

PRACTICAL-7

AIM : Working with Express Framework

1) Build a simple express js program to add, update, delete and display all products available in list of e-commerce application using get, put, post and delete methods.

Assume that products are available on products object. Product properties are product_id, product_name, product_size, product_brand, product_color.

Code:

```
console.log('21012011074')
var p = require('body-parser');
var express = require('express');
var app = express();
app.use(p.json());
var products = [
  {
    "product_id": "p1",
    "product_name": "smartphone",
    "product_size": "15.7",
    "product_brand": "lenovo",
    "product_color": "red"
  },
  {
    "product_id": "p2",
    "product_name": "Laptop",
    "product_size": "16.2",
    "product_brand": "Asus",
    "product_color": "black"
  },
  {
    "product_id": "p3",
    "product_name": "Mouse",
    "product_size": "10",
    "product_brand": "Hp",
    "product_color": "black"
  }
];

app.get('/products', (req, res) => {
  res.send(products);
});

app.post('/products', (req, res) => {
  const newProduct = req.body;
  products.push(newProduct);
  res.send('Product added successfully.');
```

```
});

// Update a product by product_id
app.put('/products/:product_id', (req, res) => {
  const product_id = req.params.product_id;
```

```

const updatedProduct = req.body;
const productIndex = products.findIndex(product => product.product_id === product_id);
if (productIndex !== -1) {
  products[productIndex] = updatedProduct;
  res.send('Product updated successfully.');
```

```

}
else {
  res.status(404).send('Product not found.');
```

```

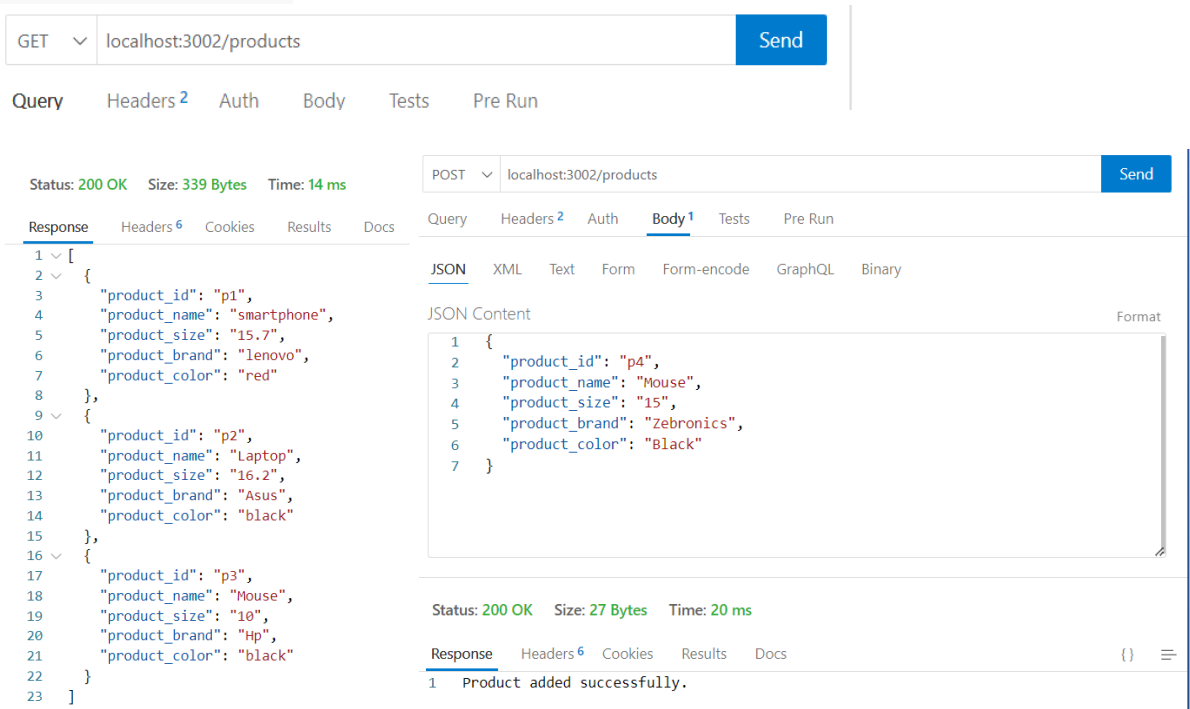
});

app.delete('/products/:product_id', (req, res) => {
  const product_id = req.params.product_id;
  const productIndex = products.findIndex(product => product.product_id === product_id);
  if (productIndex !== -1) {
    products.splice(productIndex, 1);
    res.send('Product deleted successfully.');
```

Output:

```

PS E:\Every Study Materials\SEMwise
21012011074
Server is running on port 3002
```



The screenshot displays a REST client interface with two requests. The first request is a GET to localhost:3002/products, which returns a 200 OK status with a JSON array of three products. The second request is a POST to localhost:3002/products with a JSON body containing product details (product_id: "p4", product_name: "Mouse", product_size: "15", product_brand: "Zebronics", product_color: "Black"). This request also returns a 200 OK status, and the response body shows "Product added successfully."

Request 1: GET localhost:3002/products

Status: 200 OK Size: 339 Bytes Time: 14 ms

Response:

```
[
  {
    "product_id": "p1",
    "product_name": "smartphone",
    "product_size": "15.7",
    "product_brand": "lenovo",
    "product_color": "red"
  },
  {
    "product_id": "p2",
    "product_name": "Laptop",
    "product_size": "16.2",
    "product_brand": "Asus",
    "product_color": "black"
  },
  {
    "product_id": "p3",
    "product_name": "Mouse",
    "product_size": "10",
    "product_brand": "Hp",
    "product_color": "black"
  }
]
```

Request 2: POST localhost:3002/products

Status: 200 OK Size: 27 Bytes Time: 20 ms

Body:

```
{
  "product_id": "p4",
  "product_name": "Mouse",
  "product_size": "15",
  "product_brand": "Zebronics",
  "product_color": "Black"
}
```

Response:

```
1 Product added successfully.
```

PUT localhost:3002/products/p1 **Send**

Query Headers 2 Auth **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```

1 {
2   "product_id": "p1",
3   "product_name": "Smartphone",
4   "product_size": "15.7",
5   "product_brand": "Samsung",
6   "product_color": "Black"
7 }

```

Status: 200 OK Size: 29 Bytes Time: 6 ms

Response Headers 6 Cookies Results Docs

1 Product updated successfully.

DELETE localhost:3002/products/p3 **Send**

Status: 200 OK Size: 29 Bytes Time: 6 ms

Response Headers 6 Cookies Results Docs

1 Product deleted successfully.

Status: 200 OK Size: 349 Bytes Time: 4 ms

Response Headers 6 Cookies Results Docs

```

1 [
2   {
3     "product_id": "p1",
4     "product_name": "Smartphone",
5     "product_size": "15.7",
6     "product_brand": "Samsung",
7     "product_color": "Black"
8   },
9   {
10    "product_id": "p2",
11    "product_name": "Laptop",
12    "product_size": "16.2",
13    "product_brand": "Asus",
14    "product_color": "black"
15  },
16  {
17    "product_id": "p4",
18    "product_name": "Mouse",
19    "product_size": "15",
20    "product_brand": "Zebronics",
21    "product_color": "Black"
22  }
23 ]

```

2) Create express js program using router for e-commerce application. This application has 4 modules: order, user, product and category. Each module has 4 methods, namely get, put, post and delete and has path '/', '/update-details', 'create-details', and 'deletedetails respectively. For example: 'localhost:8000/product' will be accessed and the page '/' will displayed message "get method from user module", same for other 3 methods and modules.

Code:

pr7_2_user.js

```

console.log('21012011074')
var express=require('express');
var r=express.Router();
r.get('/display',(req,res)=>{
    res.send('display user records')
})

```

```
r.post('/add',(req,res)=>{
    res.send('Add request for user')
})
r.put('/edit',(req,res)=>{
    res.send('edit request for user')
})
r.delete('/delete',(req,res)=>{
    res.send('delete request for user')
})
module.exports=r;
```

pr7_2_category.js

```
var express=require('express');
var r=express.Router();
r.get('/display',(req,res)=>{
    res.send('display category records')
})
r.post('/add',(req,res)=>{
    res.send('Add request for category')
})
r.put('/edit',(req,res)=>{
    res.send('edit request for category')
})
r.delete('/delete',(req,res)=>{
    res.send('delete request for category')
})
module.exports=r;
```

pr7_2_product.js

```
var express=require('express');
var r=express.Router();
r.get('/display',(req,res)=>{
    res.send('display product records')
})
r.post('/add',(req,res)=>{
    res.send('Add request for product')
})
r.put('/edit',(req,res)=>{
    res.send('edit request for product')
})
r.delete('/delete',(req,res)=>{
    res.send('delete request for product')
})
module.exports=r;
```

pr7_2_order.js

```
var express=require('express');
var r=express.Router();
r.get('/display',(req,res)=>{
    res.send('display order records')
```

```
    })
    r.post('/add',(req,res)=>{
        res.send('Add request for order')
    })
    r.put('/edit',(req,res)=>{
        res.send('edit request for order')
    })
    r.delete('/delete',(req,res)=>{
        res.send('delete request for order')
    })
    module.exports=r;
```

pr7_2_server.js

```
var express=require('express');
var u=require('./p_2_user');
var c=require('./p_2_category');
var p=require('./p_2_product');
var o=require('./p_2_order');
var app=express();
app.use('/p_2_user',u);
app.use('/p_2_category',c);
app.use('/p_2_product',p);
app.use('/p_2_order',o);
console.log("8000 port is running");
app.listen(8000);
```

Output:

```
PS E:\Every Study Materials\SEMwise Materials\SE
"
21012011074
21012011074
8000 port is running
```

GET localhost:8000/pr7_2_order/display Send

Status: 200 OK Size: 21 Bytes Time: 14 ms

Response Headers 6 Cookies Results Docs {} ≡

1 display order records

PUT localhost:8000/pr7_2_product/edit Send

Status: 200 OK Size: 24 Bytes Time: 7 ms

Response Headers 6 Cookies Results Docs {} ≡

1 edit request for product

DELETE	localhost:8000/pr7_2_order/delete	Send
--------	-----------------------------------	------

Status: 200 OK Size: 24 Bytes Time: 6 ms

Response Headers 6 Cookies Results Docs {} ≡

1 delete request for order

POST	localhost:8000/pr7_2_category/add	Send
------	-----------------------------------	------

Status: 200 OK Size: 24 Bytes Time: 4 ms

Response Headers 6 Cookies Results Docs {} ≡

1 Add request for category

3) Create express js program using router for e-commerce application. This application has 4 modules: order, user, product and category. Each module has 4 methods, namely get, put, post and delete and has path '/', '/update-details', 'create-details', and 'deletedetails respectively. For example: 'localhost:8000/product' will be accessed and the page '/' will displayed message "get method from user module", same for other 3 methods and modules.

Code:

```
console.log('21012011074')
const express = require('express');
const app = express();

// Middleware for timestamp
app.use((req, res, next) => {
  req.timestamp = new Date().toLocaleString();
  next();
});

// Middleware for the homepage
app.use('/home', (req, res, next) => {
  req.page = 'homepage';
  next();
});

// Middleware for /user/admin
app.use('/user/admin', (req, res, next) => {
  req.msg = 'Going to the login page';
  next();
});

// Middleware for /user/:username
app.use('/user/:username', (req, res, next) => {
```

```
    if (req.params.username !== 'admin') {
      req.msg = 'You are not admin';
    }
    next();
  });

// Route for /
app.get('/', (req, res) => {
  res.send(`Timestamp: ${req.timestamp}`);
});

// Route for /home
app.get('/home', (req, res) => {
  res.send(`Timestamp: ${req.timestamp} - Page: ${req.page}`);
});

// Route for /user/admin
app.get('/user/admin', (req, res) => {
  res.send(req.msg);
});

// Route for /user/:username
app.get('/user/:username', (req, res) => {
  res.send(req.msg);
});

app.listen(3100, () => {
  console.log(`Server is running on http://localhost:3100`);
});
```

Output:

```
PS E:\Every Study Materials\SEMwise Materials\
21012011074

Server is running on http://localhost:3100
█
```

GET

localhost:3100

Send

Status: 200 OK Size: 33 Bytes Time: 7 ms

Response

Headers⁶

Cookies

Results

Docs

{}

≡

1 Timestamp: 24/10/2023, 4:39:12 pm

GET	localhost:3100/home	Send
-----	---------------------	------

Status: 200 OK Size: 50 Bytes Time: 4 ms

Response Headers⁶ Cookies Results Docs {} ≡

1 Timestamp: 24/10/2023, 4:40:53 pm - Page: homepage

GET	localhost:3100/user/admin	Send
-----	---------------------------	------

Status: 200 OK Size: 23 Bytes Time: 5 ms

Response Headers⁶ Cookies Results Docs {} ≡

1 Going to the login page

GET	localhost:3100/user/practical7	Send
-----	--------------------------------	------

Status: 200 OK Size: 17 Bytes Time: 4 ms

Response Headers⁶ Cookies Results Docs {} ≡

1 You are not admin