1. **Develop the important backend (Node Js, Express Js, MongoDB) functionalities for a task manager application.**

- Create package.json file and install required libraries (express, mongoose)
- Create task model (schema)
- **TaskModel.js**

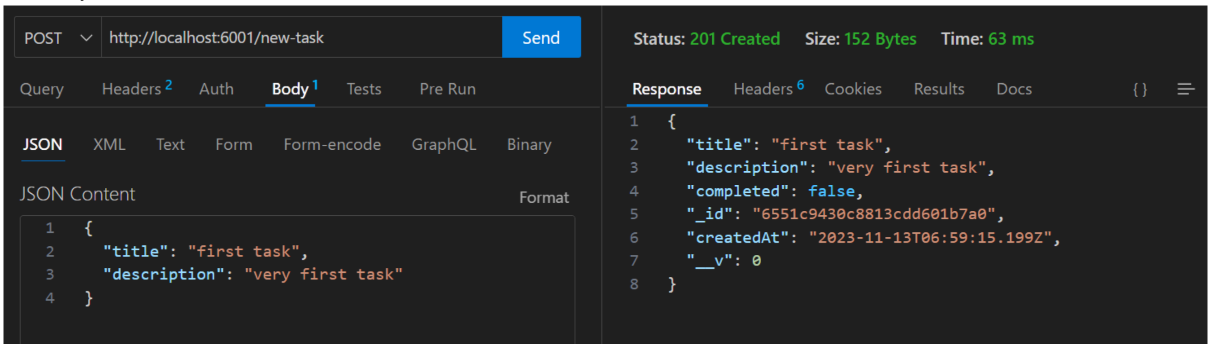
```
•   const mongoose = require('mongoose');
•
•   const taskSchema = new mongoose.Schema({
•     title: {
•       type: String,
•       required: true,
•     },
•     description: String,
•     createdAt: {
•       type: Date,
•       default: Date.now,
•     },
•   });
•
•   const Task = mongoose.model('Task', taskSchema);
•   module.exports = Task;
```

- Now create index.js file and write server code:

```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = 6001;
app.use(express.json());
const Task = require('./TaskModel.js')

mongoose.connect('mongodb://localhost:27017/database_name', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(()=>{

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

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

    // Update a task
        app.put('/:id', async (req, res) => {

            try{
                const task = await Task.findById(req.params.id);
                task.title = req.body.title;
                task.description = req.body.description;
                const updatedTask = await task.save();
                res.status(201).json(updatedTask);
            }catch(err){
                res.status(400).json({ message: err.message });
            }

        });

    // Delete a task
        app.delete('/delete-task/:id', async (req, res) => {
            try{
                await Task.deleteOne({_id: req.params.id});

                res.status(201).json({message: "task deleted"});
            }catch(err){
                res.status(400).json({ message: err.message });
            }
        });
}
)

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

If you want to verify the working of the server, use **thunderclient extension** in vs code or **postman**.

Example:



```
http://localhost:6001/new-task
```

```
{
  "title": "first task",
  "description": "very first task"
}
```

Status: 201 Created   Size: 152 Bytes   Time: 63 ms

```
{
  "title": "first task",
  "description": "very first task",
  "completed": false,
  "_id": "6551c9430c8813cdd601b7a0",
  "createdAt": "2023-11-13T06:59:15.199Z",
  "__v": 0
}
```

2. **Develop the backend (Node Js, Express Js, MongoDB) for a CRUD application.**

   Same as the above question. Crud – create, read, update, delete. Perform the same code from the previous question and use a name you wish apart from tasks.

4. **Develop the backend for a job portal using Node JS, Express JS, MongoDB. Perform important functionalities such as add new job, update job, apply for job, approve job application, etc.,**

same as previous questions. Use jobs instead of tasks. Take another schema for applications.

JobsModel.js:

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

const jobSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description:{
    type: String,
  },
  companyName: {
    type: String
  }
});


const applicationSchema = new mongoose.Schema({
    applicantId:{
        type: String
    },
    applicantName: {
        type: String
    },
    JobId:{
        type: String,
    },
    status: {
        type: String,
        default: "Pending"
    }
})

const Job = mongoose.model('Job', jobSchema);
const Application = mongoose.model('Application', applicationSchema);

module.exports = Job;
module.exports = Application;
```

Index.js:

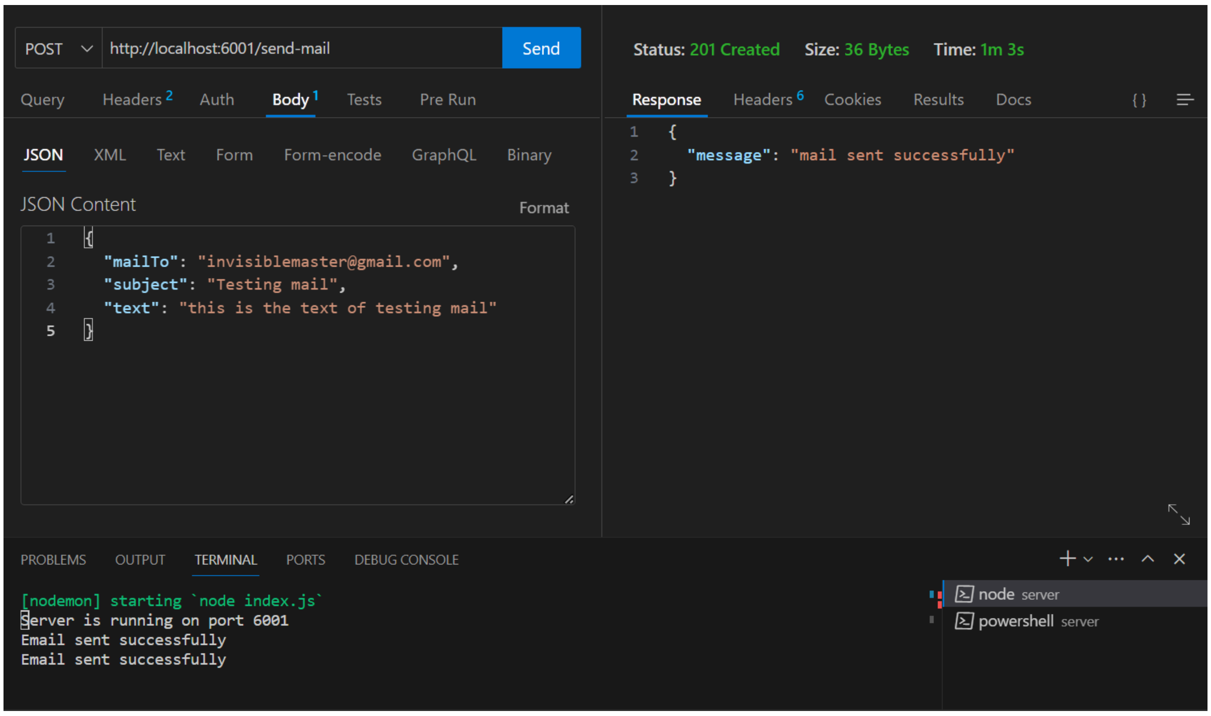
```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = 6001;
app.use(express.json());
const {Job, Application}= require('./TaskModel.js')

mongoose.connect('mongodb://localhost:27017/database_name', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(()=>{

    // Get all jobs
        app.get('/', async (req, res) => {
            try {
            const jobs = await Job.find();
            res.json(jobs);
            } catch (err) {
            res.status(500).json({ message: err.message });
            }
        });

    // Create a new job
        app.post('/new-job', async (req, res) => {
            const job = new Job({
                title: req.body.title,
                description: req.body.description,
            });
            try {
                const newjob = await Job.save();
                res.status(201).json(newjob);
            } catch (err) {
                res.status(400).json({ message: err.message });
            }
        });

    // Update a job
        app.put('/:id', async (req, res) => {

            try{
                const job = await Job.findById(req.params.id);
                job.title = req.body.title;
                job.description = req.body.description;
                const updatedjob = await job.save();
                res.status(201).json(updatedjob);
            }catch(err){
                res.status(400).json({ message: err.message });
            }

        });

    // Delete a job
        app.delete('/delete-task/:id', async (req, res) => {
            try{
                await Job.deleteOne({_id: req.params.id});

                res.status(201).json({message: "job deleted"});
            }catch(err){
                res.status(400).json({ message: err.message });
            }
        });

    //Apply for a job
        app.post('/apply-job', async(req, res)=>{
            const {jobId, applicantName, applicantId} = req.body;
            try{
                const application = new Application({jobId, applicantName, 
applicantId});

                const newApplication = await application.save();

                res.status(201).json(newApplication);
            }catch(err){
                res.status(400).json({ message: err.message });
            }
        })

    //Approve job application
        app.post('/approve-application/:id', async(req, res)=>{
            try{
                const application = await Application.findById(req.params.id);
                application.status = "Accepted";
                await application.save();
                res.status(201).json(application);
            }catch(err){
                res.status(400).json({ message: err.message });
            }
        })
}
)

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

3. **Develop and application to send emails (use libraries link Nodemailer, etc.,).**

- Create server files and add basic code.
- Install nodemailer (npm install nodemailer).
- Open ethereal.email and create new account

```
const express = require('express');
const app = express();
const PORT = 6001;
app.use(express.json());
const nodemailer = require('nodemailer');




    const transporter = nodemailer.createTransport({
        host: 'smtp.ethereal.email',
        port: 587,
        auth: {
            user: ‘replace this with user mail from ethereal’,
            pass: ‘replace this with the password from ethereal’
        }
    });


    // Send email
        app.post('/send-mail', async (req, res) => {
            const {mailTo, subject, text} = req.body;
            try {

                let mailDetails = {
                    from: 'naomie71@ethereal.email',
                    to: mailTo,
                    subject: subject,
                    text: text
                };

                transporter.sendMail(mailDetails, function(err, data) {
                    if(err) {
                        console.log('Error Occurs');
                    } else {
                        console.log('Email sent successfully');
                        res.status(201).json({ message: "mail sent
successfully" });
                    }
                });

            } catch (err) {
            res.status(500).json({ message: err.message });
            }
        });


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

Verify this with postman or thunderclient



http://localhost:6001/send-mail

```
{
    "mailTo": "invisiblemaster@gmail.com",
    "subject": "Testing mail",
    "text": "this is the text of testing mail"
}
```

Status: 201 Created   Size: 36 Bytes   Time: 1m 3s

```
{
  "message": "mail sent successfully"
}
```

```
[nodemon] starting `node index.js`
Server is running on port 6001
Email sent successfully
Email sent successfully
```