6. Develop a CRUD-based dashboard for project management. Users can Create, Read, Update, and Delete tasks, and update their status (Pending, In Progress, Completed). Use MongoDB to store tasks and Node.js with Express to build the API.

app.js

```
const express = require('express');
const dotenv = require('dotenv').config();
const connectDB = require('./config/dbConnect');
const tasksRoutes = require('./routes/taskRoutes');
connectDB();
const app = express();
app.use(express.json());
// main routes
app.use("/api/tasks",tasksRoutes);
// catch-all route (for wrong URLs)
app.use((req, res) => {
res.status(404).json({ message: 'Route not found' });
});
app.listen(3000,()=>{
console.log("server is running at port 3000");
})
```

routes/taskRoutes.js

```
const express = require("express");
const router = express.Router();
const { getAllTasks,createTask,editTask,deleteTask} =
require('../controllers/taskController');

//router.route('/').get(getContact)
router.get('/', getAllTasks);
router.post('/',createTask);
router.put('/:id',editTask);
router.delete('/:id',deleteTask);

module.exports = router;
```

config/dbConnect.js

```
const mongoose = require("mongoose");
const dotenv = require("dotenv").config()

const connectDB = async()=>{
    try{
        const connect = await mongoose.connect(process.env.CONNECTION_STRING);
        console.log("Database connected");
    }
    catch(err){

        console.log(err);
    }
}
module.exports = connectDB;
```

models/taskModel.js

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

const tasksSchema = mongoose.Schema({
    title: {
     type: String,
     required: true
   },
   description: {
     type: String,
     required: false
   },
   status: {
     type: String,
     enum: ['Pending', 'In Progress', 'Completed'],
     default: 'Pending'
   },
    })

    module.exports = mongoose.model("Task",tasksSchema);
```

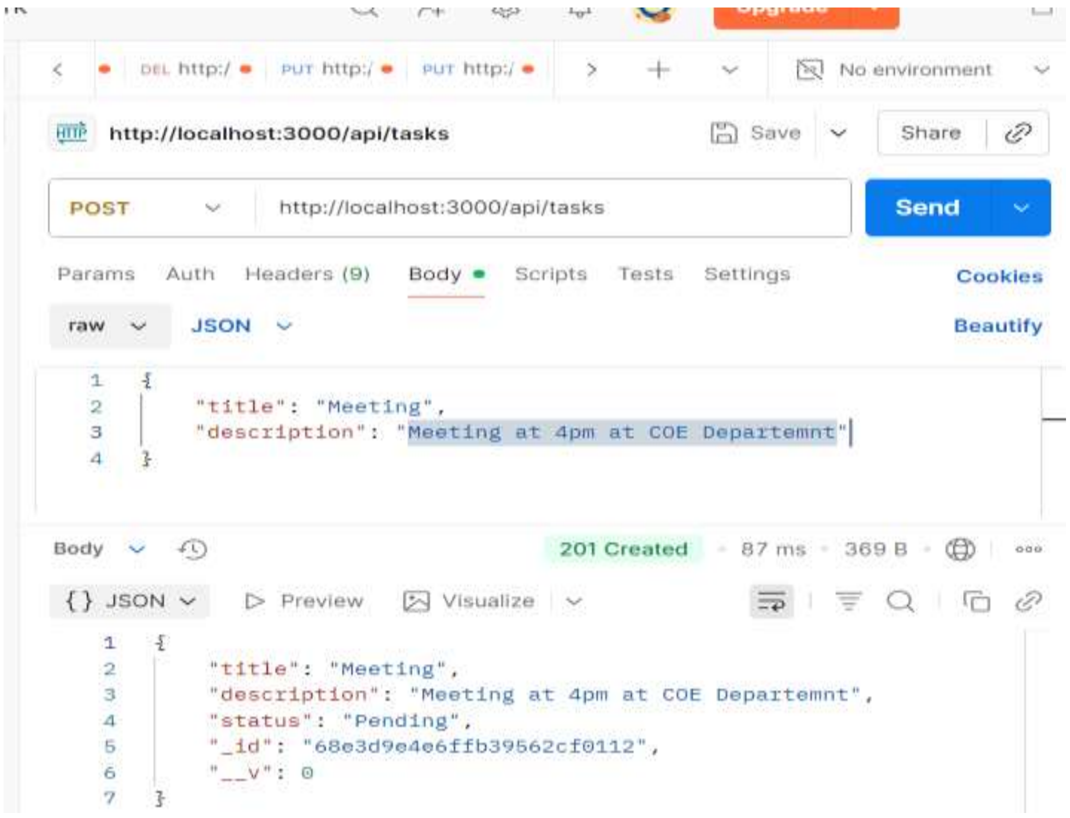
controllers/taskController.js

```
const Task = require("../models/taskModel");

const getAllTasks = async (req, res) => {
    try {
        // Find all tasks
        const tasks = await Task.find()
        res.status(200).json(tasks);
    }
    catch (error) {
        res.status(500).json({ message: 'Error retrieving tasks', error:
error.message });
    }
}
const createTask = async (req, res) => {
    try {
    const {title,description,status} = req.body;
    if(!title || !description){
        res.status(400).json({message:"Title and description is mandatory"});
    }
    const task = await Task.create({
        title,
        description,
        status
    });
    res.status(201).json(task);
    }
    catch (error) {
        res.status(500).json({ message: 'Error retrieving tasks', error:
error.message });
    }
}
const editTask = async (req, res) => {
    try {
    /* * Find the task by ID and update it.
      * { new: true } returns the modified document rather than the original.
      * { runValidators: true } ensures status enum rules are applied to the
update.
      */
    const task = await Task.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true, runValidators: true }
    );

    if (!task) {
      return res.status(404).json({ message: 'Task not found for update' });
    }
    res.status(200).json(task);
  } catch (error) {
    res.status(400).json({ message: 'Error updating task', error:
error.message });
  }

}

const deleteTask = async (req, res) => {
    try {
    const task = await Task.findByIdAndDelete(req.params.id);

    if (!task) {
      return res.status(404).json({ message: 'Task not found for deletion' });
    }
    // Respond with status 204 (No Content) for successful deletion
    res.status(204).send();
  } catch (error) {
    res.status(500).json({ message: 'Error deleting task', error: 
error.message });
  }
}

module.exports = {
    getAllTasks,
    createTask,
    editTask,
    deleteTask
}
```

.env

```
CONNECTION_STRING=”ATTACH YOUR MONGODB CONNECTION STRING HERE”
```

OUTPUT:



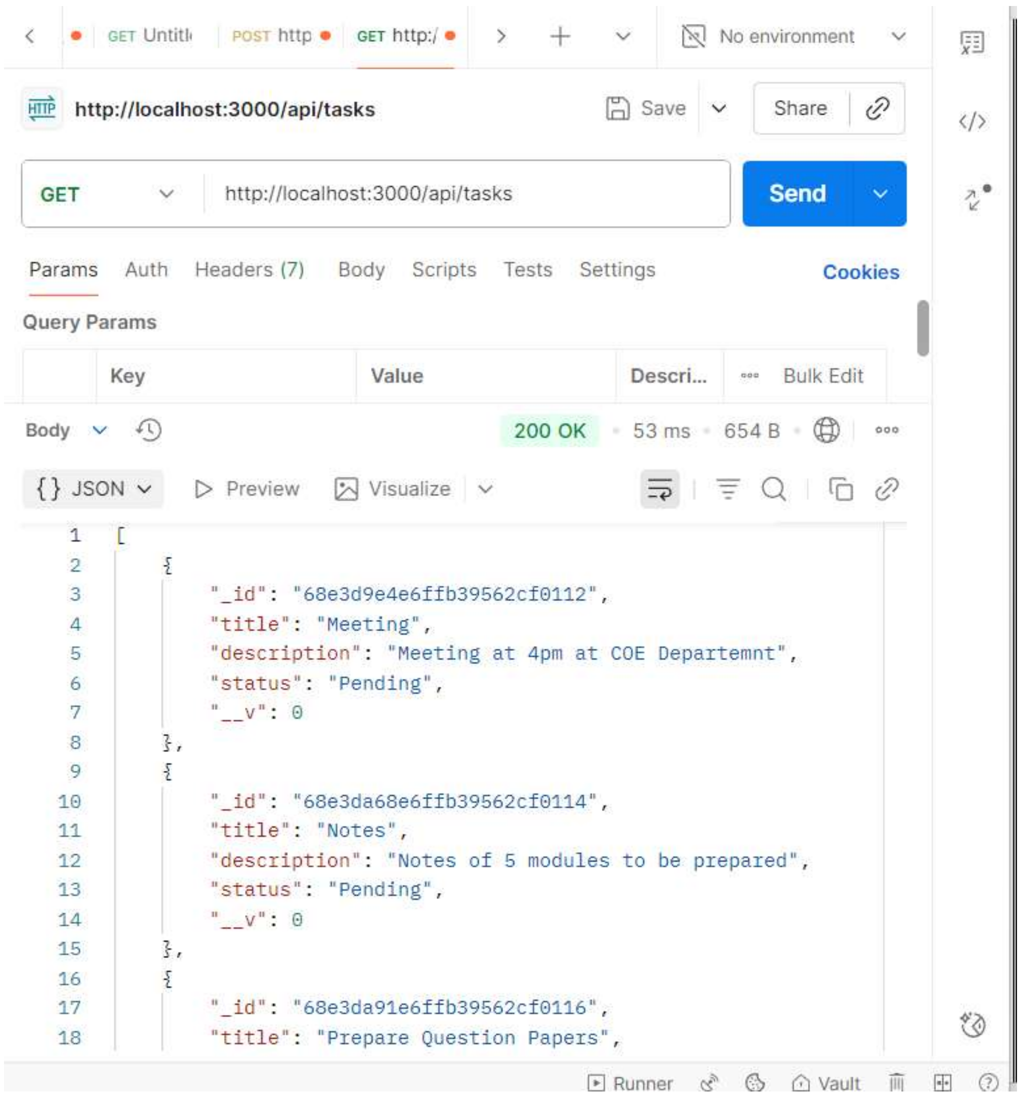
http://localhost:3000/api/tasks

POST http://localhost:3000/api/tasks

```
{
    "title": "Meeting",
    "description": "Meeting at 4pm at COE Departemnt"
}
```

201 Created · 87 ms · 369 B

```
{
    "title": "Meeting",
    "description": "Meeting at 4pm at COE Departemnt",
    "status": "Pending",
    "_id": "68e3d9e4e6ffb39562cf0112",
    "__v": 0
}
```



http://localhost:3000/api/tasks

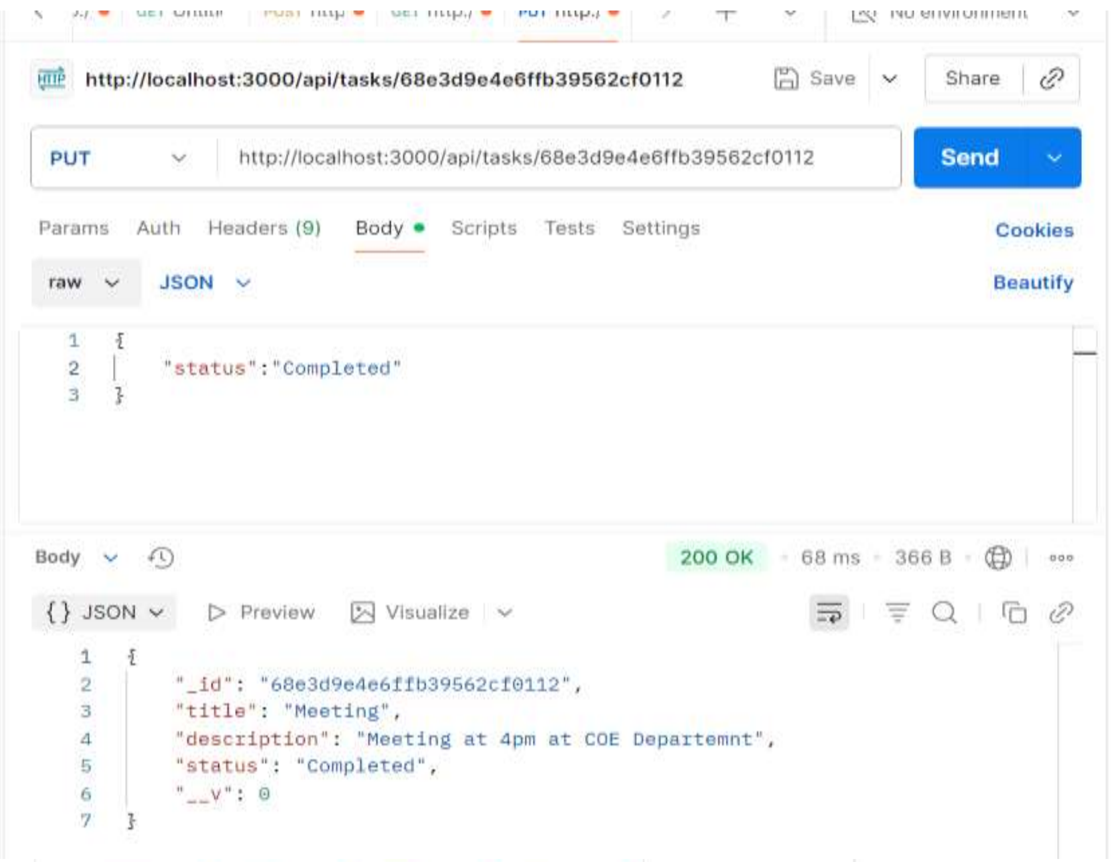
GET http://localhost:3000/api/tasks

Query Params

Key	Value	Descri...

200 OK · 53 ms · 654 B

```
[
    {
        "_id": "68e3d9e4e6ffb39562cf0112",
        "title": "Meeting",
        "description": "Meeting at 4pm at COE Departemnt",
        "status": "Pending",
        "__v": 0
    },
    {
        "_id": "68e3da68e6ffb39562cf0114",
        "title": "Notes",
        "description": "Notes of 5 modules to be prepared",
        "status": "Pending",
        "__v": 0
    },
    {
        "_id": "68e3da91e6ffb39562cf0116",
        "title": "Prepare Question Papers",
```



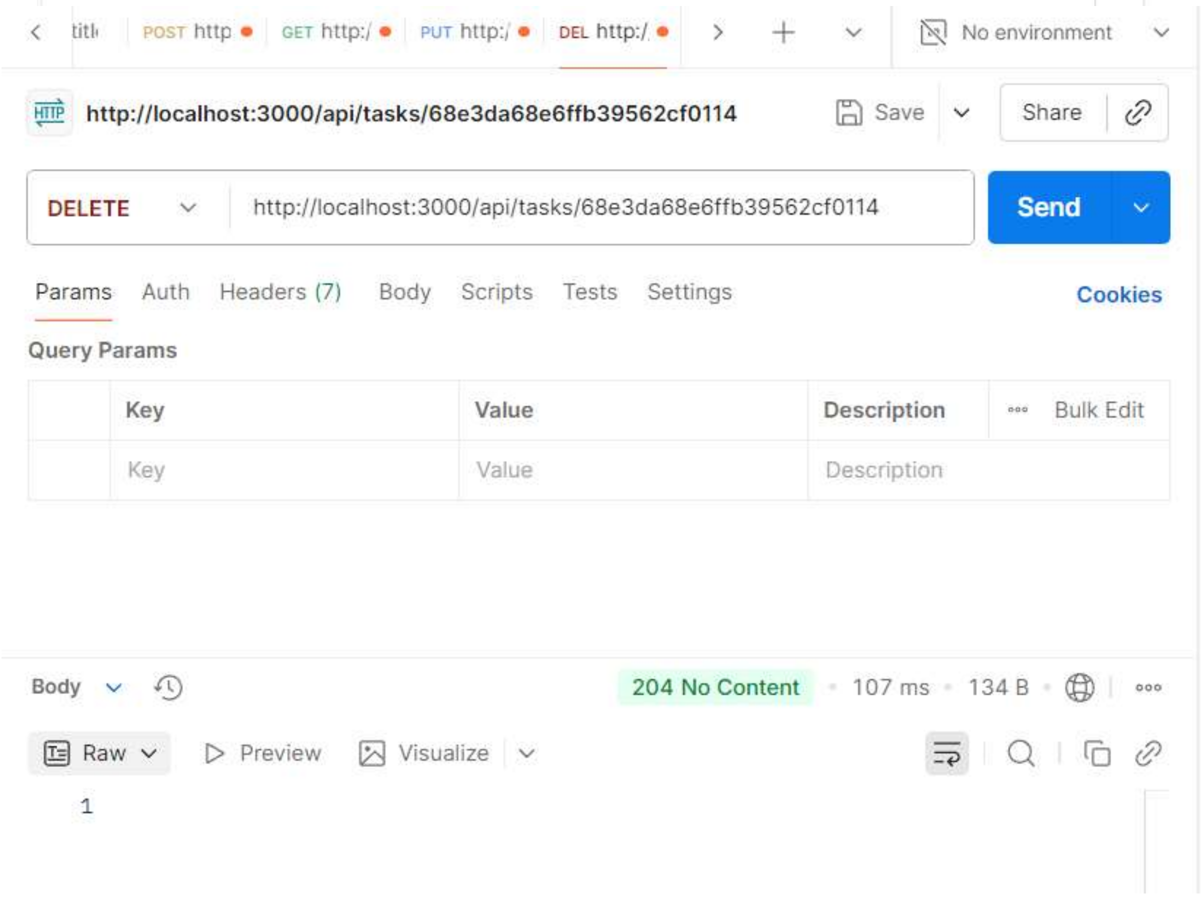
http://localhost:3000/api/tasks/68e3d9e4e6ffb39562cf0112

PUT http://localhost:3000/api/tasks/68e3d9e4e6ffb39562cf0112

```
{
  "status":"Completed"
}
```

200 OK · 68 ms · 366 B

```
{
    "_id": "68e3d9e4e6ffb39562cf0112",
    "title": "Meeting",
    "description": "Meeting at 4pm at COE Departemnt",
    "status": "Completed",
    "__v": 0
}
```



http://localhost:3000/api/tasks/68e3da68e6ffb39562cf0114

DELETE http://localhost:3000/api/tasks/68e3da68e6ffb39562cf0114

Query Params

Key	Value	Description
Key	Value	Description

204 No Content · 107 ms · 134 B