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

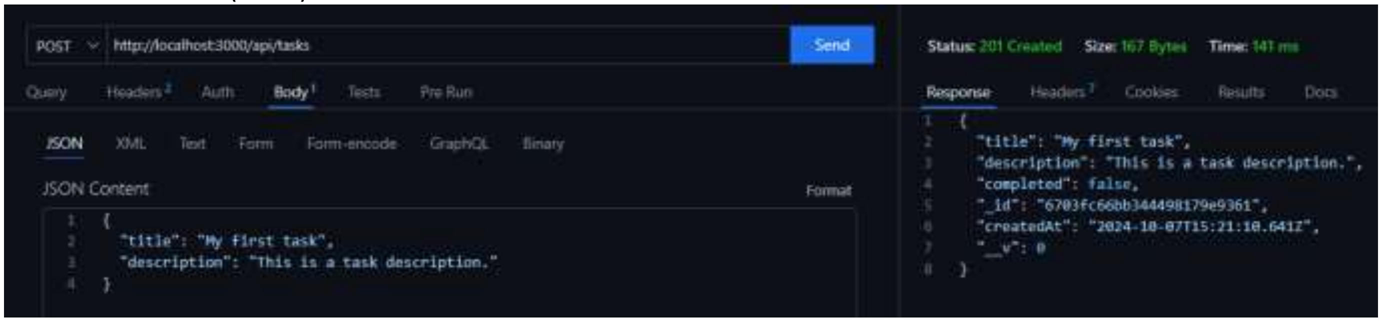
```
const mongoose = require('mongoose');

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

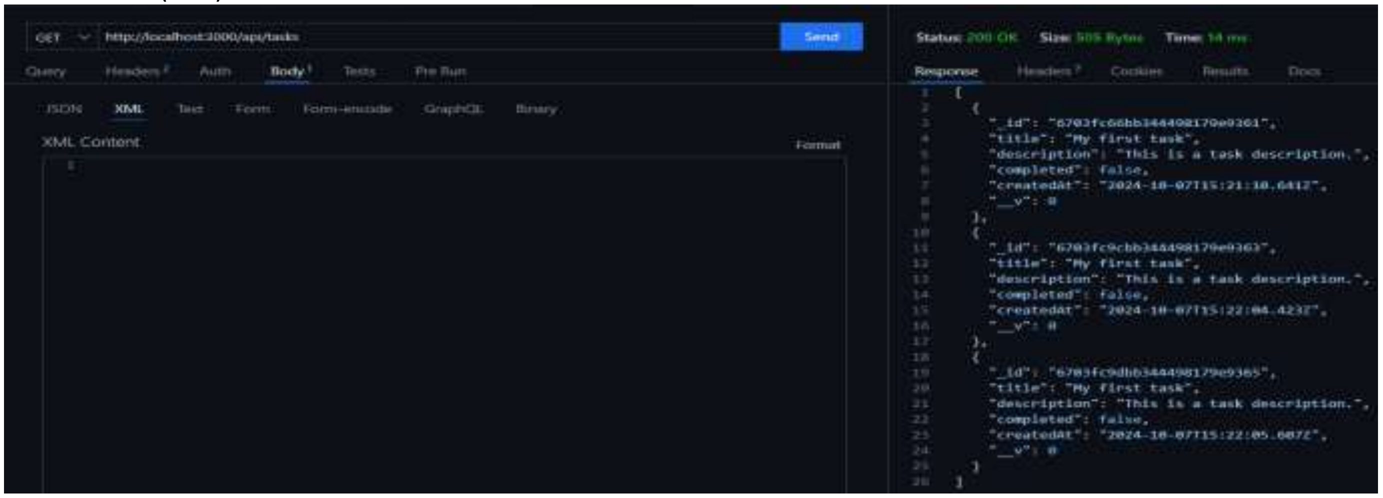
Create a new task (POST):



```
POST  http://localhost:3000/api/tasks
JSON Content
{
  "title": "My first task",
  "description": "This is a task description."
}
```

```
Status: 201 Created   Size: 167 Bytes   Time: 141 ms
{
  "title": "My first task",
  "description": "This is a task description.",
  "completed": false,
  "_id": "6703fc66bb344498179e9361",
  "createdAt": "2024-10-07T15:21:10.641Z",
  "__v": 0
}
```

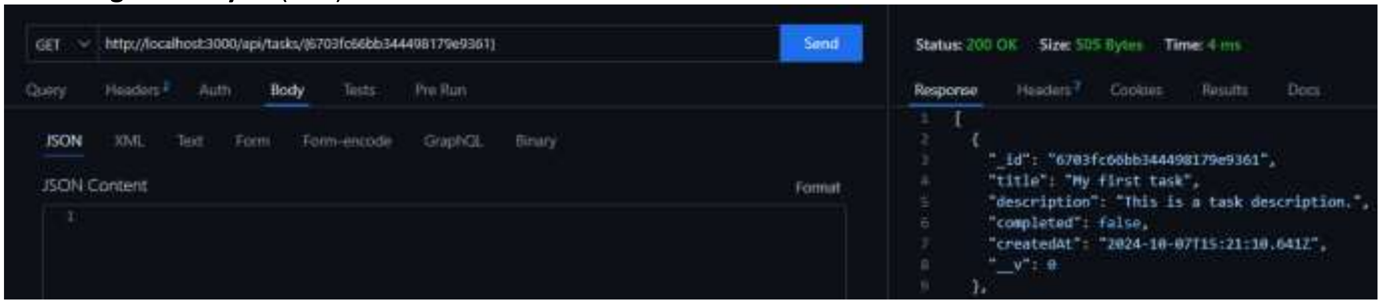
Get all tasks (GET):



```
GET  http://localhost:3000/api/tasks
```

```
Status: 200 OK   Size: 505 Bytes   Time: 14 ms
[
  {
    "_id": "6703fc66bb344498179e9361",
    "title": "My first task",
    "description": "This is a task description.",
    "completed": false,
    "createdAt": "2024-10-07T15:21:10.641Z",
    "__v": 0
  },
  {
    "_id": "6703fc9cbb344498179e9363",
    "title": "My first task",
    "description": "This is a task description.",
    "completed": false,
    "createdAt": "2024-10-07T15:22:04.423Z",
    "__v": 0
  },
  {
    "_id": "6703fc9dbb344498179e9365",
    "title": "My first task",
    "description": "This is a task description.",
    "completed": false,
    "createdAt": "2024-10-07T15:22:05.087Z",
    "__v": 0
  }
]
```

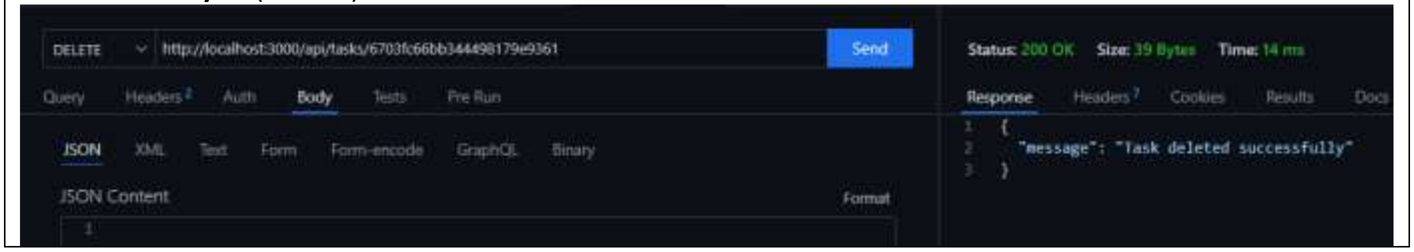
Get a single task by ID (GET):



```
GET  http://localhost:3000/api/tasks/{6703fc66bb344498179e9361}
```

```
Status: 200 OK   Size: 505 Bytes   Time: 4 ms
[
  {
    "_id": "6703fc66bb344498179e9361",
    "title": "My first task",
    "description": "This is a task description.",
    "completed": false,
    "createdAt": "2024-10-07T15:21:10.641Z",
    "__v": 0
  },
```

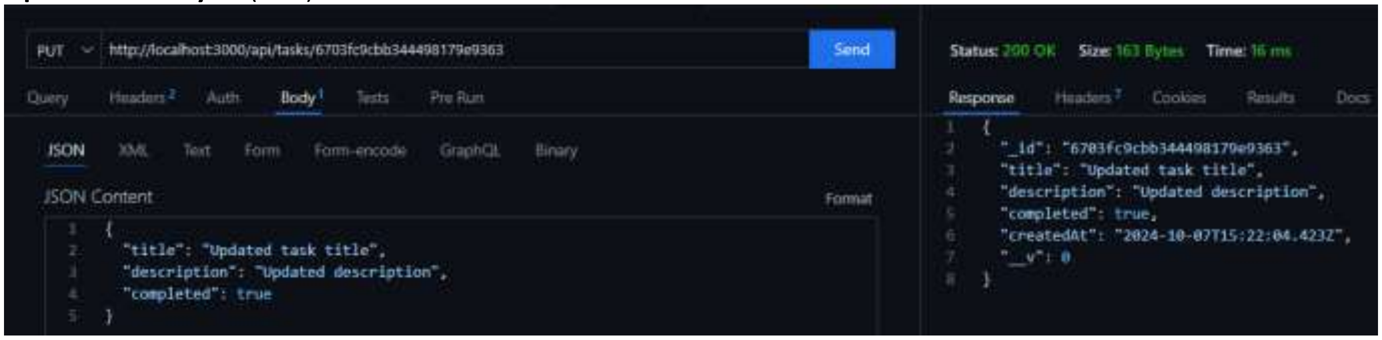
Delete a task by ID (DELETE):



```
DELETE  http://localhost:3000/api/tasks/6703fc66bb344498179e9361
```

```
Status: 200 OK   Size: 39 Bytes   Time: 14 ms
{
  "message": "Task deleted successfully"
}
```

Update a task by ID (PUT):



```
PUT  http://localhost:3000/api/tasks/6703fc9cbb344498179e9363
```

```
{
  "title": "Updated task title",
  "description": "Updated description",
  "completed": true
}
```

Status: 200 OK Size: 163 Bytes Time: 16 ms

```
{
  "_id": "6703fc9cbb344498179e9363",
  "title": "Updated task title",
  "description": "Updated description",
  "completed": true,
  "createdAt": "2024-10-07T15:22:04.423Z",
  "__v": 0
}
```

Signature :