



**FPT POLYTECHNIC**



**android**

[www.poly.edu.vn](http://www.poly.edu.vn)

## LẬP TRÌNH ANDROID VỚI RESTAPI

API DELETE, GỬI MAIL, AUTHENTICATION

- ☐ Tạo API DELETE
- ☐ Upload ảnh lên server
- ☐ Gửi mail bằng API
- ☐ Sử dụng authentication – Json Web Token (JWT)

# MỤC TIÊU

- ◎ TẠO API DELETE
- ◎ UPLOAD HÌNH ẢNH LÊN SERVER
- ◎ GỬI MAIL BẰNG API
- ◎ SỬ DỤNG AUTHENTICATION — JSON WEB TOKEN (JWT)





# VIẾT API DELETE

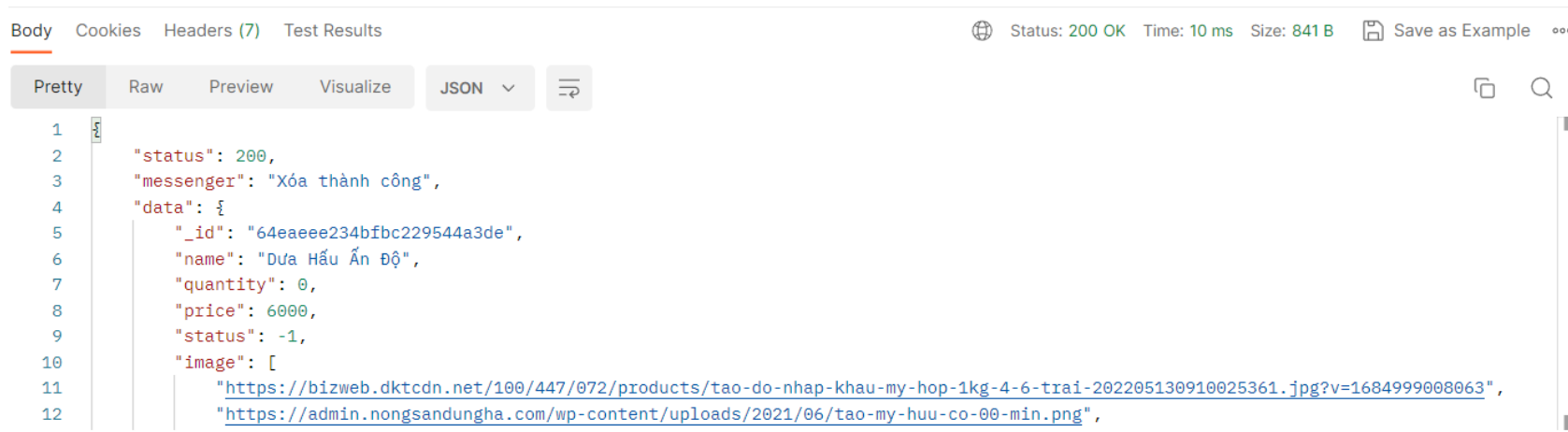
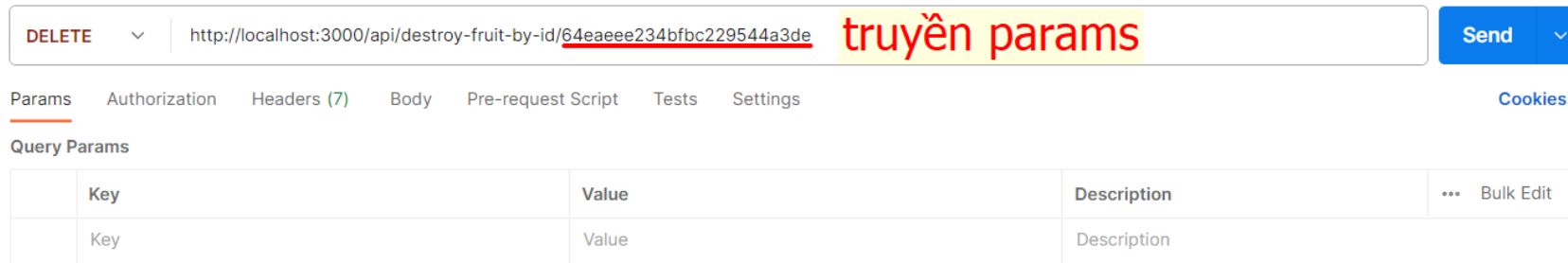
---

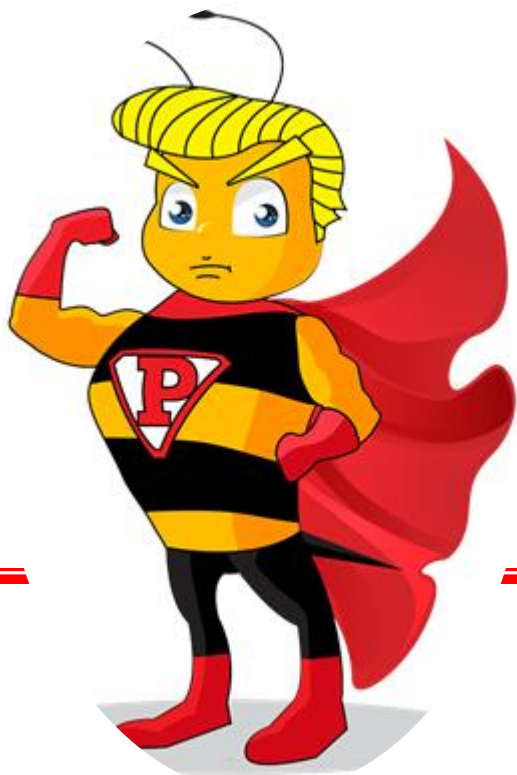
...

- ❑ Mở file **api.js** trong thư mục routes, tiếp tục tạo api để xóa 1 collection thông qua **\_id**

```
router.delete(path: '/destroy-fruit-by-id/:id',...handlers: async (req,res)=>{
  try {
    const {id} = req.params
    const result = await Fruits.findByIdAndDelete(id);
    if(result)
    {
      //Nếu xóa thành công sẽ trả về thông item đã xóa
      res.json(body: {
        "status" : 200,
        "messenger" : "Xóa thành công",
        "data" : result
      })
    }else
    {
      res.json(body: {
        "status" : 400 ,
        "messenger" : "Lỗi, Xóa không thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(message: error);
  }
})
```

## ❑ Mở Postman và sử dụng thử api DELETE





---

# UPLOAD HÌNH ẢNH LÊN SERVER

---

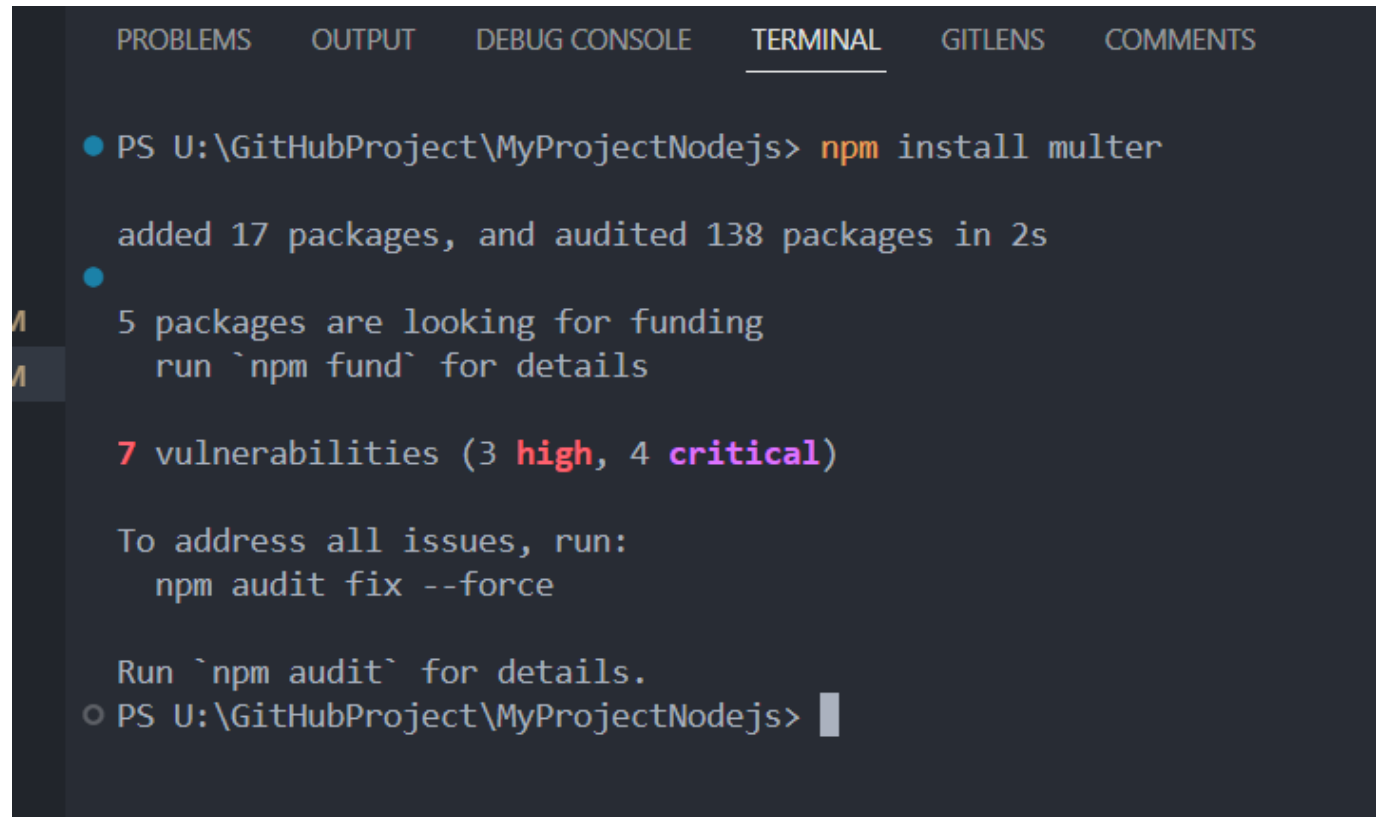
...

- ❑ **Multer** là một middleware cho Express và Nodejs giúp dễ dàng xử lý dữ liệu multipart/form-data khi người dùng upload file





- ❑ Mở terminal VS Code và sử dụng câu lệnh: **npm i multer**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  COMMENTS

• PS U:\GitHubProject\MyProjectNodejs> npm install multer

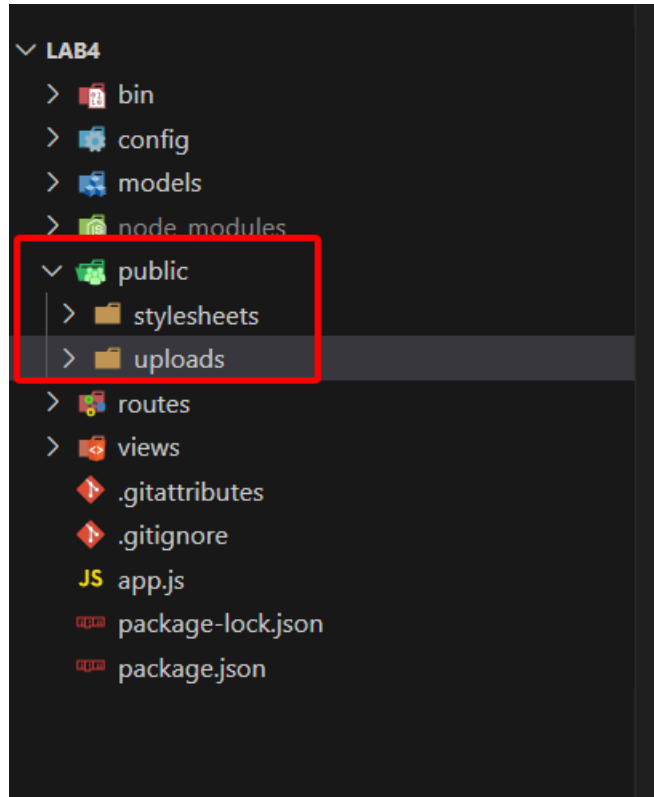
added 17 packages, and audited 138 packages in 2s
•
5 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (3 high, 4 critical)

To address all issues, run:
  npm audit fix --force

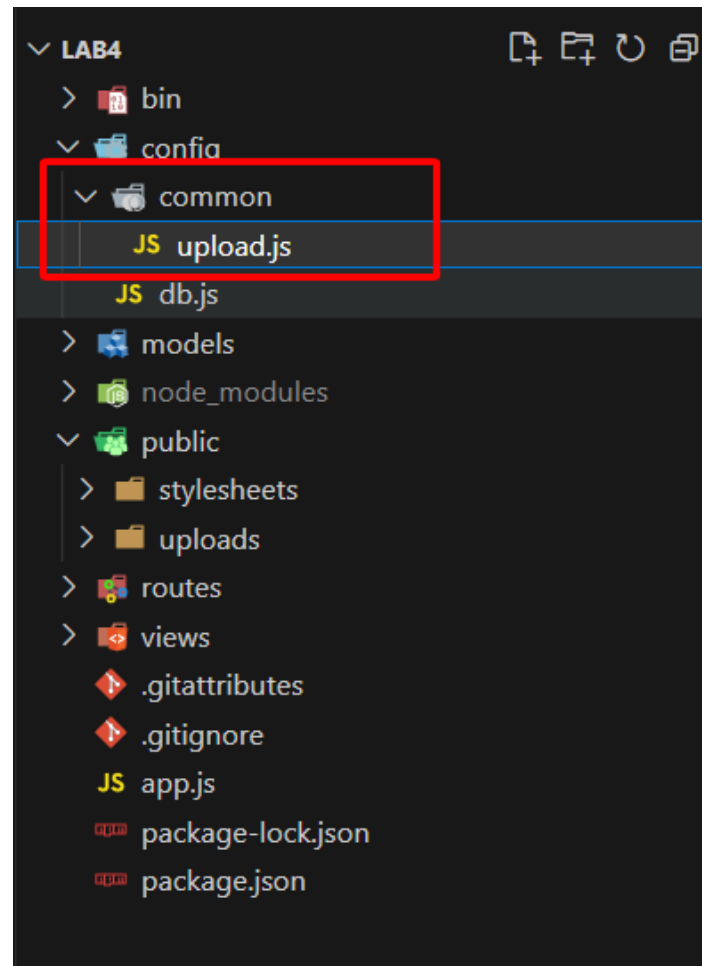
Run `npm audit` for details.
○ PS U:\GitHubProject\MyProjectNodejs> 
```

□ Tiếp theo, tạo folder **uploads** trong folder **public**



*Lưu ý: hình ảnh upload sẽ chỉ thực hiện được trong folder public hoặc các folder trong folder public*

- ❑ Trong folder **config**, tạo folder **common**, trong folder **common** tạo file **upload.js**



# VIẾT CODE XỬ LÝ HÌNH ẢNH ĐƯỢC UPLOAD

JS upload.js M X

config &gt; common &gt; JS upload.js &gt; ...

```
1  const multer = require(id: "multer");
2  /// const _storage = multer.diskStorage(options: {
3      destination: (req, file, cb) => {
4          cb(error: null, destination: "public/uploads");
5      },
6      filename: (req, file, cb) => {
7          cb(error: null, filename: file.fieldname + "-" + Date.now() + file.originalname);
8      },
9  });
10 /// const upload = multer(options: { storage: _storage });|
11
12 module.exports = upload
```

# VIẾT API POST CÓ HÌNH ẢNH ĐƯỢC UPLOAD

- ❑ Mở file api.js import file upload.js và tạo api thêm mới 1 collection có hình ảnh được upload

```
const Upload = require(id: '../config/common/upload');
```

```
router.post(path: '/add-fruit-with-file-image',...handlers: Upload.array(fieldName: 'image',maxCount: 5),async (req,res) => {  
  //Upload.array('image',5) => up nhiều file tối đa là 5  
  //upload.single('image') => up load 1 file  
  try {  
    const data = req.body; // Lấy dữ liệu từ body  
    const {files} = req // lấy files nếu upload nhiều, file nếu 1  
    const urlsImage = files.map((file)=>`${req.protocol}://${req.get(name: "host")}/uploads/${file.filename}`)  
    const newfruit = new Fruits(doc: {  
      name: data.name,  
      quantity : data.quantity,  
      price : data.price,  
      status : data.status,  
      image : urlsImage, /* Thêm cả url hình */  
      description : data.description,  
      id_distributor : data.id_distributor  
    }); //Tạo một đối tượng mới  
    const result = await newfruit.save(); //Thêm vào database  
    if(result){  
      // Nếu thêm thành công result !null trả về dữ liệu  
      res.json(body: {  
        "status" : 200,  
        "messenger" : "Thêm thành công",  
        "data" : result })  
    }else {  
      // Nếu thêm không thành công result null, thông báo không thành công  
      res.json(body: {  
        "status" : 400 ,  
        "messenger" : "Lỗi, thêm không thành công",  
        "data" : [] })  
    }  
  } catch (error) {  
    console.log(message: error);  
  });  
});
```

POST http://localhost:3000/api/add-fruit-with-file-image

Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none **form-data** x-www-form-urlencoded raw binary GraphQL

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Táo đỏ			
<input checked="" type="checkbox"/>	quantity	200			
<input checked="" type="checkbox"/>	price	8000			
<input checked="" type="checkbox"/>	status	1			
<input checked="" type="checkbox"/>	image	File 2 files selected			

This file isn't in your working directory. Teammates you share this request with

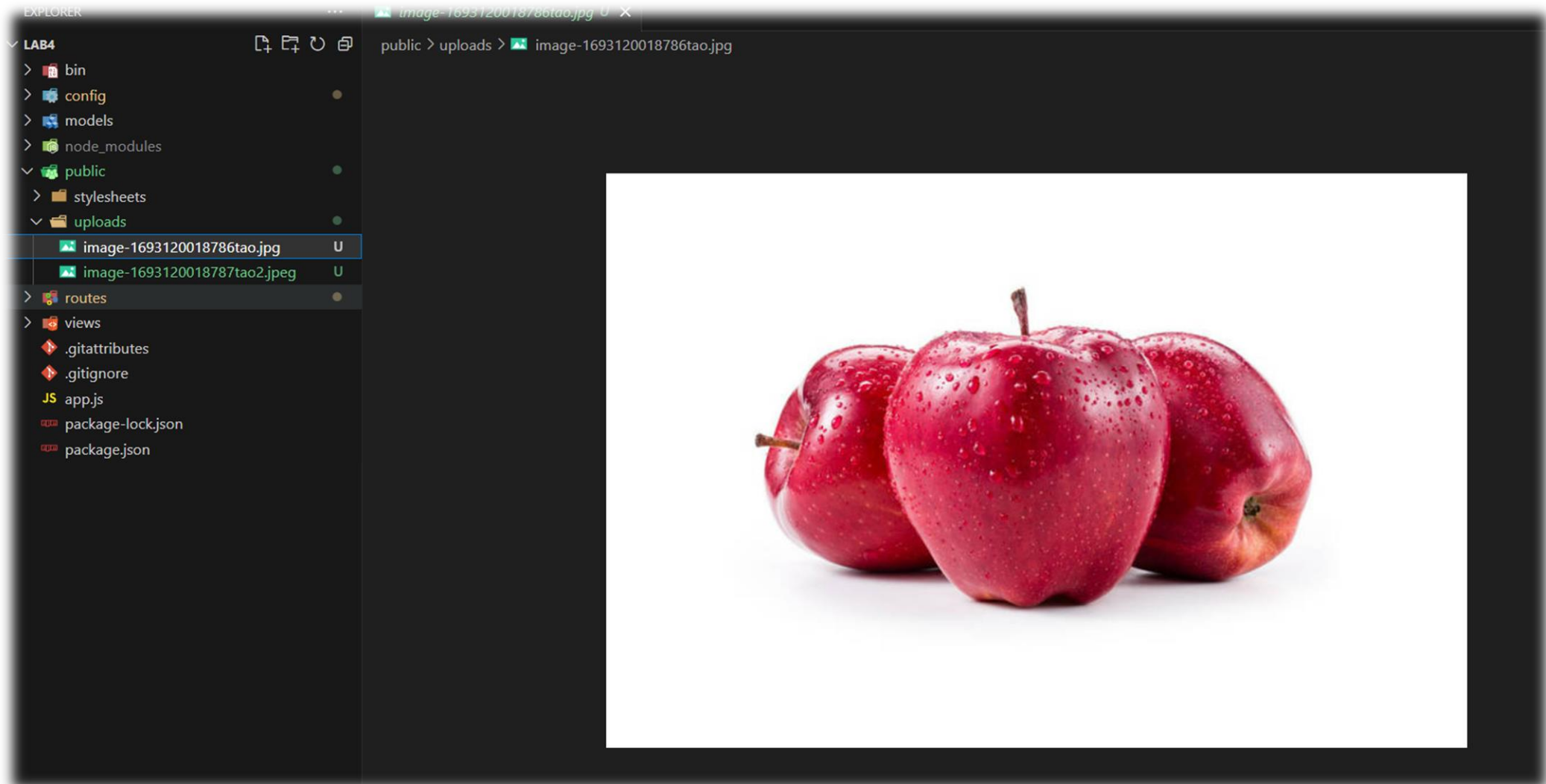
Body Cookies Headers (7) Test Results

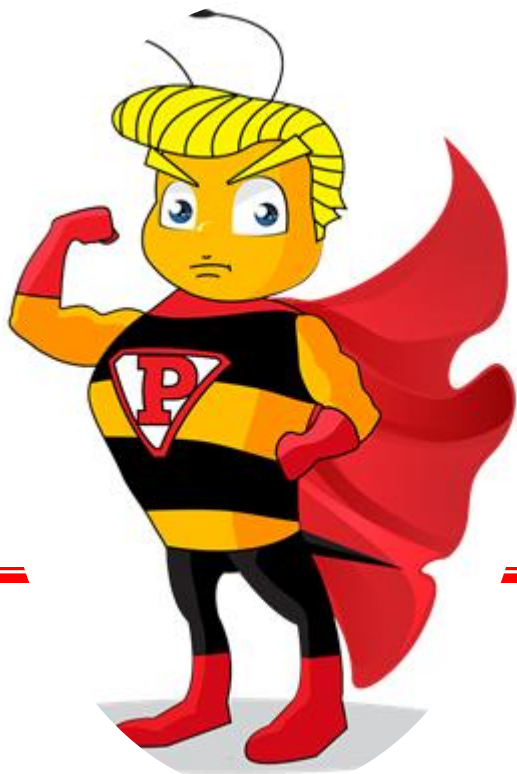
Status: 200 OK Time: 11 ms Size: 674 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": 200,
3   "messenger": "Thêm thành công",
4   "data": {
5     "name": "Táo đỏ ",
6     "quantity": 200,
7     "price": 8000,
8     "status": 1,
9     "image": [
10      "http://localhost:3000/uploads/image-1693120018786tao.jpg",
11      "http://localhost:3000/uploads/image-1693120018787tao2.jpeg"
12    ],
13     "description": "Không ngon",
```

## KIỂM TRA HÌNH ẢNH ĐƯỢC UPLOAD





---

# GỬI MAIL BẰNG API

---

...



❑ **Nodemailer** là 1 module cho ứng dụng Node.js cho phép gửi mail 1 cách dễ dàng.

Nodemailer là 1 dự án bắt đầu từ năm 2010 khi mà chưa có sự phát triển của các dịch vụ gửi mail và ngày nay đây là giải pháp mà hầu hết người dùng sử dụng cho ứng dụng Nodejs để gửi mail.



- ❑ Mở terminal VS Code, sử dụng câu lệnh: **npm i nodemailer**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS D:\MyProjectNodejs> npm install nodemailer

added 18 packages, and audited 139 packages in 4s

5 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (3 high, 4 critical)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\MyProjectNodejs> 
```

❑ Trong thư mục **common** tạo file **mail.js**

```
JS mail.js M X
config > common > JS mail.js > [🔗] transporter > ⚙ auth
1  var nodemailer = require(id: "nodemailer");
2  const transporter = nodemailer.createTransport(transport: {
3    service: "gmail",
4    auth: {
5      user: "mailtoest@gmail.com", // Email gửi đi
6      pass: "123456789", // Mật khẩu email gửi
7    },
8  });
9
10 module.exports = transporter;
```

❑ Trong file api.js import file mail.js và tạo API đăng ký user kết hợp gửi mail

```
router.post(path: '/register-send-email',...handlers: Upload.single(fieldName: 'avatar'),async(req,res) =>{
  try {
    const data = req.body;
    const {file} = req
    const newUser = Users({
      username: data.username,
      password : data.password,
      email: data.email,
      name: data.name,
      avatar: `${req.protocol}://${req.get(name: "host")}/uploads/${file.filename}`,
      //url avatar http://localhost:3000/uploads/filename
    })
    const result = await newUser.save()
    if(result) {
      //Gửi mail
      const mailOptions = {
        from: "mailtest@gmail.com", //email gửi đi
        to: result.email, // email nhận
        subject: "Đăng ký thành công", //subject
        text: "Cảm ơn bạn đã đăng ký", // nội dung mail
      };
      // Nếu thêm thành công result !null trả về dữ liệu
      await Transporter.sendMail(mailOptions); // gửi mail
      res.json(body: {
        "status" : 200,
        "messenger" : "Thêm thành công",
        "data" : result })
    } else {
      // Nếu thêm không thành công result null, thông báo không thành công
      res.json(body: {
        "status" : 400 ,
        "messenger" : "Lỗi, thêm không thành công",
        "data" : [] })
    }
  } catch (error) {
    console.log(message: error);
  }
})
```

POST

http://localhost:3000/api/register-send-email

Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

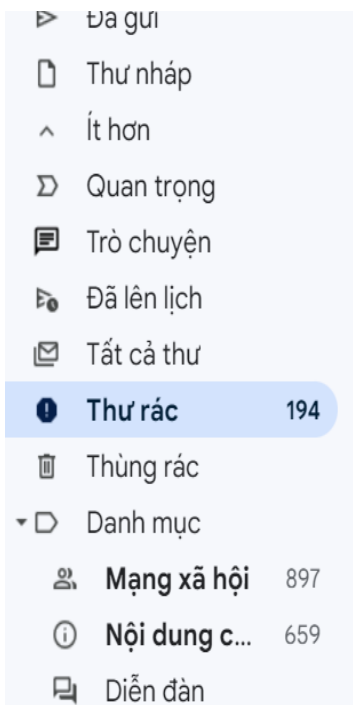
	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	polytechnic			
<input checked="" type="checkbox"/>	password	123			
<input checked="" type="checkbox"/>	email				
<input checked="" type="checkbox"/>	name	polytechnic			
<input checked="" type="checkbox"/>	avatar	tao.jpg			
	Key	Value	Description		

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 2.48 s Size: 593 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": 200,
3   "messenger": "Thêm thành công",
4   "data": {
5     "username": "polytechnic",
6     "password": "123",
7     "email": "",
8     "name": "polytechnic",
9     "avatar": "http://localhost:3000/uploads/avatar-1693125543778tao.jpg",
10    "available": false,
11    "_id": "64eb0ba7709374e8e6c92306",
12    "createdAt": "2023-08-27T08:39:03.780Z",
```



 @gmail.com

15:47 (0 phút trước) ☆ ↩ ⋮

Tại sao thư này lại ở trong thư mục spam? Thư tương tự như các thư đã bị xác định là spam trong quá khứ.

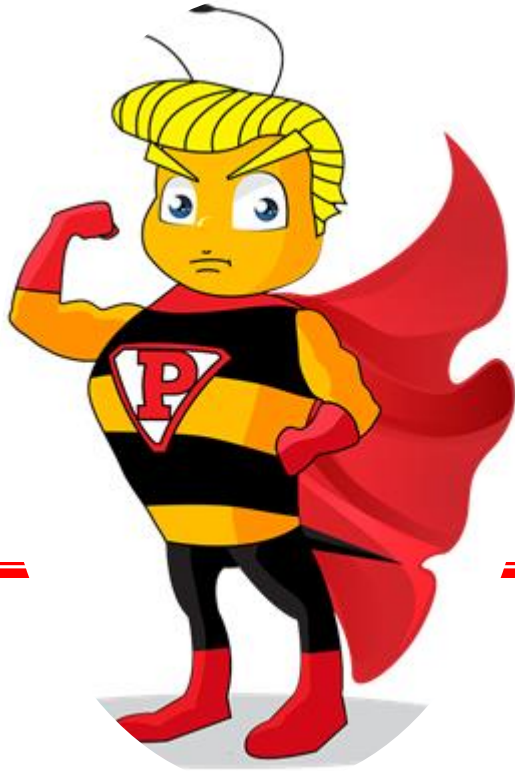
Báo cáo không phải là spam

Cảm ơn bạn đã đăng ký

← Trả lời

↪ Chuyển tiếp

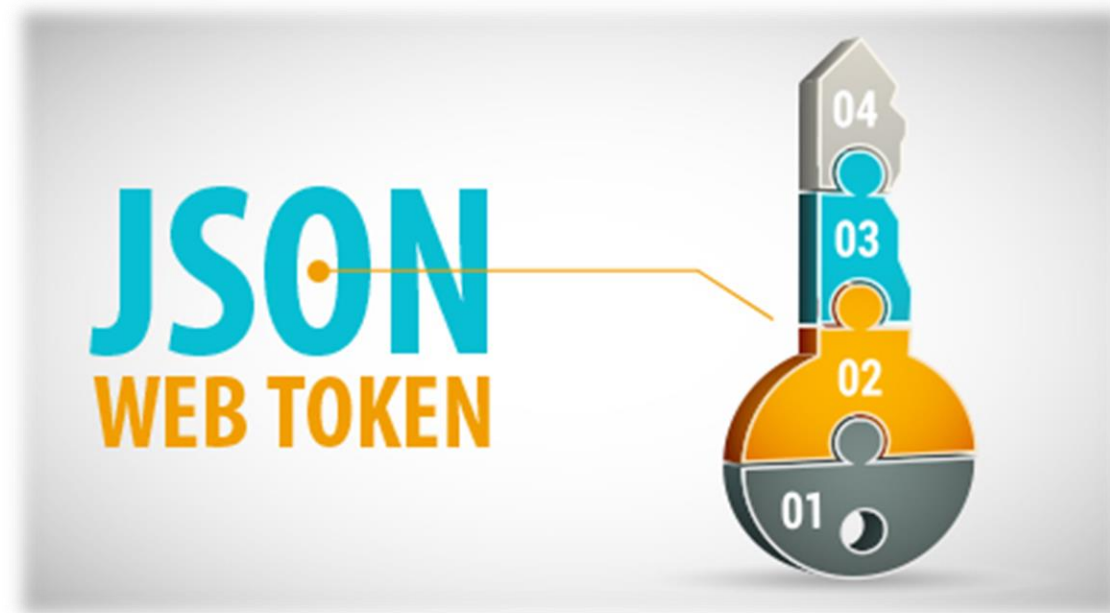
Bật thông báo trên màn hình cho Gmail. OK Không, cảm ơn X



# AUTHENTICATION - JWT

---

- ❑ JWT (JSON Web Token) là một tiêu chuẩn mở được sử dụng để chia sẻ thông tin bảo mật giữa client và server. Mỗi JWT đều chứa các đối tượng JSON được mã hoá và một bộ quyền sở hữu (claim). JWT được ký bằng thuật toán cryptographic để đảm bảo các quyền sở hữu không thể bị thay đổi sau khi phát hành token.





- ❑ Mở terminal VS Code, gõ lệnh: **npm i jsonwebtoken**
- ❑ Import jwt, tạo SECRETKEY trong file **api.js** và viết api login

```
const JWT = require('jsonwebtoken');
const SECRETKEY = "FPTPOLYTECHNIC"
router.post(path: '/login',...handlers: async (req,res)=>{
  try {
    const {username,password} = req.body;
    const user = await Users.findOne(filter: {username,password})
    if(user)
    {
      //Token người dùng sẽ sử dụng gửi lên trên header mỗi lần muốn gọi api
      const token = JWT.sign(payload: {id: user._id},secretOrPrivateKey: SECRETKEY,options: {expiresIn: '1h'});
      //Khi token hết hạn, người dùng sẽ call 1 api khác để lấy token mới
      //Lúc này người dùng sẽ truyền refreshToken lên để nhận về 1 cặp token,refreshToken mới
      //Nếu cả 2 token đều hết hạn người dùng sẽ phải thoát app và đăng nhập lại
      const refreshToken = JWT.sign(payload: {id: user._id},secretOrPrivateKey: SECRETKEY,options: {expiresIn: '1d'})
      //expiresIn thời gian token
      res.json(body: {
        "status" : 200,
        "messenger" : "Đăng nhập thành công",
        "data" : user,
        "token" : token,
        "refreshToken" : refreshToken
      })
    }
  }else
  {
    // Nếu thêm không thành công result null, thông báo không thành công
    res.json(body: {
      "status" : 400 ,
      "messenger" : "Lỗi, đăng nhập không thành công",
      "data" : []
    })
  }
} catch (error) {
  console.log(message: error);
}
```

## Kiểm tra API vừa tạo

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/api/login
- Body:** JSON format with the following content:

```
1 {
2   "username": "polytechnic",
3   "password": "123"
4 }
```
- Response:** Status 200 OK, Time: 33 ms, Size: 972 B. The response body is shown in JSON format:

```
6 {
7   "password": "123",
8   "email": "quanvp2021@gmail.com",
9   "name": "polytechnic",
10  "avatar": "http://localhost:3000/uploads/avatar-1693124561381api.webp",
11  "available": false,
12  "createdAt": "2023-08-27T08:22:41.390Z",
13  "updatedAt": "2023-08-27T08:22:41.390Z",
14  "__v": 0
15 },
16 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0ZWlwn2QxODFmZDkxYzQ5ZTYwYmM2NSIsIm1hdCI6MTY5MzE5NDY0OCwiZXhwIjoxNjkzMTI4MjQ4fQ.M1
17 "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0ZWlwn2QxODFmZDkxYzQ5ZTYwYmM2NSIsIm1hdCI6MTY5MzE5NDY0OCwiZXhwIjoxNjkzMTI4MjQ4fQ.M2
18 }
```

## □ Kiểm tra token mỗi lần gọi API

```
router.get(path: '/get-list-fruit',...handlers: async (req,res,next) => {
  const authHeader = req.headers['authorization']
  //Authorization thêm từ khóa `Bearer token`
  //nên sẽ xử lý cắt chuỗi
  const token = authHeader && authHeader.split(separator: ' ')[1]
  //Nếu không có token sẽ trả về 401
  if (token == null) return res.sendStatus(code: 401)
  let payload;
  JWT.verify(token, secretOrPublicKey: SECRETKEY , callback: (err,_payload) => {
    //Kiểm tra token, nếu token ko đúng, hoặc hết hạn
    //Trả status code 403
    //Trả status hết hạn 401 khi token hết hạn
    if(err instanceof JWT.TokenExpiredError) return res.sendStatus(code: 401)
    if (err) return res.sendStatus(code: 403)
    //Nếu đúng sẽ log ra dữ liệu
    payload = _payload;
  })
  console.log(message: payload);
  try {
    const data = await Fruits.find().populate(path: 'id_distributor');
    res.json(body: {
      "status" : 200,
      "messenger" : "Danh sách fruit",
      "data" : data
    })
  } catch (error) {
    console.log(message: error);
  }
})
```

GET

http://localhost:3000/api/get-list-fruit

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

Status: 401 Unauthorized

Time: 9 ms

Size: 250 B

Save as Example

...

Pretty

Raw

Preview

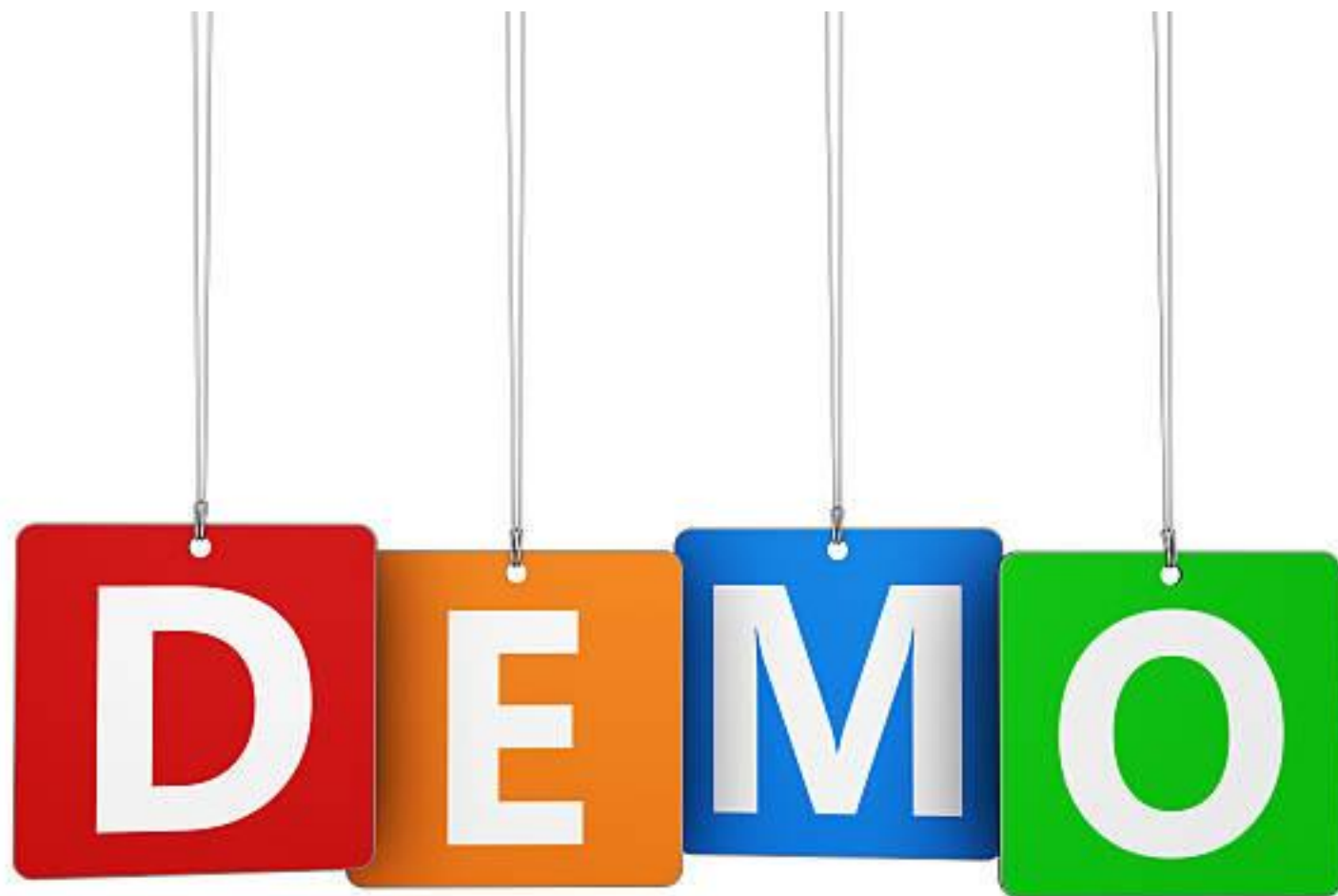
Visualize

Text

1

Unauthorized







**FPT** Education

**FPT POLYTECHNIC**

**Thank you**