MongoDB and Express JS.

### Code

**app.js**

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const tasksRoute = require('./routes/tasks');

const app = express();
const port = process.env.PORT || 3000;

app.use(cors());
app.use(bodyParser.json());

mongoose.connect(process.env.MONGODB_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('Could not connect to MongoDB:', err));

app.use('/api/tasks', tasksRoute);

app.get('/', (req, res) => {
  res.send('Welcome to the Task Manager API');
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

**.env**

MONGODB\_URI=mongodb://localhost:27017/taskmanager

**Models/Task.js**

```
const mongoose = require('mongoose');

const TaskSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  completed: {
    type: Boolean,
    default: false,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

module.exports = mongoose.model('Task', TaskSchema);
```

**routes/tasks.js**

```
const express = require('express');
const router = express.Router();
const Task = require('../models/Task');

router.post('/', async (req, res) => {
  const { title, description } = req.body;
  try {
    const newTask = new Task({ title, description });
    const savedTask = await newTask.save();
    res.status(201).json(savedTask);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

router.get('/', async (req, res) => {
  try {
    const tasks = await Task.find();
    res.status(200).json(tasks);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```

```
    }  
  });  
  
  router.get('/:id', async (req, res) => {  
    try {  
      const task = await Task.findById(req.params.id);  
      if (!task) return res.status(404).json({ message: 'Task not found' });  
      res.status(200).json(task);  
    } catch (err) {  
      res.status(500).json({ message: err.message });  
    }  
  });  
  
  router.put('/:id', async (req, res) => {  
    const { title, description, completed } = req.body;  
    try {  
      const updatedTask = await Task.findByIdAndUpdate(  
        req.params.id,  
        { title, description, completed },  
        { new: true }  
      );  
      if (!updatedTask) return res.status(404).json({ message: 'Task not found' });  
      res.status(200).json(updatedTask);  
    } catch (err) {  
      res.status(400).json({ message: err.message });  
    }  
  });  
  
  router.delete('/:id', async (req, res) => {  
    try {  
      const deletedTask = await Task.findByIdAndDelete(req.params.id);  
      if (!deletedTask) return res.status(404).json({ message: 'Task not found' });  
      res.status(200).json({ message: 'Task deleted successfully' });  
    } catch (err) {  
      res.status(500).json({ message: err.message });  
    }  
  });  
  
  module.exports = router;
```

## Output

Create a new task (POST):

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://localhost:3000/api/tasks`
- Status:** 201 Created
- Size:** 167 Bytes
- Time:** 141 ms
- Body:** JSON content: 

```
{ 1: { 2: "title": "My first task", 3: "description": "This is a task description.", 4: "completed": false, 5: "_id": "6703fc66bb344498179e9361", 6: "createdAt": "2024-10-07T15:21:10.641Z", 7: "_v": 0 8: } }
```
- Response:** JSON content: 

```
1 { 2 "title": "My first task", 3 "description": "This is a task description.", 4 "completed": false, 5 "_id": "6703fc66bb344498179e9361", 6 "createdAt": "2024-10-07T15:21:10.641Z", 7 "_v": 0 8 }
```

Get all tasks (GET):

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:3000/api/tasks`
- Status:** 200 OK
- Size:** 505 Bytes
- Time:** 14 ms
- Body:** XML content (empty)
- Response:** JSON content (array of 3 tasks): 

```
1 [ 2 { 3 "_id": "6703fc66bb344498179e9361", 4 "title": "My first task", 5 "description": "This is a task description.", 6 "completed": false, 7 "createdAt": "2024-10-07T15:21:10.641Z", 8 "_v": 0 9 }, 10 { 11 "_id": "6703fc66bb344498179e9361", 12 "title": "My first task", 13 "description": "This is a task description.", 14 "completed": false, 15 "createdAt": "2024-10-07T15:22:04.423Z", 16 "_v": 0 17 }, 18 { 19 "_id": "6703fc66bb344498179e9361", 20 "title": "My first task", 21 "description": "This is a task description.", 22 "completed": false, 23 "createdAt": "2024-10-07T15:22:05.607Z", 24 "_v": 0 25 } 26 ]
```

Get a single task by ID (GET):

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:3000/api/tasks/6703fc66bb344498179e9361`
- Status:** 200 OK
- Size:** 505 Bytes
- Time:** 4 ms
- Body:** JSON content (empty)
- Response:** JSON content (single task): 

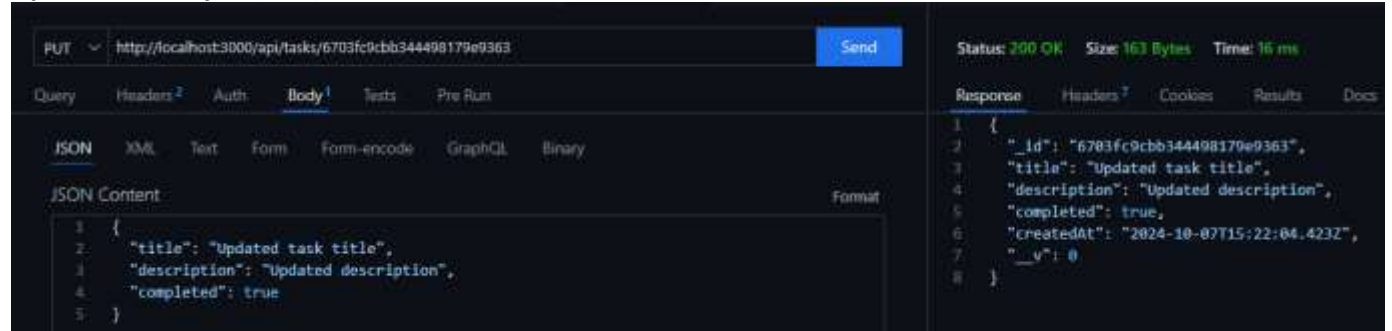
```
1 { 2 "_id": "6703fc66bb344498179e9361", 3 "title": "My first task", 4 "description": "This is a task description.", 5 "completed": false, 6 "createdAt": "2024-10-07T15:21:10.641Z", 7 "_v": 0 8 }
```

Delete a task by ID (DELETE):

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `http://localhost:3000/api/tasks/6703fc66bb344498179e9361`
- Status:** 200 OK
- Size:** 39 Bytes
- Time:** 14 ms
- Body:** JSON content (empty)
- Response:** JSON content (success message): 

```
1 { 2 "message": "Task deleted successfully" 3 }
```

**Update a task by ID (PUT):**

The screenshot displays a REST client interface with a PUT request to `http://localhost:3000/api/tasks/6703fc9cbb344498179e9363`. The request body is a JSON object: `{ "title": "Updated task title", "description": "Updated description", "completed": true }`. The response status is `200 OK` with a size of `163 Bytes` and a time of `16 ms`. The response body is a JSON object: `{ "_id": "6703fc9cbb344498179e9363", "title": "Updated task title", "description": "Updated description", "completed": true, "createdAt": "2024-10-07T15:22:04.423Z", "__v": 0 }`.

Method	URL	Status	Size	Time
PUT	http://localhost:3000/api/tasks/6703fc9cbb344498179e9363	200 OK	163 Bytes	16 ms

Query Headers Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```
1 {
2   "title": "Updated task title",
3   "description": "Updated description",
4   "completed": true
5 }
```

Format

Response Headers Cookies Results Docs

```
1 {
2   "_id": "6703fc9cbb344498179e9363",
3   "title": "Updated task title",
4   "description": "Updated description",
5   "completed": true,
6   "createdAt": "2024-10-07T15:22:04.423Z",
7   "__v": 0
8 }
```

**Signature :**