

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



**BÁO CÁO THỰC HÀNH BACKEND (1+2)
PHÁT TRIỂN ỨNG DỤNG WEB
CT449 - 01**

Giảng viên hướng dẫn:
ThS. Nguyễn Minh Trung

Sinh viên thực hiện:
Tên: Trần Trương Ngọc Uyên
MSSV: B2207576
Khóa 48

Cần Thơ, năm 2025

MỤC LỤC

MỤC LỤC	i
DANH MỤC HÌNH	ii
Ứng dụng Contactbook - Backend - Phần 1	1
Bước 0: Cài đặt môi trường Node và git	1
Bước 1: Tạo ứng dụng Node	1
Bước 2: Quản lý mã nguồn dự án với git và github	2
Bước 3: Cấu hình Visual code, ESLint và Prettier	5
Bước 4: Cài đặt Express	8
Bước 5: Định nghĩa controller và các router	11
Bước 6: Cài đặt xử lý lỗi	15
ĐƯỜNG LINK GITHUB:	17
Ứng dụng Contactbook - Backend - Phần 2	18
Bước 0: Cài đặt MongoDB	18
Bước 1: Cài đặt thư viện MongoDB, định nghĩa hàm trợ giúp kết nối và lớp dịch vụ truy xuất cơ sở dữ liệu (CSDL)	18
Bước 2: Cài đặt các handler	21
ĐƯỜNG LINK GITHUB:	33

DANH MỤC HÌNH

Hình 1: Kiểm tra môi trường	1
Hình 2: Khởi động ứng dụng Node thành công	1
Hình 3: Thông tin để tạo tập tin package.json	2
Hình 4: Tải package cần thiết	2
Hình 5: Tạo tập tin “.gitignore”	2
Hình 6: Khởi tạo dự án Git	2
Hình 7: Kiểm tra với “git status”	3
Hình 8: git quản lý 3 dự án .gitignore, package-lock.json, package.json	3
Hình 9: Kiểm tra với “git status”	3
Hình 10: Đã tạo thành công	4
Hình 11: Upload dự án lên GitHub	4
Hình 12: Chứng thực tài khoản và kiểm tra trên GitHub	4
Hình 13: Cài đặt ESLint extension thành công	5
Hình 14: Cài đặt Prettier extension thành công	5
Hình 15: Cấu hình VSCode cho phép định dạng Save & Paste	5
Hình 16: Cài đặt gói thư viện ESLint và Prettier	6
Hình 17: Tạo một tập tin eslintrc.js	6
Hình 18: Lưu thay đổi trên git	7
Hình 19: Tập đã tạo thành công trên GitHub	7
Hình 20: Tạo tập tin app.js	8
Hình 21: Tạo tập tin server.js	8
Hình 22: Tạo tập tin index.js	8
Hình 23: Kết quả chạy server thành công	9
Hình 24: Kiểm tra kết quả bằng ứng dụng Posman	9
Hình 25: Cài đặt nodemon thành công	9
Hình 26: Thay đổi cấu hình mục “scripts”	10
Hình 27: Lưu thay đổi trên vào git	10
Hình 28: Các tập đã chạy thành công	10
Hình 29: Được tạo thành công	10
Hình 30: Tập đã tải lên GitHub thành công	11
Hình 31: Tạo tập tin contact.controller.js	11
Hình 32: Tạo tập tin contact.route.js	12

Hình 33: Cập nhật lại app.js	12
Hình 34: Kết quả chạy thành công	13
Hình 35: Kết quả chạy thành công	13
Hình 36: Lưu thay đổi trên vào git	14
Hình 37: Các tệp đã chạy thành công	14
Hình 38: Được tạo thành công	14
Hình 39: Tệp đã tải lên GitHub thành công	14
Hình 40: Tạo tập tin api-error.js	15
Hình 41: Thêm các middleware xử lý lỗi trong app.js	15
Hình 42: Kiểm tra kết quả bằng ứng dụng Posman	16
Hình 43: Lưu thay đổi trên vào git	16
Hình 44: Các tệp đã chạy thành công	16
Hình 45: Được tạo thành công	16
Hình 46: Tệp đã tải lên GitHub thành công	17
Hình 47: Cây thư mục hiện tại	17
Hình 48: Cài đặt MongoDB	18
Hình 49: Cài đặt thư viện mongodb	18
Hình 50: Hiệu chỉnh thêm đoạn code trong tập tin index.js	19
Hình 51: Tạo tập tin mongodb.utils.js	19
Hình 52: Thay đổi nội dung tập tin server.js	20
Hình 53: Định nghĩa các lớp dịch vụ ContactServices	20
Hình 54: Cài đặt handler create	21
Hình 55: Tạo phương thức create()	21
Hình 56: Hiệu chỉnh Header "Content-Type: application/json"	22
Hình 57: Đặt dữ liệu JSON trong phần Body	22
Hình 58: Kết quả MongoDB	23
Hình 59: Cài đặt handler findAll	23
Hình 60: Tạo 2 phương thức find(), findByName()	23
Hình 61: Kết quả chạy thành công	24
Hình 62: Cài đặt handler findOne	24
Hình 63: Tạo phương thức findById()	25
Hình 64: Kết quả chạy thành công	25
Hình 65: Cài đặt handler update	25

Hình 66: Tạo phương thức update()	26
Hình 67: Kết quả chạy thành công	26
Hình 68: Trước khi update bên MongoDB	27
Hình 69: Sau khi update bên MongoDB	27
Hình 70: Cài đặt handler delete	27
Hình 71: Tạo phương thức create()	28
Hình 72: Kết quả chạy thành công	28
Hình 73: Kết quả được xóa thành công	28
Hình 74: Cài đặt handler findAllFavorite	29
Hình 75: Tạo phương thức findFavorite()	29
Hình 76: Kết quả chạy thành công cho ra rỗng	29
Hình 77: Kết quả hiện thị thành công các thông tin	30
Hình 78: Cài đặt handler deleteAll	30
Hình 79: Tạo phương thức deleteAll()	31
Hình 80: Kết quả được xóa thành công	31
Hình 81: Lưu thay đổi trên vào git	31
Hình 82: Các tệp đã chạy thành công	32
Hình 83: Được tạo thành công	32
Hình 84: Tệp đã tải lên GitHub thành công	32
Hình 85: Cây thư mục hiện tại	33

Ứng dụng Contactbook - Backend - Phần 1

Bước 0: Cài đặt môi trường Node và git

- Thực hiện tải nodejs và git theo hướng dẫn.
- Sau đó kiểm tra bằng lệnh như ảnh:

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1> node -v  
v22.14.0  
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1> git --version  
git version 2.45.1.windows.1  
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1> 
```

Hình 1: Kiểm tra môi trường

Bước 1: Tạo ứng dụng Node

- Tạo thư mục dự án có tên “ ContactBook-backend “
- Trong thư mục này, ta khởi tạo ứng dụng Node bằng lệnh:

```
npm init
```

Chú thích: `npm` sẽ hỏi về một số thông tin để tạo tập tin package.json cho dự án

```
○ PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
  
Press ^C at any time to quit.  
package name: (contactbook-backend)
```

Hình 2: Khởi động ứng dụng Node thành công

```

version: (1.0.0)
description:
entry point: (index.js) server.js
test command:
git repository:
keywords:
author: Uyen Tran
license: (ISC)
About to write to D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend\package.json:

{
  "name": "contactbook-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Uyen Tran",
  "license": "ISC",
  "description": ""
}

```

Hình 3: Thông tin để tạo tập tin package.json

- Cài đặt các package cần thiết

npm install express cors

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npm install express cors
added 69 packages, and audited 70 packages in 2s

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

found 0 vulnerabilities

```

Hình 4: Tải package cần thiết

Bước 2: Quản lý mã nguồn dự án với git và github

- Tạo tập tin “.gitignore” bằng lệnh **npx gitignore node**

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npx gitignore node
Need to install the following packages:
gitignore@0.7.0
Ok to proceed? (y) y

Created .gitignore file for flag type node.

```

Hình 5: Tạo tập tin “.gitignore”

- Gõ lệnh: “**git init**” để khởi tạo dự án Git.

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git init
Initialized empty Git repository in D:/CT449_PTUD_WEB/THUC_HANH/BackEnd/BackEnd_Part1/ContactBook-back
end/.git/

```

Hình 6: Khởi tạo dự án Git

- Sau đó kiểm tra với “ **git status** ”

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    package-lock.json
    package.json

nothing added to commit but untracked files present (use "git add" to track)
```

Hình 7: Kiểm tra với “ **git status** ”

- ⇒ Ta thấy có 3 tệp tin CHƯA ĐƯỢC quản lý bởi git là .gitignore, package-lock.json, package.json
- Cho git quản lý 3 dự án .gitignore, package-lock.json, package.json bằng lệnh:
git add .gitignore package-lock.json package.json

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add .gitignore package-lock.json package.json
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
```

Hình 8: git quản lý 3 dự án .gitignore, package-lock.json, package.json

- Dùng lệnh **git status** để kiểm tra lại lần nữa

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   package-lock.json
    new file:   package.json
```

Hình 9: Kiểm tra với “ **git status** ”

- ⇒ Ta thấy 3 tệp tin ĐÃ ĐƯỢC quản lý bởi git là .gitignore, package-lock.json, package.json

- Thực thi lệnh: `git commit -m "Cài đặt dự án"`

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git commit -m "Cai dat du an"
[master (root-commit) a48dc13] Cai dat du an
 3 files changed, 1002 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 package-lock.json
 create mode 100644 package.json
```

Hình 10: Đã tạo thành công

- Upload dự án lên GitHub với các lệnh:
 - `git remote add origin https://github.com/4MM4-052/B2207576_Tran_Truong_Ngoc_Uyen_BACKEND_1.git`
 - `git push origin main`

```
⑧ PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git remote add origin https://github.com/4MM4-052/B2207576_Tran
 _Truong_Ngoc_Uyen_BACKEND_1.git
error: remote origin already exists.

❶ PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git push origin master
Everything up-to-date
```

Hình 11: Upload dự án lên GitHub

- Thực hiện chứng thực tài khoản và kiểm tra lại các tập tin đã được tải lên GitHub

File	Content	Last Modified
.gitignore	Cai dat du an	25 minutes ago
package-lock.json	Cai dat du an	25 minutes ago
package.json	Cai dat du an	25 minutes ago

Hình 12: Chứng thực tài khoản và kiểm tra trên GitHub

Bước 3: Cấu hình Visual code, ESLint và Prettier

- Cài đặt ESLint extension. ESLint là một công cụ phân tích tĩnh mã Javascript, giúp tìm và sửa các vấn đề trong mã nguồn.



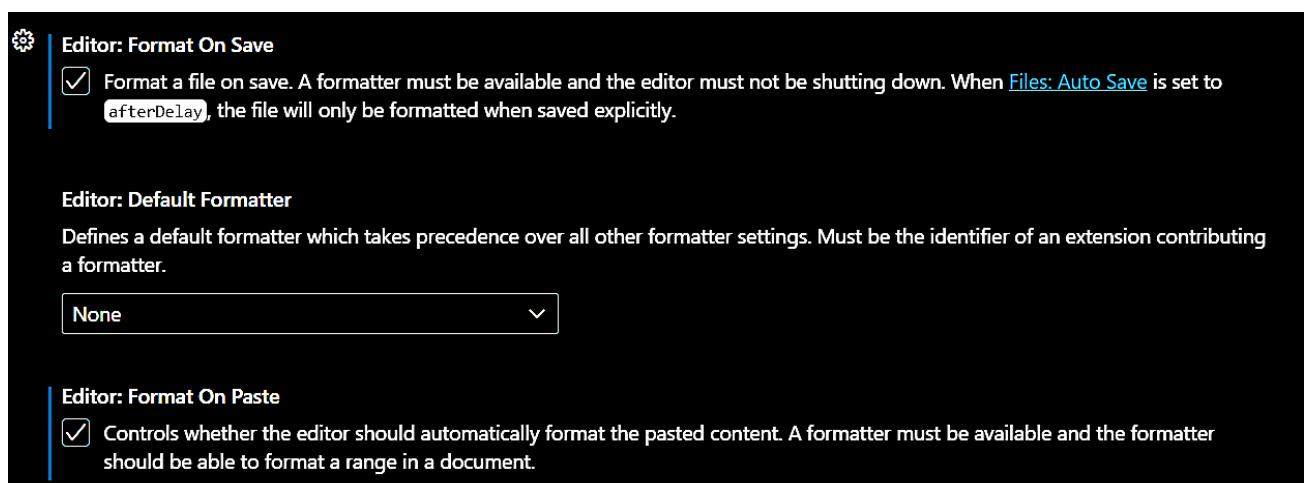
Hình 13: Cài đặt ESLint extension thành công

- Cài đặt Prettier extension. Prettier là công cụ hỗ trợ định dạng tự động các tập tin mã nguồn.



Hình 14: Cài đặt Prettier extension thành công

- Cấu hình VSCode cho phép định dạng mã lệnh khi save và paste



Hình 15: Cấu hình VSCode cho phép định dạng Save & Paste

- Cài đặt gói thư viện ESLint và Prettier với lệnh sau: **npm i -D eslint prettier eslint-config-prettier**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npm i -D eslint prettier eslint-config-prettier
added 86 packages, and audited 156 packages in 5s

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

found 0 vulnerabilities
```

Hình 16: Cài đặt gói thư viện ESLint và Prettier

- Sau đó tạo một tập tin eslintrc.js tại thư mục gốc bằng lệnh: **npx eslint --init**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npx eslint --init
You can also run this command directly using 'npm init @eslint/config@latest'.
Need to install the following packages:
@eslint/create-config@1.10.0
Ok to proceed? (y)

> contactbook-backend@1.0.0 npx
> create-config

@eslint/create-config: v1.10.0

✓ What do you want to lint? · javascript
✓ How would you like to use ESLint? · problems
✓ What type of modules does your project use? · esm
✓ Which framework does your project use? · react
```

```
eslint > node_modules > .eslintrc.js > ...
  1  module.exports = {
  2    ...
  3    env: {
  4      node: true,
  5      commonjs: true,
  6      es2021: true,
  7    },
  8    extends: ["eslint:recommended", "prettier"],
  9  };
```

Hình 17: Tạo một tập tin eslintrc.js

- Và lưu thay đổi trên git

```
git add -u
git add eslint.config.mjs
git commit -m "Cau hinh ESLint cho du an"
```

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add -u
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
② PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add .eslint.config.mjs
fatal: pathspec '.eslint.config.mjs' did not match any files
● PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add eslint.config.mjs
warning: in the working copy of 'eslint.config.mjs', LF will be replaced by CRLF the next time Git touches it
```

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   eslint.config.mjs
    modified:   package-lock.json
    modified:   package.json
```

```
● PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git commit -m "Cau hinh ESLint cho du an"
[master 739362f] Cau hinh ESLint cho du an
 3 files changed, 2973 insertions(+), 194 deletions(-)
 create mode 100644 eslint.config.mjs
○ PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> [ ]
```

Hình 18: Lưu thay đổi trên git

- Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh **git push origin master**



Hình 19: Tập đã tạo thành công trên GitHub

Bước 4: Cài đặt Express

- Tạo tập tin app.js, server.js và index.js lần lượt với nội dung như sau:

- app.js

```
JS eslint.config.mjs | JS app.js 3, U X | JS server.js 2, U
BackEnd > BackEnd_Part1 > ContactBook-backend > JS app.js > app.get("/") callback > message
1 const express = require("express");
2 const cors = require("cors");
3
4 const app = express();
5
6 app.use(cors());
7 app.use(express.json());
8
9 app.get("/", (req, res) => {
10   res.json({ message: "Welcome to contact book application." });
11 });
12
13 module.exports = app;
```

Hình 20: Tạo tập tin app.js

- server.js

```
JS eslint.config.mjs | JS app.js 3, U | JS server.js 2, U X
BackEnd > BackEnd_Part1 > ContactBook-backend > JS server.js > ...
1 const app = require("./app");
2 const config = require("./app/config");
3
4 const PORT = config.app.port;
5 app.listen(PORT, () =>{
6   console.log(`Server is running on port ${PORT}.`);
7 })
```

Hình 21: Tạo tập tin server.js

- Lần lượt tạo thư mục app, app/config và tập tin app/config/index.js

```
JS index.js 2, U X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > config > JS index.js >
1 const config ={
2   app: {
3     port: process.env.PORT || 3000,
4   }
5 };
6 module.exports = config
7
```

Hình 22: Tạo tập tin index.js

- Kiểm tra kết quả chạy server bằng lệnh: **node server.js**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> node server.js
Sever is running on port 3000.
```

Hình 23: Kết quả chạy server thành công

- Kiểm tra kết quả bằng **ứng dụng Postman**

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:3000/
- Headers:** (6 items)
- Body:** (Empty JSON object)
- Response Status:** 200 OK
- Response Time:** 23 ms
- Response Size:** 317 B
- Description:** "Welcome to contact book application."

Hình 24: Kiểm tra kết quả bằng ứng dụng Posman

⇒ Kết nối thành công

- Cài đặt nodemon bằng lệnh: **npm install nodemon --save-dev**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> npm install nodemon --save-dev
added 22 packages, and audited 274 packages in 2s

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

found 0 vulnerabilities
```

Hình 25: Cài đặt nodemon thành công

- Sau khi cài đặt, mở tập tin package.json và thay đổi cấu hình mục “scripts”

```

  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon server.js"
  },
  "author": "Uyen Tran",

```

Hình 26: Thay đổi cấu hình mục “scripts”

- Lưu thay đổi trên vào git

```

git add -u
git add app/ app.js server.js

```

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add -u
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add app/ app.js server.js

```

Hình 27: Lưu thay đổi trên vào git

- Kiểm tra lại bằng lệnh **git status**

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.js
    modified:   package-lock.json
    modified:   package.json
    new file:   server.js

```

Hình 28: Các tệp đã chạy thành công

- Thực hiện lệnh **git commit -m** để lưu các thay đổi lên git

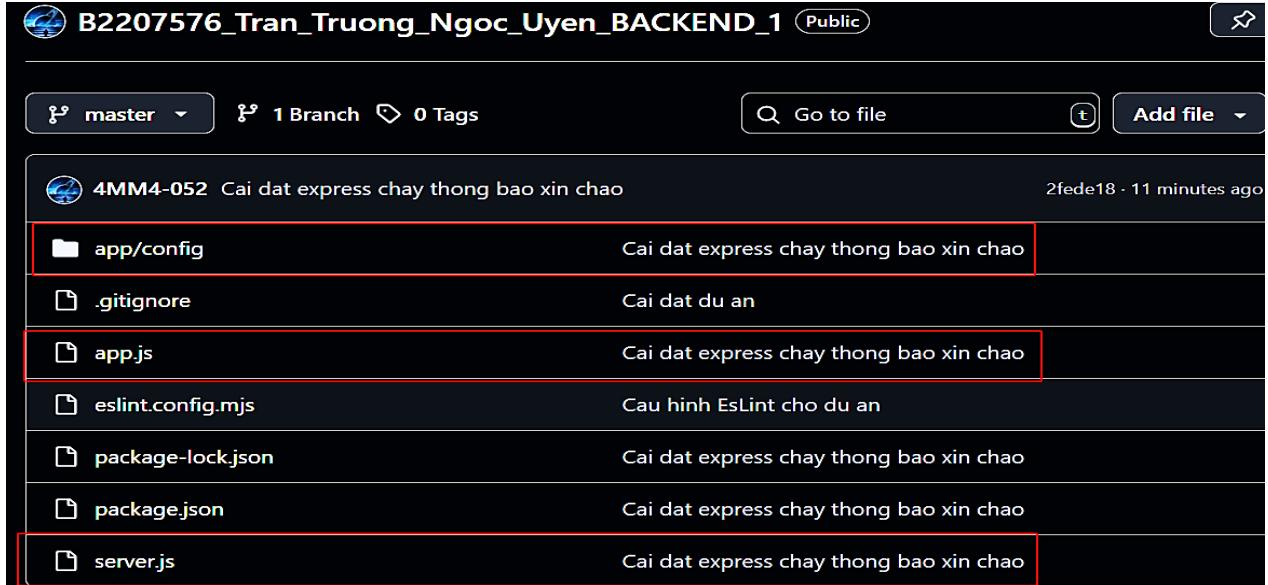
```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git commit -m "Cai dat express chay thong bao xin chao"
[master 5763349] Cai dat express chay thong bao xin chao
 5 files changed, 332 insertions(+), 1 deletion(-)
 create mode 100644 app.js
 create mode 100644 app/config/index.js
 create mode 100644 server.js

```

Hình 29: Được tạo thành công

- Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh `git push`



Hình 30: Tệp đã tải lên GitHub thành công

Bước 5: Định nghĩa controller và các router

- Tạo thư mục controllers trong thư mục app
- Tạo tập tin contact.controller.js trong thư mục controllers

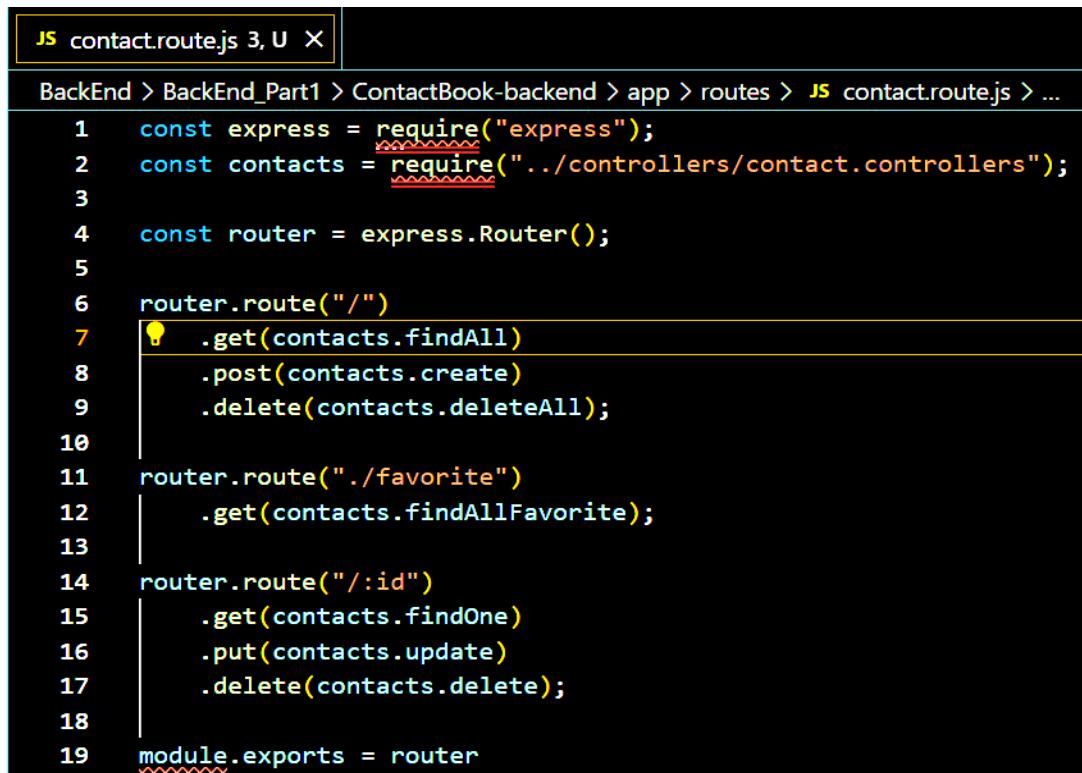
```

JS contact.controller.js 7, U X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > controllers > JS contact.controller.js > ..
1 exports.create = (req, res) =>{
2   res.send({message: "create handler"});
3 };
4 exports.findAll = (req, res) =>{
5   res.send({message: "findAll handler"});
6 };
7 exports.findOne = (req, res) =>{
8   res.send({message: "findOne handler"});
9 };
10 exports.update = (req, res) =>{
11   res.send({message: "update handler"});
12 };
13 exports.delete = (req, res) =>{
14   res.send({message: "delete handler"});
15 };
16 exports.deleteAll = (req, res) =>{
17   res.send({message: "deleteAll handler"});
18 };
19 exports.findAllFavorite = (req, res) =>{
20   res.send({message: "findAllFavorite handler"});
21 };

```

Hình 31: Tạo tập tin contact.controller.js

- Tạo thư mục routes trong thư mục app
- Tạo tập tin contact.route.js trong thư mục routes



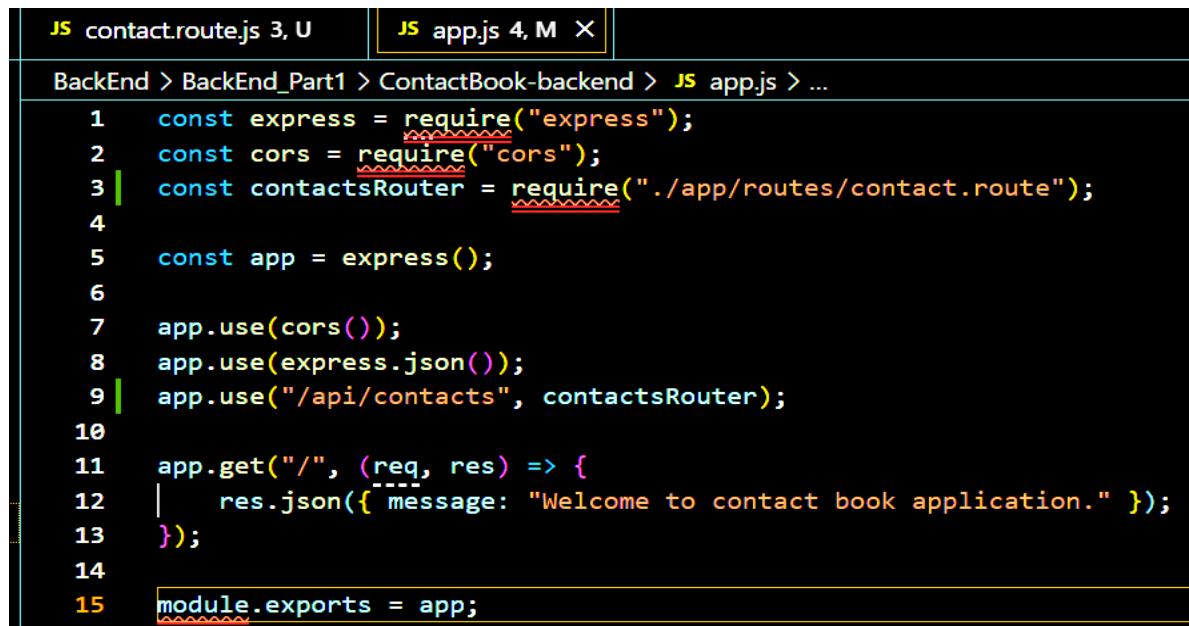
```

JS contact.route.js 3, U X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > routes > JS contact.route.js > ...
1 const express = require("express");
2 const contacts = require("../controllers/contact.controllers");
3
4 const router = express.Router();
5
6 router.route("/")
7   .get(contacts.findAll)
8   .post(contacts.create)
9   .delete(contacts.deleteAll);
10
11 router.route("./favorite")
12   .get(contacts.findAllFavorite);
13
14 router.route("/:id")
15   .get(contacts.findOne)
16   .put(contacts.update)
17   .delete(contacts.delete);
18
19 module.exports = router

```

Hình 32: Tạo tập tin contact.route.js

- Cập nhật lại app.js



```

JS contact.route.js 3, U JS app.js 4, M X
BackEnd > BackEnd_Part1 > ContactBook-backend > JS app.js > ...
1 const express = require("express");
2 const cors = require("cors");
3 const contactsRouter = require("./app/routes/contact.route");
4
5 const app = express();
6
7 app.use(cors());
8 app.use(express.json());
9 app.use("/api/contacts", contactsRouter);
10
11 app.get("/", (req, res) => {
12   res.json({ message: "Welcome to contact book application." });
13 });
14
15 module.exports = app;

```

Hình 33: Cập nhật lại app.js

- Kiểm tra các route vừa tạo bằng ứng dụng Postman

- Ví dụ 1: findAll

GET http://localhost:3000/api/contacts

Params Auth Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Body	200 OK • 4 ms • 296 B • e.g. Sav	{} JSON ▾ Preview Visualize

```

1 {
2   "message": "findAll handler"
3 }

```

Hình 34: Kết quả chạy thành công

- Ví dụ 2: findAllFavorite

GET http://localhost:3000/api/contacts/favorite

Params Authorization Headers (7) Body Scripts Settings

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

This request does not have a body

Body Cookies Headers (8) Test Results (1/1) ↻ 200 OK

{} JSON ▾ Preview Visualize

```

1 {
2   "message": "findAllFavorite handler"
3 }

```

Hình 35: Kết quả chạy thành công

- Lưu thay đổi trên vào git

```
git add -u
git add app/controllers app/routes
```

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add -u
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add app/controllers app/routes
```

Hình 36: Lưu thay đổi trên vào git

- Kiểm tra lại bằng lệnh **git status**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   app.js
    new file:   app/controllers/contact.controller.js
    new file:   app/routes/contact.route.js
```

Hình 37: Các tệp đã chạy thành công

- Thực hiện lệnh **git commit -m** để lưu các thay đổi lên git

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git commit -m "Dinh nghia cac route cho tai nguyen Contact"
[master 3419b70] Dinh nghia cac route cho tai nguyen Contact
 3 files changed, 42 insertions(+)
 create mode 100644 app/controllers/contact.controller.js
 create mode 100644 app/routes/contact.route.js
```

Hình 38: Được tạo thành công

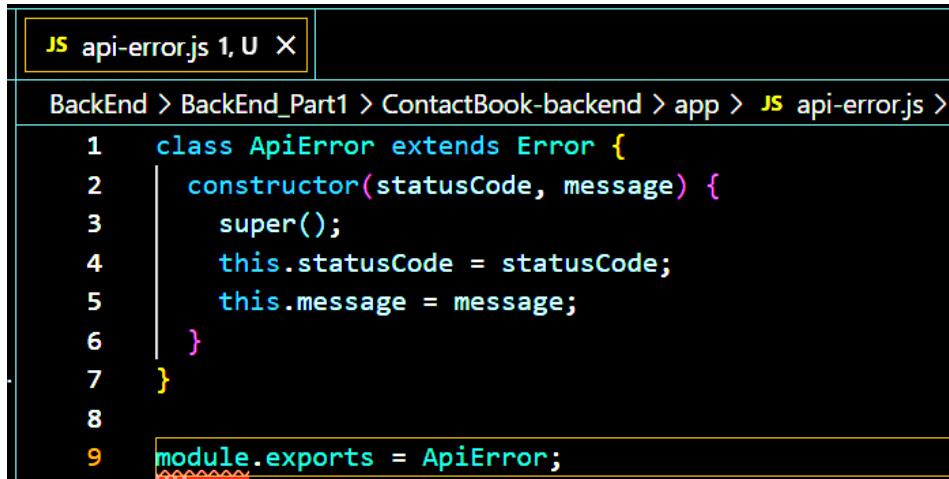
- Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh **git push**



Hình 39: Tệp đã tải lên GitHub thành công

Bước 6: Cài đặt xử lý lỗi

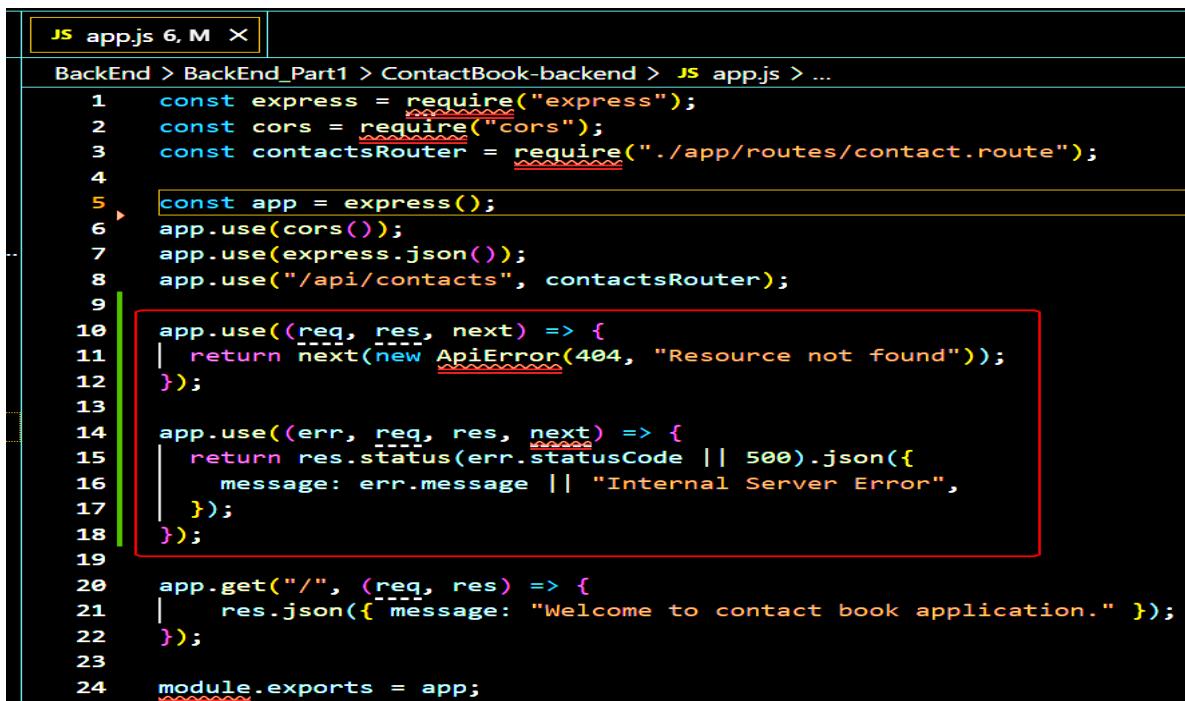
- Tạo tập tin app/api-error.js



```
JS api-error.js 1, U X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > JS api-error.js >
1  class ApiError extends Error {
2  |   constructor(statusCode, message) {
3  |     super();
4  |     this.statusCode = statusCode;
5  |     this.message = message;
6  |   }
7  }
8
9  module.exports = ApiError;
```

Hình 40: Tạo tập tin api-error.js

- Mở tập tin app.js thêm các middleware xử lý lỗi



```
JS app.js 6, M X
BackEnd > BackEnd_Part1 > ContactBook-backend > JS app.js > ...
1  const express = require("express");
2  const cors = require("cors");
3  const contactsRouter = require("./app/routes/contact.route");
4
5  const app = express();
6  app.use(cors());
7  app.use(express.json());
8  app.use("/api/contacts", contactsRouter);
9
10 app.use((req, res, next) => {
11   return next(new ApiError(404, "Resource not found"));
12 });
13
14 app.use((err, req, res, next) => {
15   return res.status(err.statusCode || 500).json({
16     message: err.message || "Internal Server Error",
17   });
18 });
19
20 app.get("/", (req, res) => {
21   res.json({ message: "Welcome to contact book application." });
22 });
23
24 module.exports = app;
```

Hình 41: Thêm các middleware xử lý lỗi trong app.js

- Kiểm tra lại bằng ứng dụng Postman

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/112233`. The 'Body' tab is selected, showing an empty JSON object. The response section indicates a `404 Not Found` error with the message `"message": "Resource not found"`.

Hình 42: Kiểm tra kết quả bằng ứng dụng Posman

- Lưu thay đổi trên vào git

```
git add -u
git add app/api-error.js
```

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add -u
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git add app/api-error.js
```

Hình 43: Lưu thay đổi trên vào git

- Kiểm tra lại bằng lệnh `git status`

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   app.js
    new file:   app/api-error.js
```

Hình 44: Các tệp đã chạy thành công

- Thực hiện lệnh `git commit -m` để lưu các thay đổi lên git

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part1>ContactBook-backend> git commit -m "Cai dat xu ly loi"
[master f641bae] Cai dat xu ly loi
2 files changed, 19 insertions(+), 1 deletion(-)
create mode 100644 app/api-error.js
```

Hình 45: Được tạo thành công

- Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh **git push**

The screenshot shows a GitHub repository interface. On the left, the file tree for the 'master' branch is visible, including 'app', 'config', 'controllers', 'routes', 'api-error.js', '.gitignore', and 'app.js'. The 'api-error.js' file is selected. On the right, the code editor displays the 'api-error.js' file with the following content:

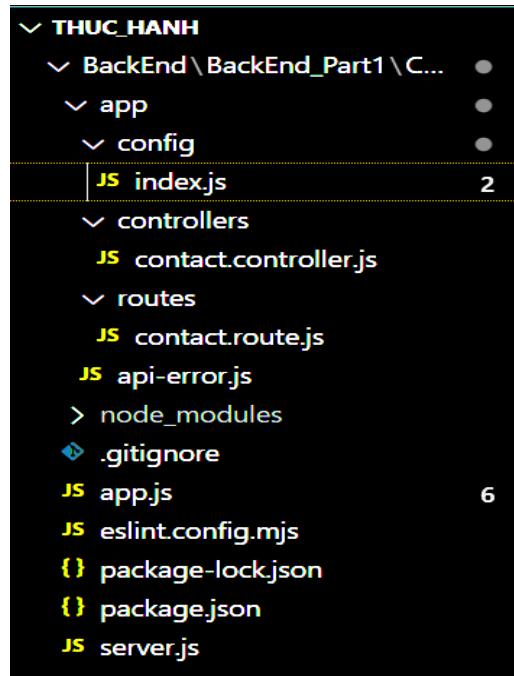
```

1  class ApiError extends Error {
2    constructor(statusCode, message) {
3      super();
4      this.statusCode = statusCode;
5      this.message = message;
6    }
7  }
8
9  module.exports = ApiError;

```

Hình 46: Tệp đã tải lên GitHub thành công

- Cấu trúc thư mục dự án đến hiện tại như sau:



Hình 47: Cây thư mục hiện tại

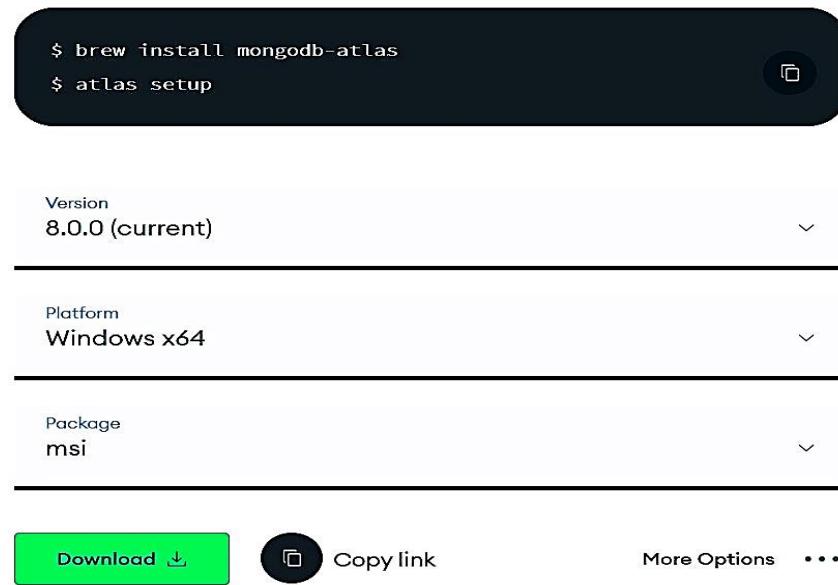
ĐƯỜNG LINK GITHUB:

Link mã nguồn:

https://github.com/4MM4-052/B2207576_Tran_Truong_Ngoc_Uyen_BACKEND_1

Ứng dụng Contactbook - Backend - Phần 2

Bước 0: Cài đặt MongoDB



Hình 48: Cài đặt MongoDB

Bước 1: Cài đặt thư viện MongoDB, định nghĩa hàm trợ giúp kết nối và lớp dịch vụ truy xuất cơ sở dữ liệu (CSDL)

- Cài đặt thư viện mongodb vào dự án: **npm install mongodb**

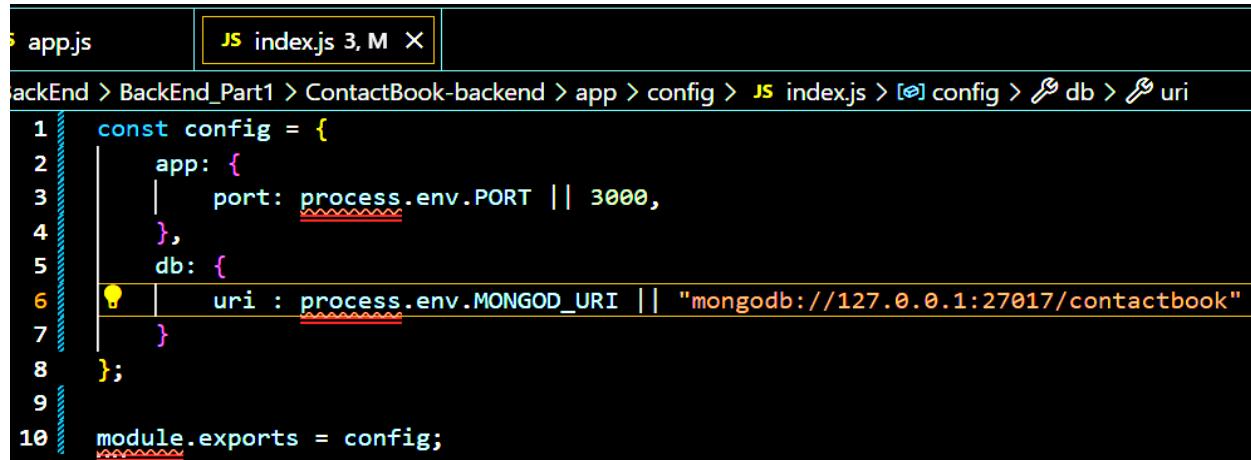
```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> npm install mongodb
+ 11 packages from 11 authors and 285 dependencies added
added 11 packages, and audited 285 packages in 3s

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

found 0 vulnerabilities
```

Hình 49: Cài đặt thư viện mongodb

- Trong thư mục app/config, hiệu chỉnh thêm đoạn code trong tập tin index.js



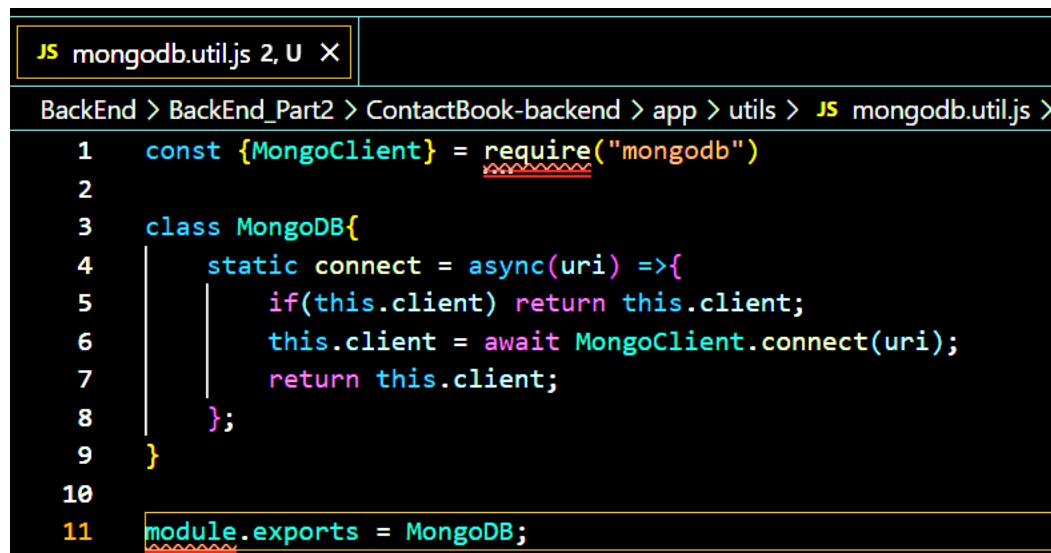
```

app.js          JS index.js 3, M X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > config > JS index.js > [o] config > db > uri
1 const config = {
2   app: {
3     port: process.env.PORT || 3000,
4   },
5   db: {
6     uri : process.env.MONGOD_URI || "mongodb://127.0.0.1:27017/contactbook"
7   }
8 };
9
10 module.exports = config;

```

Hình 50: Hiệu chỉnh thêm đoạn code trong tập tin index.js

- Định nghĩa lớp trợ giúp kết nối đến MongoDB: app/utils/mongodb.util.js



```

JS mongodb.util.js 2, U X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > utils > JS mongodb.util.js >
1 const {MongoClient} = require("mongodb")
2
3 class MongoDB{
4   static connect = async(uri) =>{
5     if(this.client) return this.client;
6     this.client = await MongoClient.connect(uri);
7     return this.client;
8   }
9 }
10
11 module.exports = MongoDB;

```

Hình 51: Tạo tập tin mongodb.util.js

- Thay toàn bộ tập tin server.js

```

JS server.js 4, M X
BackEnd > BackEnd_Part2 > ContactBook-backend > JS server.js > startServer
1  const app = require("./app");
2  const config = require("./app/config");
3  const MongoDB = require("./app/utils/mongodb.util");
4
5  async function startServer() {
6    try {
7      await MongoDB.connect(config.db.uri);
8      console.log("Connected to the database!");
9
10     const PORT = config.app.port;
11     app.listen(PORT, () => {
12       |   console.log(`Server is running on port ${PORT}`);
13     });
14   } catch (error) {
15     console.log("Cannot connect to the database!", error);
16     process.exit();
17   }
18 }
19
20 startServer();

```

Hình 52: Thay đổi nội dung tập tin server.js

- Định nghĩa các lớp dịch vụ ContactServices (trong tập tin app/services/service.mjs) chứa các API của thư viện MongoDB để thực hiện thao tác với CSDL MongoDB

```

1  const { ObjectId } = require("mongodb");
2
3  class ContactService {
4    constructor(client) {
5      this.Contact = client.db().collection("contacts");
6    }
7
8    // Định nghĩa các phương thức truy xuất CSDL sử dụng mongodb API
9  }
10
11 module.exports = ContactService;
12

```

Hình 53: Định nghĩa các lớp dịch vụ ContactServices

Bước 2: Cài đặt các handler

1. Cài đặt handler create

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
JS contact.controller.js 9+, M X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > controllers > JS contact.controller.js > ...
104 exports.create = async(req,res,next) =>{
105   if(!req.body?.name){
106     return next(new ApiError(400, "name can not be empty"));
107   }
108   try{
109     const contactService = new ContactService(MongoDB.client);
110     const document = await contactService.create(req.body);
111     return res.send(document);
112   } catch(error){
113     return next(
114       new ApiError(500, "An error occurred while creating the contact")
115     );
116   }
117};
```

Hình 54: Cài đặt handler create

- Tạo phương thức create() trong lớp ContactService

```
JS server.js 4, M JS contact.service.js 2, U X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > services > JS contact.service.js > ...
1  const { ObjectId } = require("mongodb");
2
3  class ContactService {
4    constructor(client) {
5      this.Contact = client.db().collection("contacts");
6    }
7    // Định nghĩa các phương thức truy xuất CSDL sử dụng mongodb API
8    extractContactData(payload) {
9      const contact = {
10        name: payload.name,
11        email: payload.email,
12        address: payload.address,
13        phone: payload.phone,
14        favorite: payload.favorite,
15      };
16
17      // Remove undefined fields
18      Object.keys(contact).forEach(
19        (key) => contact[key] === undefined && delete contact[key];
20      );
21
22      return contact;
23    }
24
25    async create(payload) {
26      const contact = this.extractContactData(payload);
27      const result = await this.Contact.findOneAndUpdate(
28        contact,
29        { $set: { favorite: contact.favorite === true } },
30        { returnDocument: "after", upsert: true }
31      );
32      return result;
33    }
34  }
```

Hình 55: Tạo phương thức create()

– Kiểm tra với **ứng dụng Postman cho create**

- Để gửi dữ liệu JSON về sever với Postman, cần đặt Header “Content-Type: application/json” và đặt dữ liệu JSON trong phần Body của yêu cầu

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/contacts`. The 'Headers' tab is selected, showing a table with one row. The 'Key' column contains 'Content-Type' and the 'Value' column contains 'application/json'. A red box highlights the 'Content-Type' row.

Key	Value
Content-Type	application/json

Hình 56: Hiệu chỉnh Header "Content-Type: application/json"

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/contacts`. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2   "name": "Tran Truong Ngoc Uyen",  
3   "email": "uyenb2207576@student.edu.ctu.vn",  
4   "address": "Can Tho",  
5   "phone": "0123456789",  
6   "favorite": false  
7 }
```

The response shows a 200 OK status with 79 ms and 433 bytes. The JSON response body is identical to the payload sent:

```
1 {  
2   "_id": "68aaafca324ce9a9ff12b537",  
3   "address": "Can Tho",  
4   "favorite": false,  
5   "name": "Tran Truong Ngoc Uyen",  
6   "phone": "0123456789",  
7   "email": "uyenb2207576@student.edu.ctu.vn"  
8 }
```

Hình 57: Đặt dữ liệu JSON trong phần Body

- Kết quả bên MongoDB:

```

_id: ObjectId('68aaafca324ce9a9ff12b537')
address : "Can Tho"
favorite : false
name : "Tran Truong Ngoc Uyen"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"

```

Hình 58: Kết quả MongoDB

2. Cài đặt handler findAll

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```

exports.findAll = async (req, res, next) => {
  let document = [];
  try{
    const contactService = new ContactService(MongoDB.client);
    const {name} = req.query;
    if(name){
      document = await contactService.findByName(name);
    }
    else{
      document = await contactService.find({});
    }
  } catch(error){
    return next(
      new ApiError(500, "An error occurred while retrieving contacts")
    );
  }
  return res.send(document)
}

```

Hình 59: Cài đặt handler findAll

- Tạo 2 phương thức find(), findByName() trong lớp ContactService

```

    async find(filter) {
      const cursor = await this.Contact.find(filter);
      return await cursor.toArray();
    }

    async findByName(name) {
      return await this.find({
        name: { $regex: new RegExp(name), $options: "i" },
      });
    }

```

Hình 60: Tạo 2 phương thức find(), findByName()

- Kiểm tra với ứng dụng Postman cho findAll

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/api/contacts`. The request body is a JSON object:

```

1  {
2   "name": "Tran Truong Ngoc Uyen",
3   "email": "uyenb2207576@student.edu.ctu.vn",
4   "address": "Can Tho",
5   "phone": "0123456789",
6   "favorite": false
7 }
  
```

The response status is 200 OK with a response time of 79 ms and a size of 433 bytes. The response body is also a JSON object:

```

1  {
2   "_id": "68aaafca324ce9a9ff12b537",
3   "address": "Can Tho",
4   "favorite": false,
5   "name": "Tran Truong Ngoc Uyen",
6   "phone": "0123456789",
7   "email": "uyenb2207576@student.edu.ctu.vn"
8 }
  
```

Hình 61: Kết quả chạy thành công

3. Cài đặt handler findOne

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```

exports.findOne = async (req, res, next) =>{
  try{
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.findById(req.params.id);
    if(!document){
      return next(new ApiError(404, "Contact not found"));
    }
    return res.send(document);
  } catch(error){
    return next(
      new ApiError(
        500,
        `Error retrieving contact with id = ${req.params.id}`
      )
    );
  };
}
  
```

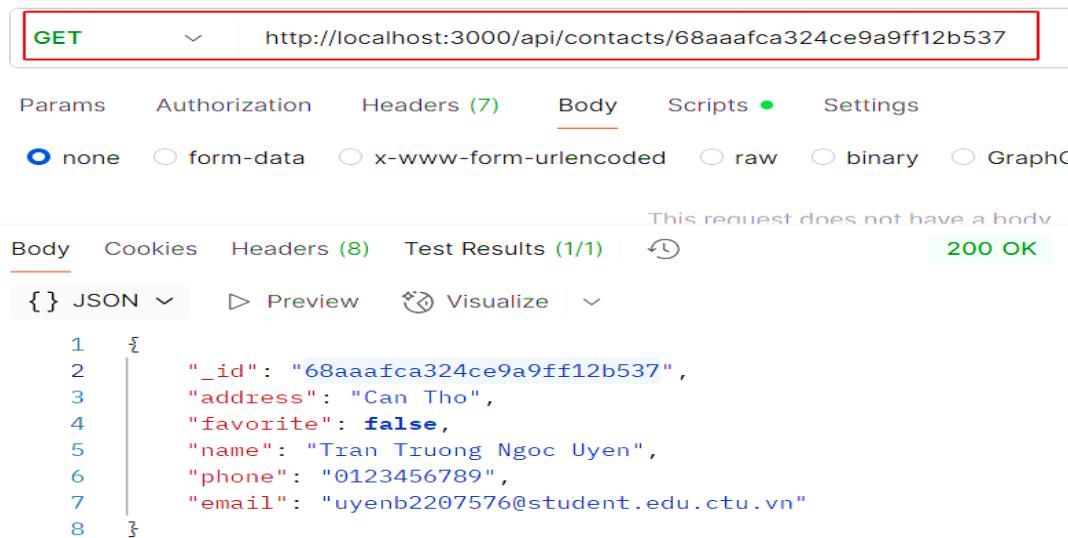
Hình 62: Cài đặt handler findOne

- Tạo phương thức findById() trong lớp ContactService

```
async findById(id) {
  return await this.Contact.findOne({
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
  });
}
```

Hình 63: Tạo phương thức findById()

- Kiểm tra với ứng dụng Postman cho findOne



Hình 64: Kết quả chạy thành công

4. Cài đặt handler update

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
exports.update = async (req, res, next) =>{
  if(Object.keys(req.body).length === 0){
    return next(new ApiError(400, "data to update can not be empty"));
  }
  try{
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.update(req.params.id, req.body);
    if(!document){
      return next(new ApiError(404, "contact not found"));
    }
    return res.send({message: "Contact was updated successfully"});
  } catch(error){
    return next(
      new ApiError(404, `Error updating contact with id = ${req.params.id}`));
  }
};
```

Hình 65: Cài đặt handler update

- Tạo phương thức update() trong lớp ContactService

```

async update(id, payload) {
  const filter = {
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
  };
  const update = this.extractContactData(payload);
  const result = await this.Contact.findOneAndUpdate(
    filter,
    { $set: update },
    { returnDocument: "after" }
  );
  return result.value; //return the updated document
}

```

Hình 66: Tạo phương thức update()

- Kiểm tra với ứng dụng Postman cho update

The screenshot shows a POST request in Postman to the URL `http://localhost:3000/api/contacts/68aaafca324ce9a9ff12b537`. The request body is a JSON object:

```

1  {
2    "name": "Uyen Tran B2207576 Update"
3    "email": "uyenb2207576@student.edu.ctu.vn",
4    "address": "An Giang",
5    "phone": "0123456789",
6    "favorite": false
7  }

```

The response status is 200 OK with a response time of 54 ms. The response body is:

```

1  {
2    "message": "Contact was updated successfully"
3  }

```

Hình 67: Kết quả chạy thành công

- Trước khi update bên MongoDB

```

_id: ObjectId('68aaafca324ce9a9ff12b537')
address : "Can Tho"
favorite : false
name : "Tran Truong Ngoc Uyen"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"

```

Hình 68: Trước khi update bên MongoDB

- Sau khi update bên MongoDB

```

▶ _id: ObjectId('68aaafca324ce9a9ff12b537')
address : "An Giang"
favorite : false
name : "Uyen Tran B2207576 Update"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"

```

Hình 69: Sau khi update bên MongoDB

⇒ Update thành công

5. Cài đặt handler delete

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```

exports.delete = async (req, res, next) =>{
  try{
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.delete(req.params.id);
    if(!document){
      return next (new ApiError(404, " Contact not found "));
    }
    return res.send({massege: "Contact was deleted successfully"});
  } catch(error){
    return next(
      new ApiError(
        500,
        `Could not delete contact with id =${req.params.id}`
      )
    );
  }
};

```

Hình 70: Cài đặt handler delete

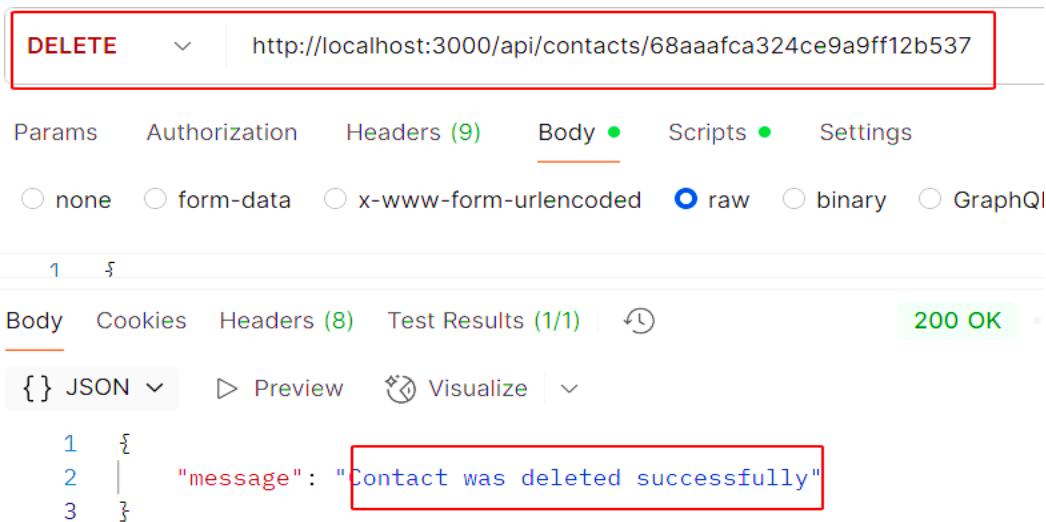
- Tạo phương thức delete() trong lớp ContactService

```

    async delete(id) {
      const result = await this.Contact.findOneAndDelete({
        _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
      });
      return result;
    }
  
```

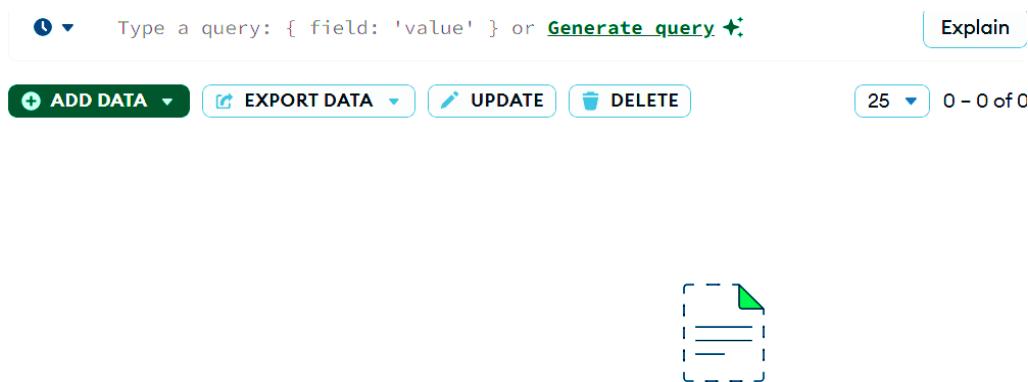
Hình 71: Tạo phương thức create()

- Kiểm tra với ứng dụng Postman cho delete



Hình 72: Kết quả chạy thành công

- Kiểm tra bên MongoDB được xóa thành công



Hình 73: Kết quả được xóa thành công

6. Cài đặt handler findAllFavorite

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
exports.findAllFavorite = async (_req, res, next) =>{
  try{
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.findFavorite();
    return res.send(document);
  } catch(error){
    return next(
      new ApiError(
        500,
        "An error occurred while retrieving favorite contacts"
      );
  }
};
```

Hình 74: Cài đặt handler findAllFavorite

- Tạo phương thức findFavorite() trong lớp ContactService

```
async findFavorite() {
  return await this.find({ favorite: true });
}
```

Hình 75: Tạo phương thức findFavorite()

- Kiểm tra với ứng dụng Postman cho findAllFavorite



Hình 76: Kết quả chạy thành công cho ra rỗng

- Khi **favorite = true** thì kết quả sẽ hiện thị toàn bộ thông tin của người đó

The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** http://localhost:3000/api/contacts/favorite
- Params:** none
- Headers:** (7)
- Body:** (Raw JSON response shown below)
- Test Results:** (1/1) 200 OK
- Visualize:** Preview of the JSON response.

The JSON response body is:

```

1   [
2     {
3       "_id": "68aab988324ce9a9ff12b53a",
4       "favorite": true,
5       "email": "uyenb2207576@student.edu.ctu.vn",
6       "address": "Can Tho",
7       "name": "Uyen Tran B2207576",
8       "phone": "0123456789"
9     }
10  ]

```

Hình 77: Kết quả hiện thị thành công các thông tin

7. Cài đặt handler deleteAll

- Hiệu chỉnh tập tin app/controllers/contact.controller.js

```

exports.deleteAll = async (req, res, next) =>{
  try{
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.deleteAll();
    return res.send({
      message: `${deleteCount} contacts were deleted successfully`,
    });
  }catch(error){
    return next(
      new ApiError(500, "An error occurred while removing all contacts")
    );
  }
};

```

Hình 78: Cài đặt handler deleteAll

- Tạo phương thức deleteAll() trong lớp ContactService

```

        }
    }

module.exports = ContactService;

```

Hình 79: Tạo phương thức deleteAll()

- Kiểm tra với **íng dụng Postman** cho deleteAll

The screenshot shows the Postman interface with a red box highlighting the URL field: `DELETE http://localhost:3000/api/contacts/`. Below the URL, the 'Body' tab is selected, showing the request method as 'none'. The response section shows a status code of 200 and a JSON body:

```

{
  "message": "1 contacts were deleted successfully"
}

```

Hình 80: Kết quả được xóa thành công

- Lưu thay đổi trên vào git

```

git add -u
git add app/utils app/services

```

```

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> git add -u
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'eslint.config.mjs', LF will be replaced by CRLF the next time Git touches it

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> git add app/utils app/services

```

Hình 81: Lưu thay đổi trên vào git

- Kiểm tra lại bằng lệnh `git status`

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> git status
On branch master
Your branch is up to date with 'backup/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   app.js
    modified:   app/api-error.js
    modified:   app/config/index.js
    modified:   app/controllers/contact.controller.js
    modified:   app/routes/contact.route.js
    new file:   app/services/contact.service.js
    new file:   app/utils/mongodb.util.js
```

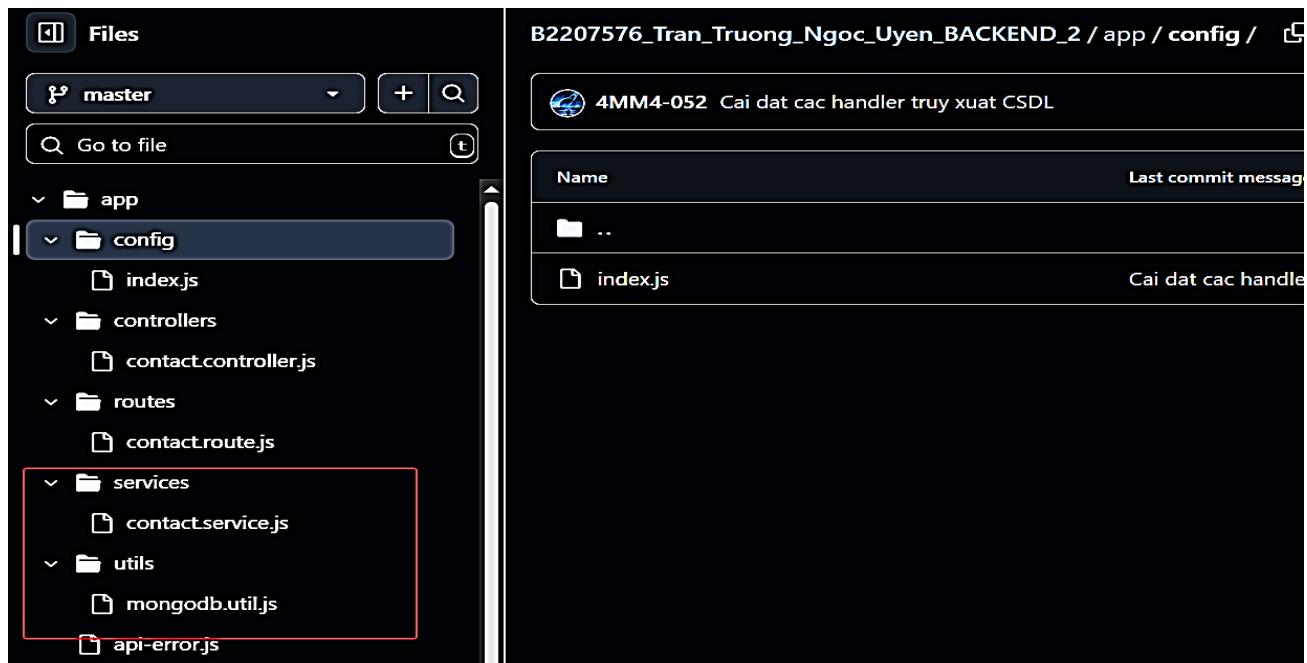
Hình 82: Các tệp đã chạy thành công

- Thực hiện lệnh `git commit -m` để lưu các thay đổi lên git

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> git commit -m "Cai dat cac handler truy xuat CSDL"
[master cb889e4] Cai dat cac handler truy xuat CSDL
 10 files changed, 420 insertions(+), 46 deletions(-)
 create mode 100644 app/services/contact.service.js
 create mode 100644 app/utils/mongodb.util.js
```

Hình 83: Được tạo thành công

- Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh `git push`



Hình 84: Tệp đã tải lên GitHub thành công

- Cấu trúc thư mục dự án đến hiện tại như sau:

```
BackEnd_Part2
  ContactBook-backend
    app
      config
        index.js
      controllers
        contact.controller.js
      routes
        contact.route.js
      services
        contact.service.js
      utils
        mongodb.util.js
      api-error.js
    node_modules
    .gitignore
    app.js
    eslint.config.mjs
    package-lock.json
    package.json
    server.js
```

Hình 85: Cây thư mục hiện tại

ĐƯỜNG LINK GITHUB:

Link mã nguồn:

https://github.com/4MM4-052/B2207576_Tran_Truong_Ngoc_Uyen_BACKEND_2