

ASSIGNMENT-4

Name: vadala yamini

Roll no:20NN1A1261

E-Mail: yaminivadala27@gmail.com

**College: Vignan's Nirula Institute of Science
and Technology for Women's**

Index.js

```
express = require('express') const mongoose =
require ('mongoose'); const Product =
require('./models/product.model.js'); const app =
express() app.use(express.json());

//reading all products app.get('/', function (req,
res) { res.send("hello from the node api update");
}); app.get('/api/products', async

(req,res)=> {

  try { const products = await
    Product.find({});
    res.status(200).json(products);

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

    }
});

  //read api but by only one product
app.get('/api/product/:id', async (req,res) =>{
try{ const {id} = req.params; const product =
await Product.findById(id); res.status(200).json(
product );

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

//creat api
app.post('/api/products',async (req,res)=>{ try{
const product = await Product.create(req.body);
res.status(200).json(product);
```

```

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

  //update a product app.put('/api/product/:id',
  async (req,res) => { try { const{id} =
    req.params;

    const product = await Product.findByIdAndUpdate(id ,
req.body);

    if(!product){ return res.status(404).json({message:"Product
      not found"}});
    }

    const updatedProduct = await Product.findById(id);
    res.status(200).json(updatedProduct);

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

  //delete a product

  app.delete("/api/product/:id", async(req,res)=>{
    try{ const{id}= req.params;
      const product = await Product.findByIdAndDelete(id); if
      (!product){
        return res.status(404).json({message: "Product not found"}});
      } res.status(200).json({message:"Product deleted

        successfully"}});

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

```

```

//here first i connected db and then listened to the port

mongoose.connect("mongodb+srv://akashvaddi333:K5m18vy6fB6aU7K2@cluster0.h
p9gamr.mongodb.net/Node-API?retryWrites=true&w=majority&appName=Cluster0")
.then(() => {console.log('Connected!');
app.listen(3000, () =>{ console.log('server is
running on port 3000') });
});

```

Package. Json

```

{
  "name": "aka-qpi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "serve": "node index.js",
    "dev": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.3",
    "mongodb": "^6.5.0",
    "mongoose": "^8.2.2"
  },
  "devDependencies": {
    "nodemon": "^3.1.0"
  }
}

```

Product.model.js

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

const ProductSchema = mongoose.Schema(
  { name: { type:String,
           required: [true,"proto"],

           }, quantity:{
            type:Number,
            required:true,
            default:0
          },
    image:{
      type:String,
      required: false
    },

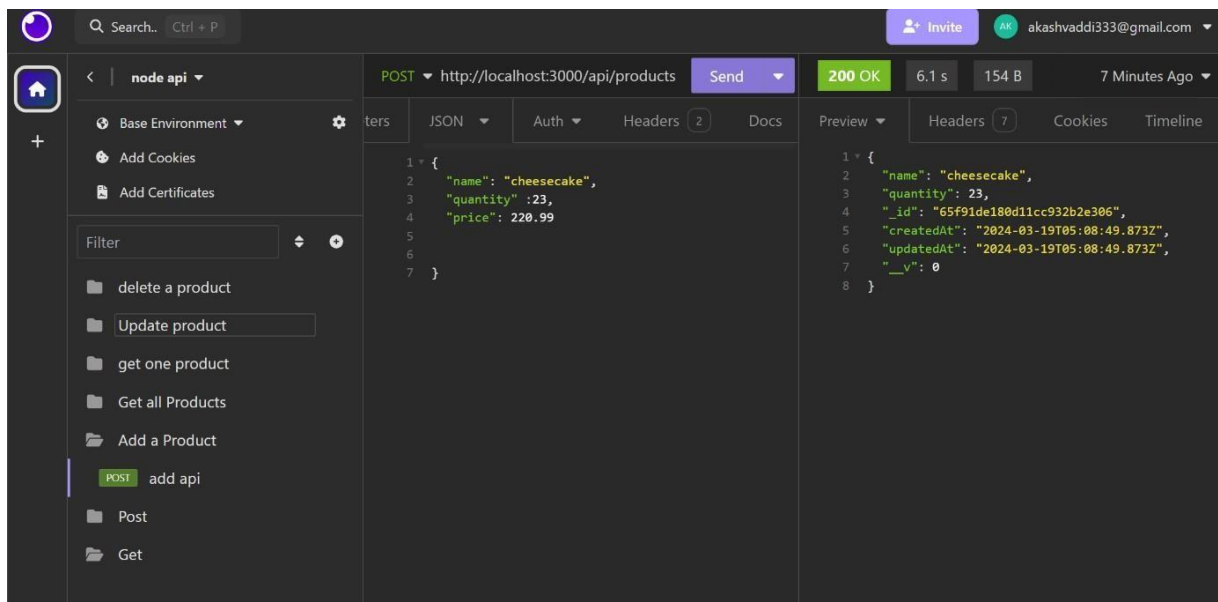
  } ,

  { timestamps: true,
  }
);

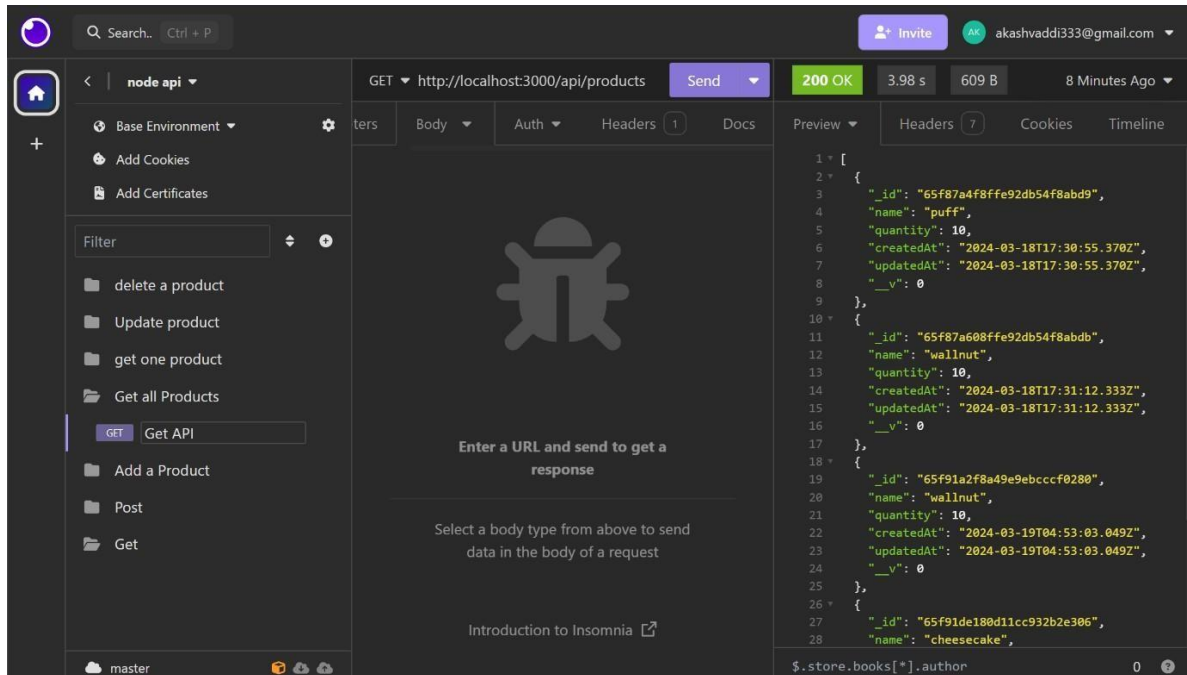
const Product = mongoose.model("Product",ProductSchema);
module.exports= Product;
```

□ *CRUD operations*

Create api:



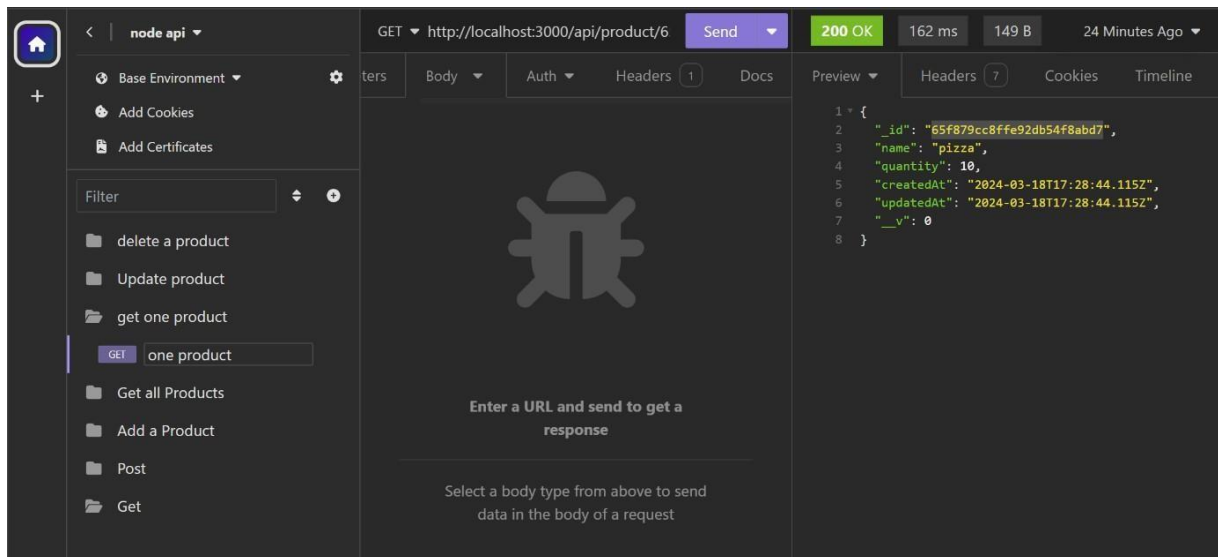
Read Api:



Insomnia API client interface showing a GET request to `http://localhost:3000/api/products`. The response is a 200 OK status with a 3.98 s response time and 609 B body. The response body is a JSON array of three product objects.

```
1 * [
2 * {
3   "_id": "65f87a4f8ffe92db54f8abd9",
4   "name": "puff",
5   "quantity": 10,
6   "createdAt": "2024-03-18T17:30:55.370Z",
7   "updatedAt": "2024-03-18T17:30:55.370Z",
8   "__v": 0
9 },
10 * {
11   "_id": "65f87a608ffe92db54f8abdb",
12   "name": "wallnut",
13   "quantity": 10,
14   "createdAt": "2024-03-18T17:31:12.333Z",
15   "updatedAt": "2024-03-18T17:31:12.333Z",
16   "__v": 0
17 },
18 * {
19   "_id": "65f91a2f8a49e9ebccc0280",
20   "name": "wallnut",
21   "quantity": 10,
22   "createdAt": "2024-03-19T04:53:03.049Z",
23   "updatedAt": "2024-03-19T04:53:03.049Z",
24   "__v": 0
25 },
26 * {
27   "_id": "65f91de180d11cc932b2e306",
28   "name": "cheesecake",
29   "quantity": 10,
30   "createdAt": "2024-03-19T04:53:03.049Z",
31   "updatedAt": "2024-03-19T04:53:03.049Z",
32   "__v": 0
33 }
```

Read one Api:



Insomnia API client interface showing a GET request to `http://localhost:3000/api/product/6`. The response is a 200 OK status with a 162 ms response time and 149 B body. The response body is a JSON object representing a single product.

```
1 * {
2   "_id": "65f879cc8ffe92db54f8abd7",
3   "name": "pizza",
4   "quantity": 10,
5   "createdAt": "2024-03-18T17:28:44.115Z",
6   "updatedAt": "2024-03-18T17:28:44.115Z",
7   "__v": 0
8 }
```

Update Api:

The screenshot shows the Insomnia API client interface. On the left, a sidebar lists API endpoints under 'node api'. The 'update product' endpoint is selected, showing a PUT method. The main panel displays the request details for the URL 'http://localhost:3000/api/product/6'. The request body is a JSON object:

```
{ 1: { 2: "name": "biryani" 3: }
```

. The response status is '200 OK' with a response time of 272 ms and a body size of 151 B. The response body is a JSON object:

```
{ 1: { 2: "_id": "65f879cc8ffe92db54f8abd7", 3: "name": "biryani", 4: "quantity": 10, 5: "createdAt": "2024-03-18T17:28:44.115Z", 6: "updatedAt": "2024-03-19T04:57:08.514Z", 7: "__v": 0 8: }
```

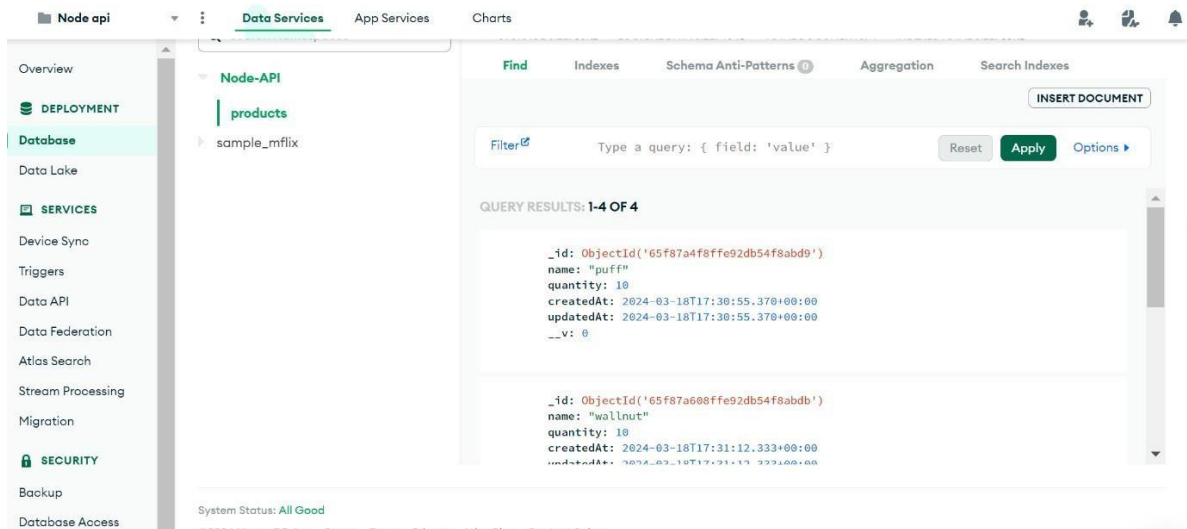
Delete Api:

The screenshot shows the Insomnia API client interface. On the left, the 'delete product' endpoint is selected, showing a DELETE method. The main panel displays the request details for the URL 'http://localhost:3000/api/produ'. The response status is '200 OK' with a response time of 1.14 s and a body size of 42 B. The response body is a JSON object:

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

. The interface also includes a large bug icon and a message: 'Enter a URL and send to get a response'. Below this, it says: 'Select a body type from above to send data in the body of a request'. At the bottom, there is a link to 'Introduction to Insomnia'.

MongoDB : (final view)



AKA QPI:

