

Assignment-3

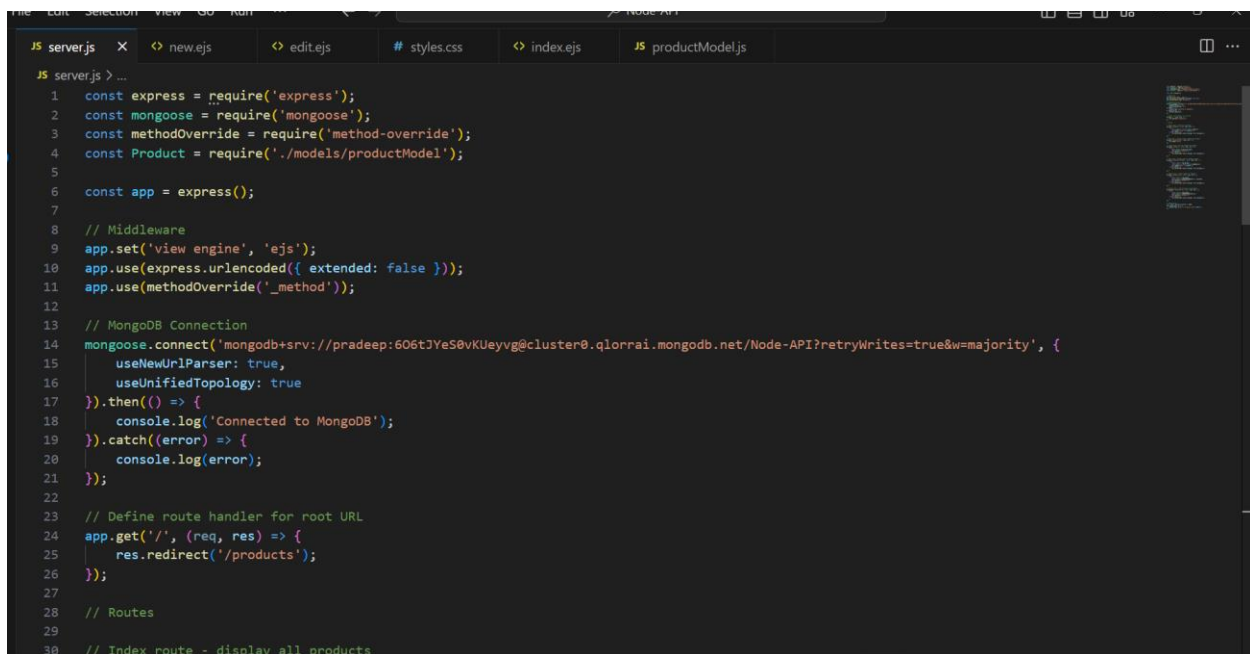
NAME: GUDE PRADEEP

College: Kallam Haranadhareddy Institute of Technology

E mail: 208x1a0520@khitguntur.ac.in

All codes :

Server.js code :



```
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const methodOverride = require('method-override');
4  const Product = require('./models/productModel');
5
6  const app = express();
7
8  // Middleware
9  app.set('view engine', 'ejs');
10 app.use(express.urlencoded({ extended: false }));
11 app.use(methodOverride('_method'));
12
13 // MongoDB Connection
14 mongoose.connect('mongodb+srv://pradeep:606t3YeS0vKUeyvg@cluster0.qlorrai.mongodb.net/Node-API?retryWrites=true&w=majority', {
15   useNewUrlParser: true,
16   useUnifiedTopology: true
17 }).then(() => {
18   console.log('Connected to MongoDB');
19 }).catch((error) => {
20   console.log(error);
21 });
22
23 // Define route handler for root URL
24 app.get('/', (req, res) => {
25   res.redirect('/products');
26 });
27
28 // Routes
29
30 // Index route - display all products
```

This screenshot shows the first part of a Node.js application in VS Code. The editor has tabs for 'server.js', 'new.ejs', 'edit.ejs', 'styles.css', 'index.ejs', and 'productModel.js'. The 'server.js' file contains the following code:

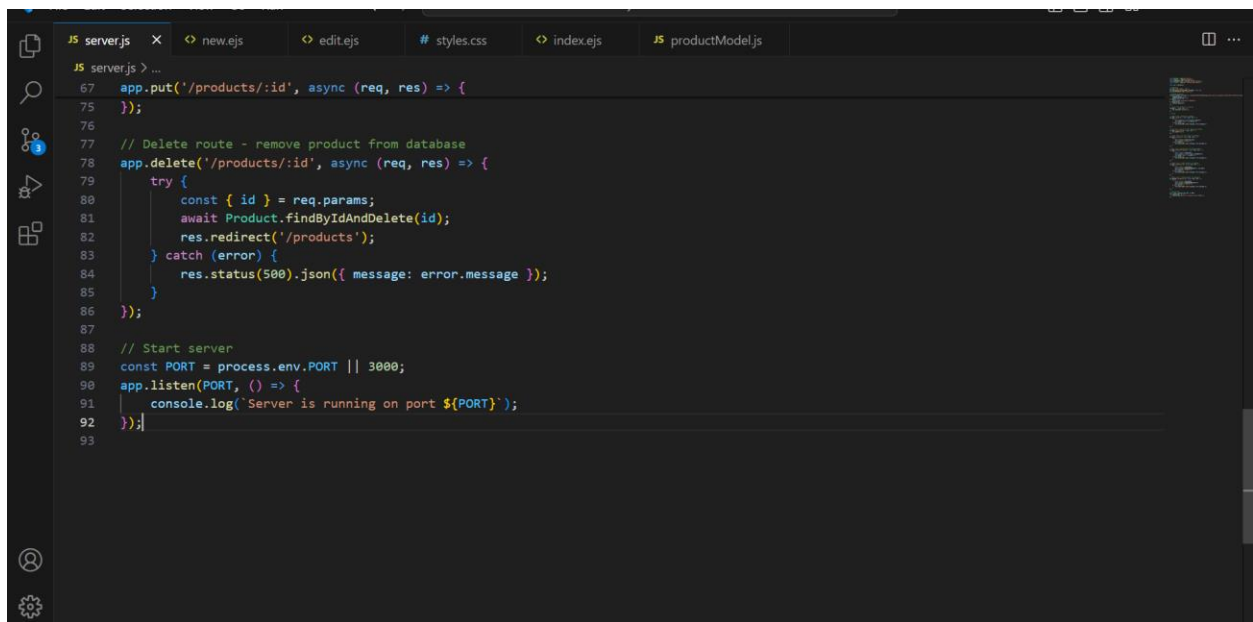
```
28 // Routes
29
30 // Index route - display all products
31 app.get('/products', async (req, res) => {
32   try {
33     const products = await Product.find({});
34     res.render('index', { products });
35   } catch (error) {
36     res.status(500).json({ message: error.message });
37   }
38 });
39
40 // New route - display form for adding new product
41 app.get('/products/new', (req, res) => {
42   res.render('new');
43 });
44
45 // Create route - add new product to database
46 app.post('/products', async (req, res) => {
47   try {
48     await Product.create(req.body);
49     res.redirect('/products');
50   } catch (error) {
51     res.status(500).json({ message: error.message });
52   }
53 });
54
55 // Edit route - display form for editing product
56 app.get('/products/:id/edit', async (req, res) => {
```

The status bar at the bottom indicates 'Ln 92, Col 4', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

This screenshot shows the second part of the Node.js application in VS Code, continuing from the previous file. The code in 'server.js' includes:

```
56 app.get('/products/:id/edit', async (req, res) => {
57   try {
58     const { id } = req.params;
59     const product = await Product.findById(id);
60     res.render('edit', { product });
61   } catch (error) {
62     res.status(500).json({ message: error.message });
63   }
64 });
65
66 // Update route - update product in database
67 app.put('/products/:id', async (req, res) => {
68   try {
69     const { id } = req.params;
70     await Product.findByIdAndUpdate(id, req.body);
71     res.redirect('/products');
72   } catch (error) {
73     res.status(500).json({ message: error.message });
74   }
75 });
76
77 // Delete route - remove product from database
78 app.delete('/products/:id', async (req, res) => {
79   try {
80     const { id } = req.params;
81     await Product.findByIdAndDelete(id);
82     res.redirect('/products');
83   } catch (error) {
84     res.status(500).json({ message: error.message });
85   }
86 });
```

The status bar at the bottom indicates 'Ln 92, Col 4', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.



```
JS server.js x new.ejs edite.js # styles.css indexe.js JS productModel.js
67 app.put('/products/:id', async (req, res) => {
75 });
76
77 // Delete route - remove product from database
78 app.delete('/products/:id', async (req, res) => {
79   try {
80     const { id } = req.params;
81     await Product.findByIdAndDelete(id);
82     res.redirect('/products');
83   } catch (error) {
84     res.status(500).json({ message: error.message });
85   }
86 });
87
88 // Start server
89 const PORT = process.env.PORT || 3000;
90 app.listen(PORT, () => {
91   console.log(`Server is running on port ${PORT}`);
92 });
93
```

Mongo db connection:

Access M
vices
24-03-07
O
R
ES REPL
iciently l
agging.
ai (ap-sc
hard-00
shard-0
hard-00

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

| Driver | Version |
|---------|--------------|
| Node.js | 5.5 or later |

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

☐ View full code sample

```
mongodb+srv://pradeep:<password>@cluster0.qlorrai.mongodb.net/?  
retryWrites=true&w=majority&appName=Cluster0
```

Replace **<password>** with the password for the **pradeep** user. Ensure any option params are [URL encoded](#).

RESOURCES

[Get started with the Node.js Driver](#)
[Access your Database Users](#)

[Node.js Starter Sample App](#)
[Troubleshoot Connections](#)

```
// MongoDB Connection  
mongoose.connect('mongodb+srv://pradeep:606tJYeS0vKUeyvg@cluster0.qlorrai.mongodb.net/Node-API?retryWrites=true&w=majority', {  
  useNewUrlParser: true,  
  useUnifiedTopology: true  
}).then(() => {  
  console.log('Connected to MongoDB');  
}).catch((error) => {  
  console.log(error);  
});
```

Routes:

```
// Create route - add new product to database
app.post('/products', async (req, res) => {
  try {
    await Product.create(req.body);
    res.redirect('/products');
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

```
// Edit route - display form for editing product
app.get('/products/:id/edit', async (req, res) => {
  try {
    const { id } = req.params;
    const product = await Product.findById(id);
    res.render('edit', { product });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

```
// Update route - update product in database
app.put('/products/:id', async (req, res) => {
  try {
    const { id } = req.params;
    await Product.findByIdAndUpdate(id, req.body);
    res.redirect('/products');
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

```
// Delete route - remove product from database
app.delete('/products/:id', async (req, res) => {
  try {
    const { id } = req.params;
    await Product.findByIdAndDelete(id);
    res.redirect('/products');
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

Middleware:

```
// Middleware
app.set('view engine', 'ejs');
app.use(express.urlencoded({ extended: false }));
app.use(methodOverride('_method'));
```

Models/ProductModel.js:

```
models > JS productModel.js > ...
1  const mongoose = require('mongoose');
2
3  const productSchema = new mongoose.Schema({
4    name: {
5      type: String,
6      required: [true, "Please enter a product name"]
7    },
8    quantity: {
9      type: Number,
10     required: true,
11     default: 0
12   },
13   price: {
14     type: Number,
15     required: true,
16   },
17   image: {
18     type: String,
19     required: false,
20   }
21 }, { timestamps: true });
22
23 const Product = mongoose.model('Product', productSchema);
24
25 module.exports = Product;
26 |
```

Views:

Edit.ejs:

```

views > edit.ejs > html > body > div.container
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Edit Product</title>
7    <link rel="stylesheet" href="/styles/style.css"> <!-- Link to your CSS file -->
8  </head>
9  <body>
10   <div class="container">
11     <h1>Edit Product</h1>
12     <form action="/products/<%= product.id %>?_method=PUT" method="post">
13       <label for="name">Product Name</label>
14       <input type="text" id="name" name="name" value="<%= product.name %>" required>
15       <label for="quantity">Quantity</label>
16       <input type="number" id="quantity" name="quantity" value="<%= product.quantity %>" required>
17       <label for="price">Price</label>
18       <input type="number" id="price" name="price" value="<%= product.price %>" required>
19       <label for="image">Image URL</label>
20       <input type="text" id="image" name="image" value="<%= product.image %>">
21       <button type="submit">Update Product</button>
22     </form>
23   </div>
24 </body>
25 </html>
26

```

Index.ejs:


```
views > index.ejs > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Product List</title>
7   <!-- Bootstrap CSS -->
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
9   <!-- Custom CSS -->
10  <style>
11    /* Custom styles */
12    .product-card {
13      margin-bottom: 20px;
14      border-radius: 10px;
15      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
16    }
17
18    .product-image {
19      max-width: 100%;
20      max-height: 200px; /* Adjust the maximum height as needed */
21      border-radius: 10px 10px 0 0;
22      object-fit: cover; /* Ensure the image covers the entire container */
23    }
24
25    .product-info {
26      padding: 20px;
27    }
28  </style>
29 </head>
30 <body>
```

```
views > index.ejs > html > head > style
2 <html lang="en">
3 </head>
29 </head>
30 <body>
31   <div class="container mt-5">
32     <h1 class="text-center mb-4">Product List</h1>
33     <div class="text-center mt-4">
34       <form action="/products/new" method="get">
35         <button class="btn btn-success">Add New Product</button>
36       </form>
37     </div>
38     <div class="row">
39       <% products.forEach(product => { %>
40         <div class="col-lg-4">
41           <div class="card product-card">
42             ">
43             <div class="card-body product-info">
44               <h5 class="card-title"><%= product.name %></h5>
45               <p class="card-text">Price: $<%= product.price %></p>
46               <p class="card-text">Quantity: <%= product.quantity %></p>
47               <form action="/products/<%= product.id %>/edit" method="get">
48                 <button class="btn btn-primary">Edit</button>
49               </form>
50               <form action="/products/<%= product.id %>?method=DELETE" method="post">
51                 <button class="btn btn-danger">Delete</button>
52               </form>
53             </div>
54           </div>
55         </div>
56       <% }); %>
```

```
39       <% products.forEach(product => { %>
50         <div class="col-lg-4">
51           <div class="card product-card">
52             ">
53             <div class="card-body product-info">
54               <h5 class="card-title"><%= product.name %></h5>
55               <p class="card-text">Price: $<%= product.price %></p>
56               <p class="card-text">Quantity: <%= product.quantity %></p>
57               <form action="/products/<%= product.id %>/edit" method="get">
58                 <button class="btn btn-primary">Edit</button>
59               </form>
60               <form action="/products/<%= product.id %>?method=DELETE" method="post">
61                 <button class="btn btn-danger">Delete</button>
62               </form>
63             </div>
64           </div>
65         </div>
66       <% }); %>
```

New.ejs

```
JS server.js • new.ejs x edit.ejs # styles.css > index.ejs JS productModel.js
views > new.ejs > html > body > div.container > form
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Add New Product</title>
7   <link rel="stylesheet" href="/styles.css"> <!-- Include your CSS file -->
8 </head>
9 <body>
10   <div class="container">
11     <h1>Add New Product</h1>
12     <form action="/products" method="post">
13       <div class="form-group">
14         <label form="name">Product Name:</label>
15         <input type="text" id="name" name="name" required>
16       </div>
17       <div class="form-group">
18         <label form="quantity">Quantity:</label>
19         <input type="number" id="quantity" name="quantity" required>
20       </div>
21       <div class="form-group">
22         <label form="price">Price:</label>
23         <input type="number" id="price" name="price" required>
24       </div>
25       <div class="form-group">
26         <label form="image">Image URL:</label>
27         <input type="text" id="image" name="image">
28       </div>
29       <button type="submit" class="btn">Add Product</button>
30     </form>
31   </div>
32 </body>
33 </html>
34
```

```
27         <input type="text" id="image" name="image">
28       </div>
29       <button type="submit" class="btn">Add Product</button>
30     </form>
31   </div>
32 </body>
33 </html>
34
```

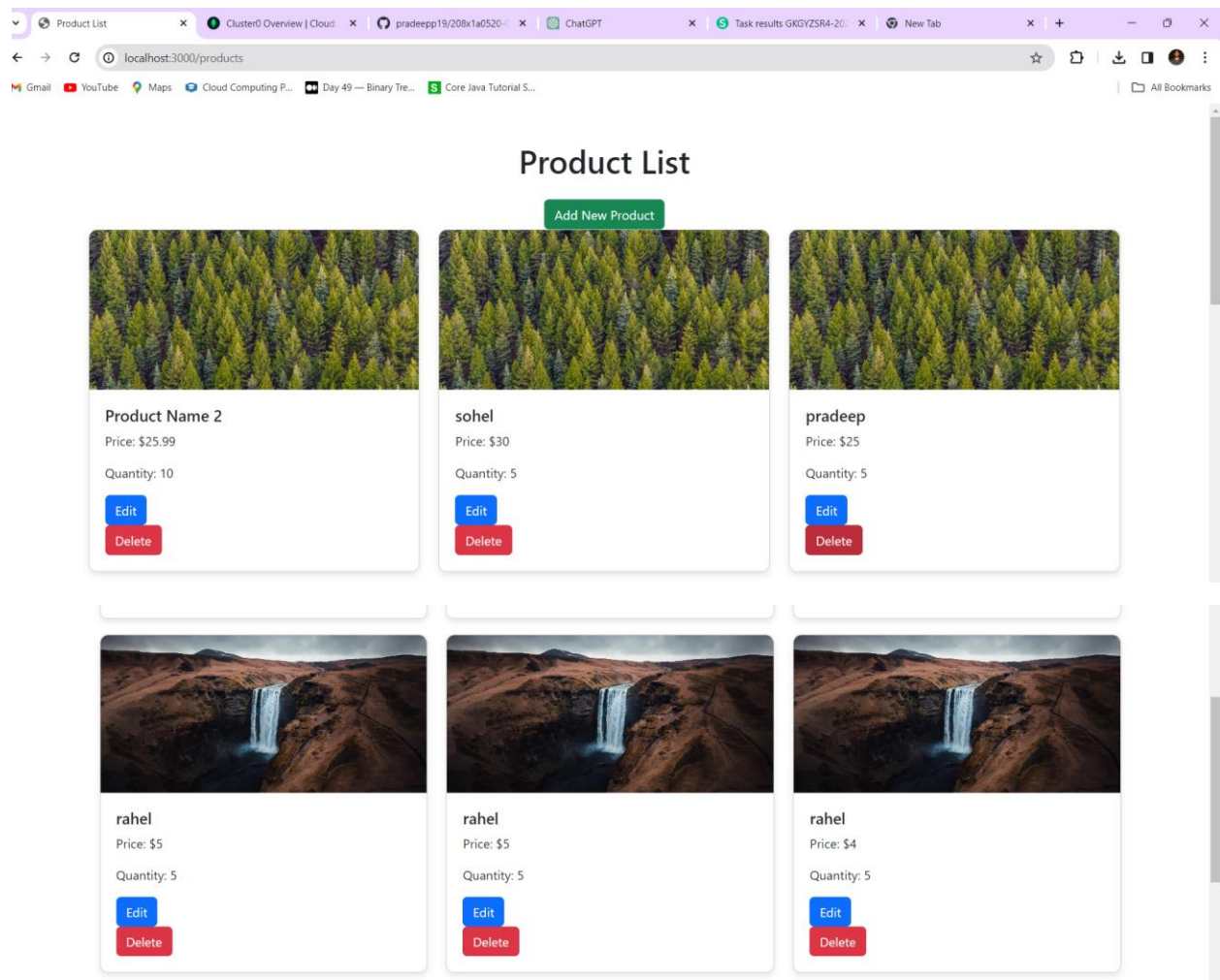
Styles.css:

```
File Edit Selection View Go Run ... Node-API
EXPLORER
  > NODE-API
  > Assignment3
  > models
  JS productModel.js
  > node_modules
  > styles
  # styles.css
  > views
  .gitignore
  ASSIGNMENT-1.pdf
  ASSIGNMENT-2.pdf
  package-lock.json
  package.json
  README.md
  JS server.js

styles > # styles.css > .add-btn: hover
1 /* General styles */
2 body {
3   font-family: Arial, sans-serif;
4   background-color: #f9f9f9;
5   margin: 0;
6   padding: 0;
7 }
8
9 .container {
10   max-width: 800px;
11   margin: 0 auto;
12   padding: 20px;
13 }
14
15 h1 {
16   text-align: center;
17   margin-bottom: 20px;
18 }
19
20 /* Form styles */
21 form {
22   margin-top: 20px;
23 }
24
25 label {
26   display: block;
27   margin-bottom: 5px;
28 }
```

Outputs:

```
PS C:\Users\sohel\Node-API> node server.js
(node:7632) [MONGODB] DeprecationWarning: Mongoose: the 'strictQuery' option will be switched back to 'false' by default in Mongoose 7. Use 'mongoose.set('strictQuery', false);' if you want to prepare for this change. Or use 'mongoose.set('strictQuery', true);' to suppress this warning.
(Use 'node --trace-deprecation ...' to show where the warning was created)
Server is running on port 3000
Connected to MongoDB
[]
```



After clicking add product:

Add New Product

Product Name:

Quantity:

Price:

Image URL:

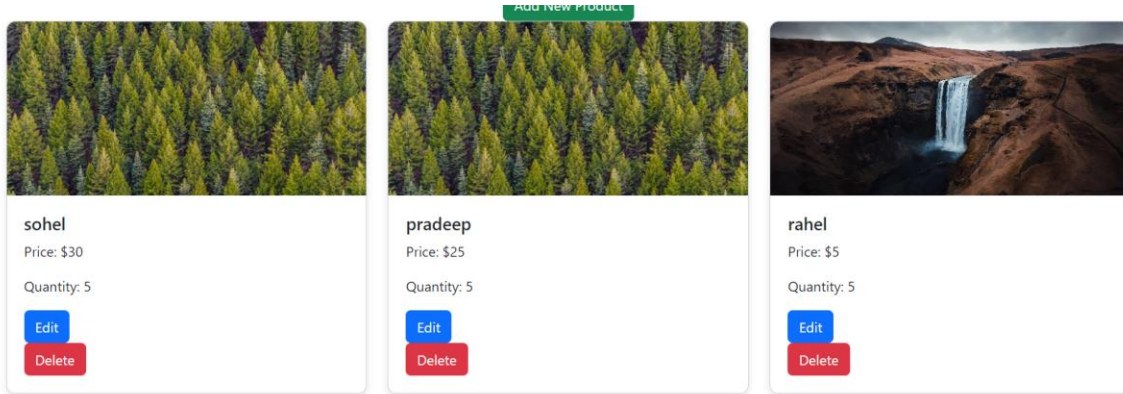
By clicking edit of any photo:

Edit Product

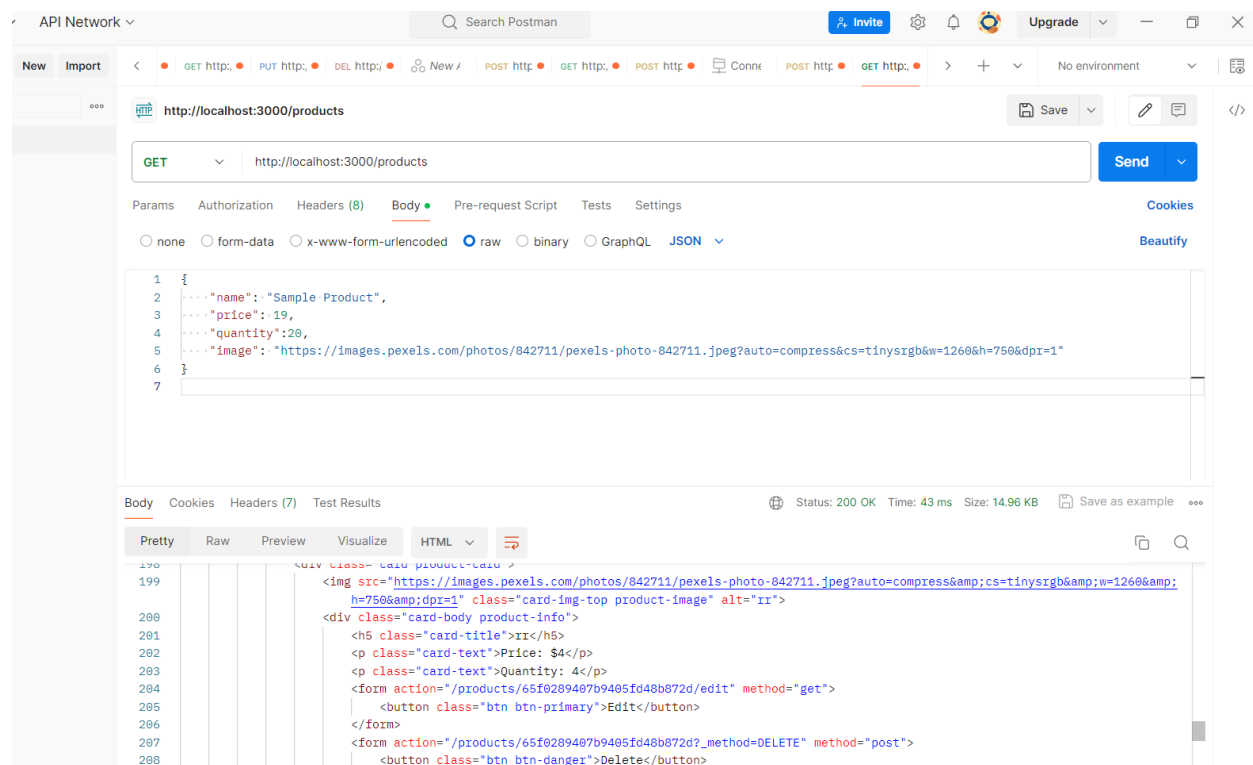
Product Name Quantity Price Image URL

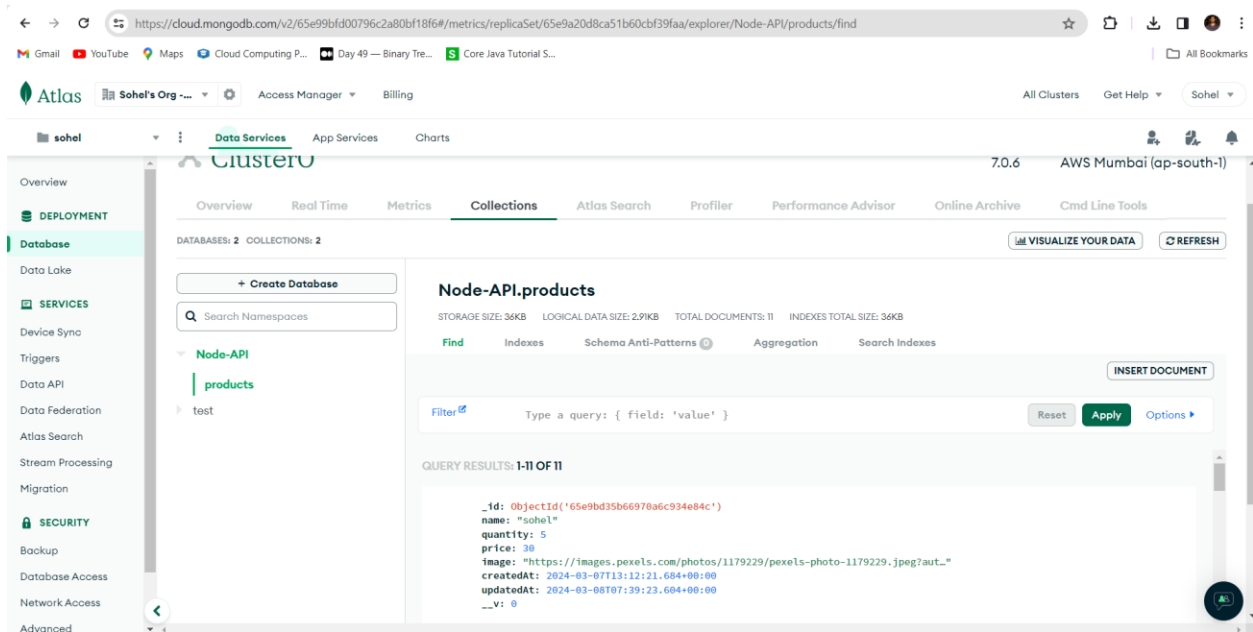
By clicking delete product info will be deleted :

I have deleted product name 2



We have used post man api





How to run :

First connect mongo db by

'mongodb+srv://pradeep:6O6tJYeS0vKUeyvg@cluster0.qlorrai.mongodb.net/Node-API?retryWrites=true&w=majority'

Run the local host by : node server.js