

GANPAT UNIVERSITY
U.V. PATEL COLLEGE OF ENGINEERING

B.Tech 5th Semester CE/IT

2CEIT5PE4: Software Packages

Practical: 7

Working with Express Framework

Que: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.

```
var express = require('express');
var bap = require('body-parser');
var app = express();

var products=[
  {
    product_id:"123456789",
    product_name:"HP Pavilion Gaming Laptop 15",
    product_size:"15inch",
    product_brand:"HP",
    product_color:"Infrared"
  }
];

app.use(bap.urlencoded({extended:false}));
app.use(bap.json());

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

app.post("/Add_Product/", (req, res)=>{
  var sdata = req.body;
  products.push(sdata);
  res.send(products);
});

app.put("/Edit_Product/:product_id", (req, res)=>{
  var old_product_id = req.params.product_id;
  var newdata = req.body;
```

```

    var index = products.findIndex((element) =>
    element.product_id === old_product_id);
    if(index !== -1)
    {
        products[index].product_id=newdata.product_id;
        products[index].product_name=newdata.product_name;
        products[index].product_size=newdata.product_size;
        products[index].product_brand=newdata.product_brand;
        products[index].product_color=newdata.product_color;
    }
    res.send(products);
});

app.delete("/Delete_Product/:product_id", (req,res)=>{
    var old_product_id = req.params.product_id;
    products = products.filter(item => item.product_id !== old_product_id);
    res.send(products);
});

app.listen(3000);

```

Output:

```
PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q1.js"
```

GET http://localhost:3000/ Status: 200 OK Size: 146 Bytes Time: 21 ms

Response:

```

1 [
2   {
3     "product_id": "123456789",
4     "product_name": "HP Pavilion Gaming Laptop 15",
5     "product_size": "15inch",
6     "product_brand": "HP",
7     "product_color": "Infrared"
8   }
9 ]

```

POST http://localhost:3000/Add_Product/ Status: 200 OK Size: 276 Bytes Time: 32 ms

Response:

```

1 [
2   {
3     "product_id": "123456789",
4     "product_name": "HP Pavilion Gaming Laptop 15",
5     "product_size": "15inch",
6     "product_brand": "HP",
7     "product_color": "Infrared"
8   },
9   {
10    "product_id": "12450154412",
11    "product_name": "Sandisk Pendrive",
12    "product_size": "64GB",
13    "product_brand": "WD",
14    "product_color": "Black"
15  }
16 ]

```

PUT http://localhost:3000/Edit_Product/123456789

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

Json Xml Text Form Form-encode GraphQL Binary

Json Content Format

```
1 {
2   "product_id": "931134979311",
3   "product_name": "SanDisk Dual Drive Go 128 GB OTG Drive",
4   "product_size": "128GB",
5   "product_brand": "WD(Sandisk)",
6   "product_color": "Blue"
7 }
```

Status: 200 OK Size: 293 Bytes Time: 7 ms

Response Headers ⁶ Cookies Results Docs

```
1 [
2   {
3     "product_id": "931134979311",
4     "product_name": "SanDisk Dual Drive Go 128 GB OTG Drive",
5     "product_size": "128GB",
6     "product_brand": "WD(Sandisk)",
7     "product_color": "Blue"
8   },
9   {
10    "product_id": "12450154412",
11    "product_name": "Sandisk Pendrive",
12    "product_size": "64GB",
13    "product_brand": "WD",
14    "product_color": "Black"
15  }
16 ]
```

DELETE http://localhost:3000/Delete_Product/12450154412

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

Json Xml Text Form Form-encode GraphQL Binary

Json Content Format

```
1
```

Status: 200 OK Size: 163 Bytes Time: 5 ms

Response Headers ⁶ Cookies Results Docs

```
1 [
2   {
3     "product_id": "931134979311",
4     "product_name": "SanDisk Dual Drive Go 128 GB OTG Drive",
5     "product_size": "128GB",
6     "product_brand": "WD(Sandisk)",
7     "product_color": "Blue"
8   }
9 ]
```

Que: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 ‘delete-details’ 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.

q2.js

```
var express = require('express');
var user = require('./user.js');
var order = require('./order.js');
var product = require('./product.js');
var category = require('./category.js');
var app = express();

app.use("/user",user.router1);
app.use("/order",order.router1);
app.use("/product",product.router1);
app.use("/category",category.router1);

app.listen(3000);
```

user.js

```
var express = require('express');
var router1 = express.Router();
router1.get('/', function (req, res, next) {
    res.send("This is get method of user");
});
router1.post('/create-details', function (req, res, next) {
    res.send("This is post method of user");
});
router1.put('/update-details', function (req, res, next) {
    res.send("This is put method of user");
});
router1.delete('/delete-details', function (req, res, next) {
    res.send("This is delete method of user");
});
exports.router1=router1
```

order.js

```
var express = require('express');
var router1 = express.Router();
router1.get('/', function (req, res, next) {
    res.send("This is get method of order");
});
router1.post('/create-details', function (req, res, next) {
    res.send("This is post method of order");
});
router1.put('/update-details', function (req, res, next) {
    res.send("This is put method of order");
});
router1.delete('/delete-details', function (req, res, next) {
    res.send("This is delete method of order");
});
exports.router1=router1
```

product.js

```
var express = require('express');
var router1 = express.Router();
router1.get('/', function (req, res, next) {
    res.send("This is get method of product");
});
router1.post('/create-details', function (req, res, next) {
    res.send("This is post method of product");
});
router1.put('/update-details', function (req, res, next) {
    res.send("This is put method of product");
});
router1.delete('/delete-details', function (req, res, next) {
    res.send("This is delete method of product");
});
exports.router1=router1;
```

category.js

```
var express = require('express');
var router1 = express.Router();
router1.get('/', function (req, res, next) {
    res.send("This is get method of category");
});
router1.post('/create-details', function (req, res, next) {
    res.send("This is post method of category");
});
router1.put('/update-details', function (req, res, next) {
    res.send("This is put method of category");
});
router1.delete('/delete-details', function (req, res, next) {
    res.send("This is delete method of category");
});
exports.router1=router1;
```

Output:

```
PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q2.js"
```

GET	http://localhost:3000/user/	Send	Status: 200 OK	Size: 26 Bytes	Time: 20 ms
Query	Headers ²	Auth	Body	Tests	Pre Run ^{New}
Json	Xml	Text	Form	Form-encode	Graphql
Json Content	Format				
1	Response				
1 This is get method of user					

POST	http://localhost:3000/order/create-details/	Send	Status: 200 OK	Size: 28 Bytes	Time: 5 ms
Query	Headers ²	Auth	Body	Tests	Pre Run ^{New}
Json	Xml	Text	Form	Form-encode	Graphql
Json Content	Format				
1	Response				
1 This is post method of order					

PUT	http://localhost:3000/product/update-details/	Send	Status: 200 OK	Size: 29 Bytes	Time: 5 ms
Query	Headers ²	Auth	Body	Tests	Pre Run ^{New}
Json	Xml	Text	Form	Form-encode	Graphql
Json Content	Format				
1	Response				
1 This is put method of product					

DELETE	http://localhost:3000/category/delete-details/	Send	Status: 200 OK	Size: 33 Bytes	Time: 5 ms
Query	Headers ²	Auth	Body	Tests	Pre Run ^{New}
Json	Xml	Text	Form	Form-encode	Graphql
Json Content	Format				
1	Response				
1 This is delete method of category					

Que:3 Create express js program that employ middleware using next() function.

```
const express = require('express');
const app = express();
app.use(Employ);
app.get('/', (req, res) => {
  res.send("This is Home Page");
  console.log('This is Home Page');
});
function Employ (req, res, next) {
  console.log('This is Employ Middleware');
  next();
}
app.listen(3000);
```

Output:

```
PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q3.js"
```

```
PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q3.js"
This is Employ Middleware
This is Home Page
```

The screenshot shows a web client interface. At the top, a GET request is made to `http://localhost:3000/`. The status is `200 OK`, the size is `17 Bytes`, and the time is `23 ms`. The response body is `This is Home Page`. The interface includes tabs for Query, Headers, Auth, Body, Tests, and Pre Run. The Body tab is selected, showing the response content in JSON format.

Que:4 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. Using monodb database. Assume that products are available on 'products' collection in 'ecom' database. Product properties are product_id product_name product_size product_brand product_color.

```
var express = require('express');
var bap = require('body-parser');
var app = express();
const MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/"
const dbname = "ecom";

app.use(bap.urlencoded({extended:false}));
app.use(bap.json());

app.get("/", (req, res)=>{
  MongoClient.connect(url, (err1, db)=>{
    if(err1){
      throw err1;
    }
    else{
      var database = db.db(dbname);
      console.log("Database Successfully Connected!");
      database.collection('products').find({})
      .toArray(function(err2, data_result){
        if(err2)
        {
          throw err2;
        }
        else{
          res.send(data_result);
          db.close();
        }
      })
    }
  });
});

app.post("/Add_Product/", (req, res1)=>{
  var sdata = req.body;
  MongoClient.connect(url, (err1, db)=>{
    if(err1){
      throw err1;
    }
    else{
      var database = db.db(dbname);
      console.log("Database Successfully Connected!");
      database.collection('products')
      .insertMany(sdata, function(err2, res2){
        if(err2)
        {
          throw err2;
        }
      })
    }
  })
});
```



```

        else{
            console.log(res2.insertedCount+" Data inserted");
            database.collection('products').find({})
            .toArray(function(err3,res3){
                if(err3)
                {
                    throw err3;
                }
                else{
                    res1.send(res3);
                    db.close();
                }
            });
        }
    });
}
});
});

app.put("/Edit_Product/:product_id", (req,res1)=>{
    var old_product_id = {"product_id":req.params.product_id};
    var data = req.body;
    var newdata = {$set:data};
    MongoClient.connect(url, (err1, db)=>{
        if(err1){
            throw err1;
        }
        else{
            var database = db.db(dbname);
            console.log("Database Successfully Connected!");
            database.collection('products')
            .updateOne(old_product_id,newdata,function(err2,res2){
                if(err2)
                {
                    throw err2;
                }
                else{
                    console.log(res2.modifiedCount+" Data Updated");
                    database.collection('products').find({})
                    .toArray(function(err3,res3){
                        if(err3)
                        {
                            throw err3;
                        }
                        else{
                            res1.send(res3);
                            db.close();
                        }
                    });
                }
            });
        }
    });
});
});
});
});

```

```

app.delete("/Delete_Product/:product_id", (req, res1) => {
    var old_product_id = {"product_id": req.params.product_id};
    MongoClient.connect(url, (err1, db) => {
        if (err1) {
            throw err1;
        }
        else {
            var database = db.db(dbname);
            console.log("Database Successfully Connected!");
            database.collection('products')
                .deleteOne(old_product_id, function (err2, res2) {
                    if (err2) {
                        {
                            throw err2;
                        }
                    }
                    else {
                        console.log(res2.deletedCount + " Data Deleted");
                        database.collection('products').find({})
                            .toArray(function (err3, res3) {
                                if (err3) {
                                    {
                                        throw err3;
                                    }
                                }
                                else {
                                    res1.send(res3);
                                    db.close();
                                }
                            }
                        );
                    }
                });
        }
    });
});
app.listen(3000);

```

Output:

```

PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q4.js"

```

GET ⌵ http://localhost:3000/ Send		Status: 200 OK Size: 179 Bytes Time: 114 ms	
Query	Headers ²	Auth	Body
Tests Pre Run ^{New}			
Json Xml Text Form Form-encode GraphQL Binary		Response Headers ⁶ Cookies Results Docs	
Json Content Format		1 2 3 4 5 6 7 8 9 10	
1		<pre> [{ "_id": "63707fdc3d7ddd290b75d0b7", "product_id": "123456789", "product_name": "HP Pavilion Gaming Laptop 15", "product_size": "15inch", "product_brand": "HP", "product_color": "Infrared" }] </pre>	

POST http://localhost:3000/Add_Product/ Send

Status: 200 OK Size: 588 Bytes Time: 74 ms

Query Headers Auth **Body** Tests Pre Run New

Response Headers Cookies Results Docs

Json Content Format

```

1 [
2   {
3     "product_id": "9223566547811",
4     "product_name": "Simmtronics ZIPX 128 GB
      USB 3.0 Pen Drive 128 GB Pen Drive",
5     "product_size": "128GB",
6     "product_brand": "Simmtronics",
7     "product_color": "Gold"
8   },
9   {
10    "product_id": "65435313211",
11    "product_name": "HP V220W 64 GB Pen
      Drive 64 GB Pen Drive",
12    "product_size": "64GB",
13    "product_brand": "HP",
14    "product_color": "Grey, Black"
15  }
16 ]

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE JUPYTER TERMINAL

```

PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q4.js"
Database Successfully Connected!
Database Successfully Connected!
2 Data inserted

```

PUT http://localhost:3000/Edit_Product/123456789 Send

Status: 200 OK Size: 622 Bytes Time: 32 ms

Query Headers Auth **Body** Tests Pre Run New

Response Headers Cookies Results Docs

Json Content Format

```

1 {
2   "product_id": "1234",
3   "product_name": "KBR PRODUCT SUPER SPECIAL
      MULTI FUNCTION BALL POINT PEN 32 GB Pen
      Drive",
4   "product_size": "32GB",
5   "product_brand": "KBR",
6   "product_color": "Black"
7 }

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE JUPYTER TERMINAL

```

PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q4.js"
Database Successfully Connected!
Database Successfully Connected!
2 Data inserted
Database Successfully Connected!
1 Data Updated

```

The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/Delete_Product/9223566547811`. The response is a JSON array of two product objects. The terminal at the bottom shows the execution of a Node.js script that interacts with a database.

Request:

- Method: DELETE
- URL: `http://localhost:3000/Delete_Product/9223566547811`
- Body: (Empty)

Response:

```
[
  {
    "_id": "63707fdc3d7ddd290b75d0b7",
    "product_id": "1234",
    "product_name": "KBR PRODUCT SUPER SPECIAL MULTI FUNCTION BALL POINT PEN 32 GB Pen Drive",
    "product_size": "32GB",
    "product_brand": "KBR",
    "product_color": "Black"
  },
  {
    "_id": "63708102564d62cda6b9eb55",
    "product_id": "65435313211",
    "product_name": "HP V220W 64 GB Pen Drive 64 GB Pen Drive",
    "product_size": "64GB",
    "product_brand": "HP",
    "product_color": "Grey, Black"
  }
]
```

Terminal Output:

```
PS E:\B_Tech\SEM_5\SP\Practical\Code\Practical 7> node "e:\B_Tech\SEM_5\SP\Practical\Code\Practical 7\q4.js"
Database Successfully Connected!
Database Successfully Connected!
2 Data inserted
Database Successfully Connected!
1 Data Updated
Database Successfully Connected!
1 Data Deleted
[]
```