



Département de Génie Informatique et Communication
Department of Information and Communication Engineering

Topic : TP10

Subject : Internet Programming

Teacher : Mr. Hok Tin

Name : YORNG TONGHY

ID : e20191313

Group : I4-GIC-C

Year : 2022-2023

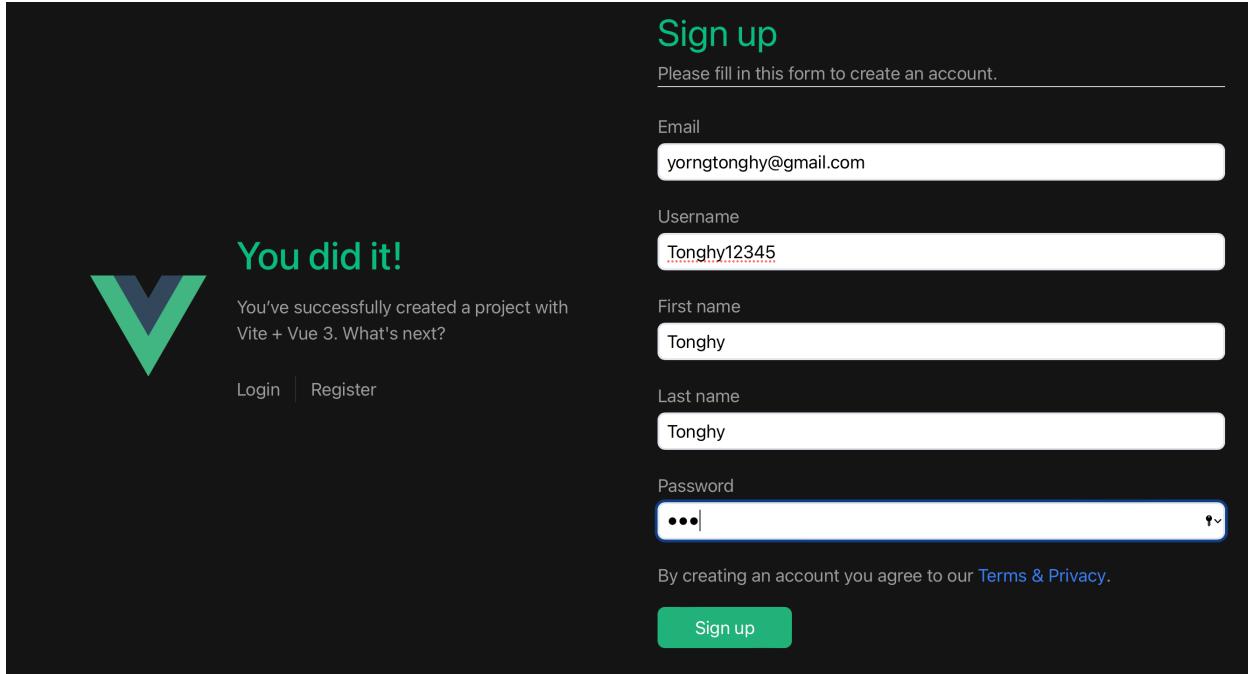
| | |
|---|----------|
| - Table of Contents | |
| - <i>Table of Contents</i> | 2 |
| • <i>GitHub link</i> | 3 |
| <i>Exercise1: Integrate the previous authentication APIs with VueJS.</i> | 3 |
| • Register | 3 |
| • Register account successfully..... | 4 |
| • Data in MongoDB after register it's load new data in database..... | 4 |
| • Home-Page | 4 |
| <i>Exercise2: Continue implementing the authentication and user APIs</i> | 5 |
| • Result of Ex02: | 5 |
| • login user and get token: | 5 |
| • Register | 5 |
| • Register already existed..... | 6 |
| • When No Token..... | 6 |
| • Invalid Token | 7 |
| • Access by username via correct token..... | 7 |
| • When update user with invalid email format:..... | 8 |
| • Update user Successful: | 8 |
| • When update password successfully:..... | 9 |
| • Delete User..... | 9 |
| • Data before delete:..... | 9 |
| • When delete user does not exist in database..... | 10 |
| • Delete user successfully..... | 10 |

- GitHub link: https://github.com/yorng-tonghy/TP08_IP-TP09_IP/tree/TP10

Exercise1: Integrate the previous authentication APIs with VueJS.

- Result of Exercise1:

- Login



Sign up

Please fill in this form to create an account.

Email
yorngtonghy@gmail.com

Username
Tonghy12345

First name
Tonghy

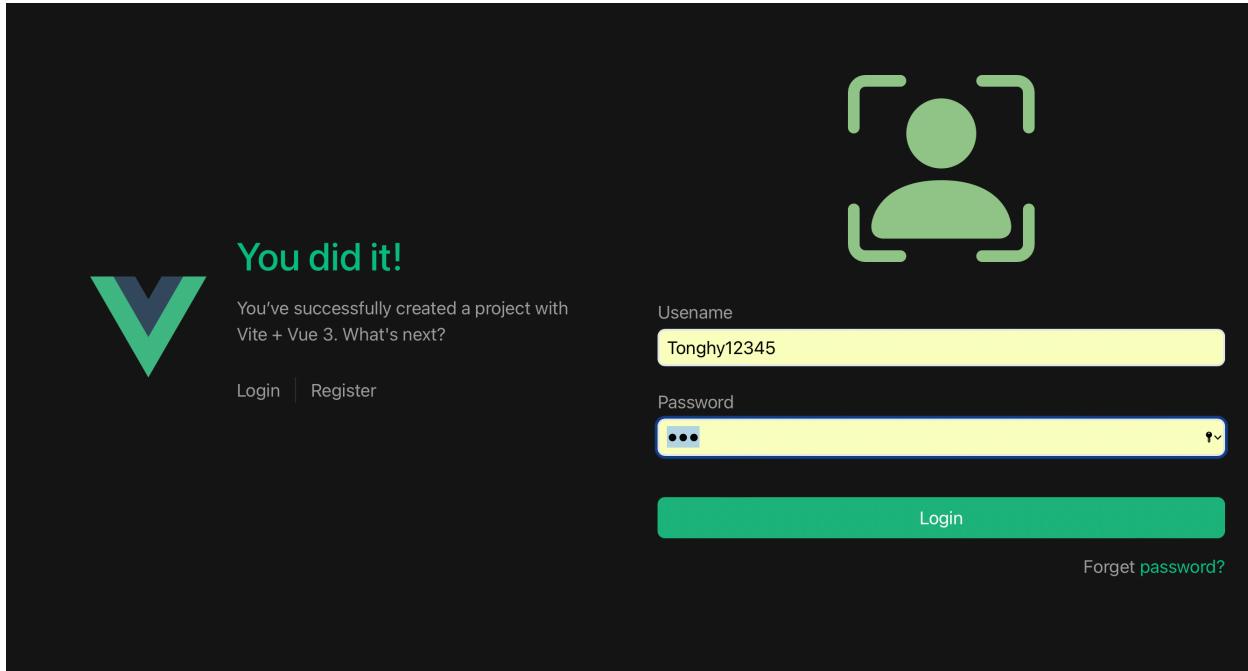
Last name
Tonghy

Password
•••|

By creating an account you agree to our [Terms & Privacy](#).

[Sign up](#)

- Register



You did it!

You've successfully created a project with Vite + Vue 3. What's next?

Login | Register

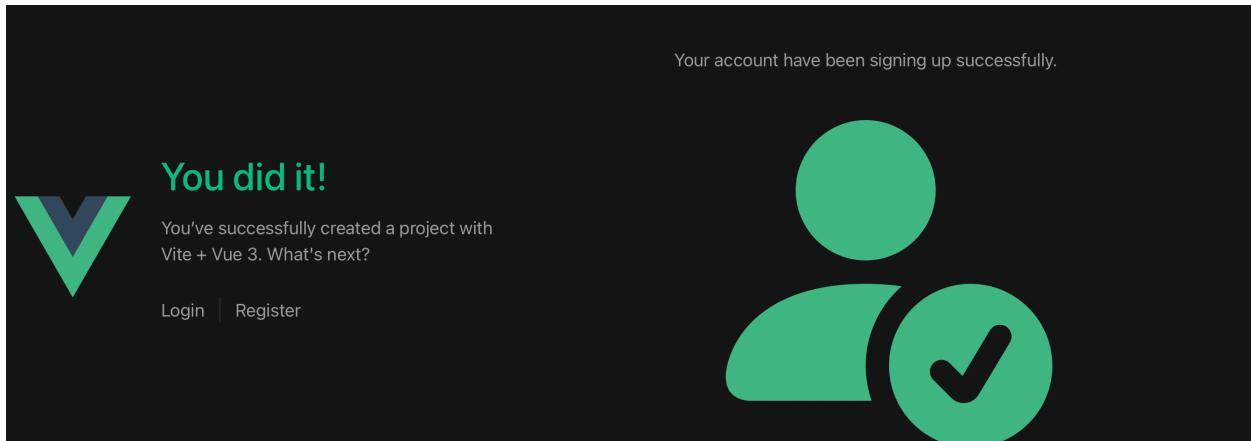
Username
Tonghy12345

Password
•••|

[Login](#)

[Forgot password?](#)

- Register account successfully



- Data in MongoDB after register it's load new data in database.

rs

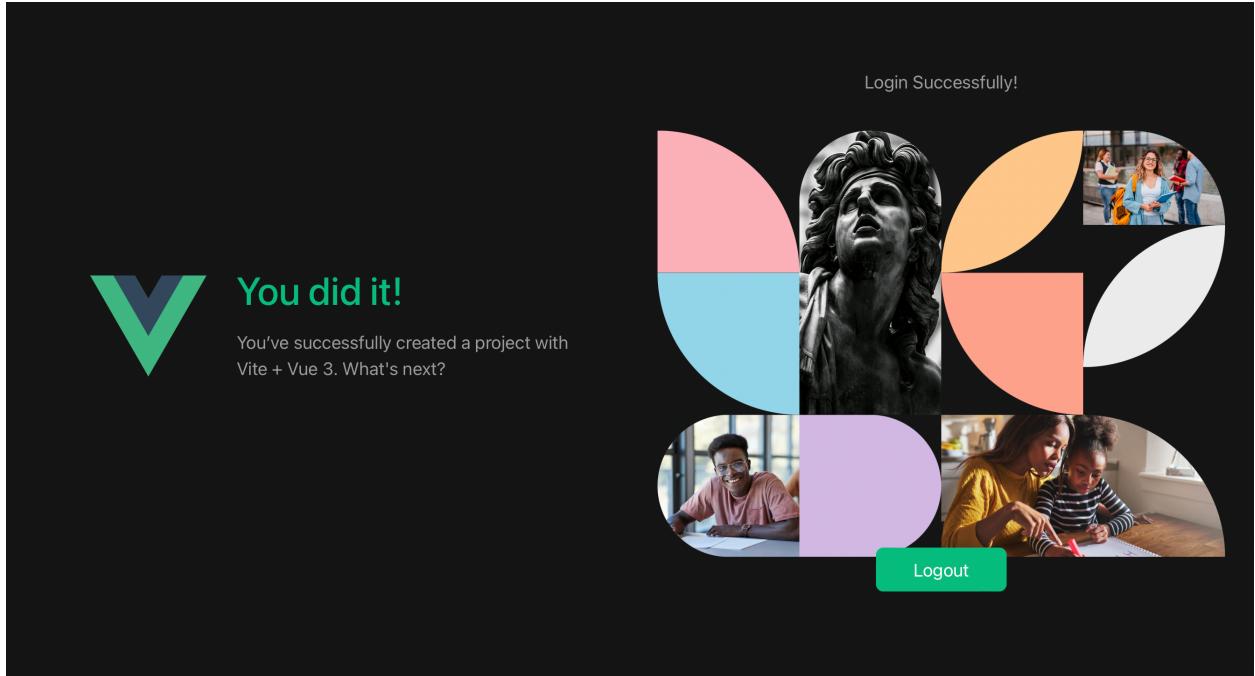
users

ADD DATA **EXPORT DATA**

1 - 7 of 7

| | _id | ObjectId | username | String | firstname | String | lastname | String | email | String | pe | | |
|---|----------------------------------|----------------|----------|----------------|-----------|----------------|----------|--------------------------|-------|--------|----|--|--|
| 1 | ObjectId('6471926a5819ba802...') | "Channararoth" | | "Channararoth" | | "Channararoth" | | "channararoth@gmail.com" | "\$ | | | | |
| 2 | ObjectId('64718dcf5819ba802...') | "Guest" | | "Guest" | | "Guest" | | "Guest@gmail.com" | "\$ | | | | |
| 3 | ObjectId('64718aa25819ba802...') | "Admin" | | "Admin" | | "Admin" | | "admin@gmail.com" | "\$ | | | | |
| 4 | ObjectId('64717f1a5819ba802...') | "Tonghy12345" | | "Tonghy" | | "Tonghy" | | "yongtonghy@gmail.com" | "\$ | | | | |
| 5 | ObjectId('6471c90e1ace404c0...') | "SokVatey3899" | | "SokVatey" | | "SokVatey" | | "SokVatey@gmail.com" | "\$ | | | | |
| 6 | ObjectId('6472a017aabba63e0...') | "Votey" | | "Votey" | | "Chan" | | "chanvatey@gmail.com" | "\$ | | | | |
| 7 | ObjectId('6472a2836c943499b...') | "SokDara" | | "Dara" | | "Sok" | | "sokdara1234@gmail.com" | "\$ | | | | |

- Home-Page



Exercise2: Continue implementing the authentication and user APIs

- Result of Ex02:
- login user and get token:

The screenshot shows a POST request in Postman. The request body is a JSON object with "username": "admin123" and "password": "123". The response status is 200 OK, and the response body is a JSON token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NDdmZWR1NjRmY2Q5ODBhOWRmYjU0Y2QiLCJpYXQiOjE2ODYxMDU3MjIsImV4cCI6MTY4NjEwOTMyMn0.M30lEnYrwJItF1egz3gPW-Tq2J21SQNfFPu0vLRUmWA.

- Register

The screenshot shows a POST request in Postman. The request body is a JSON object with "email": "admin@gmail.com", "username": "admin123", and "password": "123". The response status is 200 OK, and the response body is a JSON object with "user": "647fede64fc980a9dfb54cd".

- Register already existed.

The screenshot shows a POST request to `/register`. The request body is JSON:

```

1
2   "email": "admin@gmail.com",
3   "username": "admin123",
4   "password": "123"
5
6
7
8
  
```

The response status is 400 Bad Request, with the message: "Username is already existed!"

- When No Token

The screenshot shows a GET request to `/user/me`. The Headers tab shows several standard headers checked:

| Key | Description |
|-----------------|--------------------------------|
| Accept | <code>*/*</code> |
| Accept-Encoding | <code>gzip, deflate, br</code> |
| Connection | <code>keep-alive</code> |
| | |
| | |

The response status is 401 Unauthorized, with the message: "Access Denied!"

- Invalid Token

The screenshot shows a Postman request for `TP08 EX2 / getUserById` with a `GET` method and URL `http://localhost:3001/user/me`. The **Headers** tab is selected, containing the following fields:

| Header | Value |
|-----------------|---|
| User-Agent | PostmanRuntime/7.32.2 |
| Accept | * |
| Accept-Encoding | gzip, deflate, br |
| Connection | keep-alive |
| auth-token | eyJhbGciOiJIUzI1NilsInR5cCl6lkpXVCJ9eyJfaWQi... |

The response status is **400 Bad Request**, time **44 ms**, size **437 B**. The body of the response contains the message **1 Invalid Token**.

- Access by username via correct token

The screenshot shows a Postman request for `TP08 EX2 / getUserById` with a `GET` method and URL `http://localhost:3001/user/me`. The **Headers** tab is selected, containing the following fields:

| Header | Value |
|-----------------|---|
| Accept | * |
| Accept-Encoding | gzip, deflate, br |
| Connection | keep-alive |
| auth-token | eyJhbGciOiJIUzI1NilsInR5cCl6lkpXVCJ9eyJfaWQi... |

The response status is **200 OK**, time **40 ms**, size **588 B**. The body of the response is displayed in **Pretty** format:

```

1 {
2   "_id": "647fede64fc980a9dfb54cd",
3   "username": "admin123",
4   "email": "admin@gmail.com",
5   "password": "$2b$10$.u64mtazV2fnpBdbX033LOIDY7372oApBzQmDXUpbcXuZiE./GTu",
6   "__v": 0
7 }

```

- When update user with invalid email format:

The screenshot shows a POST request to `http://localhost:3001/user/update-user`. The request body is JSON with the following fields:

```

1   {
2     "firstName": "Tonghy",
3     "lastName": "Yorg",
4     "email": "tonghy@@.com",
5     "username": "tonghy123"
6   }

```

The response status is 400 Bad Request, with the message "email" must be a valid email.

- Update user Successful:

The screenshot shows a POST request to `http://localhost:3001/user/update-user`. The request body is JSON with the following fields:

```

1   {
2     "firstName": "Tonghy",
3     "lastName": "Yorg",
4     "email": "tonghy@gmail.com",
5     "username": "tonghy123"
6   }

```

The response status is 200 OK, with the following JSON data returned:

```

1   {
2     "_id": "647fede64fc0980a9dfb54cd",
3     "username": "tonghy123",
4     "email": "tonghy@gmail.com",
5     "password": "$2b$10$.u64mtazV2fnpBdbX033L0IDY73720oApBzQmDXUpbcXuZiE./GTu",
6     "__v": 0,
7     "firstName": "Tonghy",
8     "lastName": "Yorg"
9   }

```

- When update password successfully:

The screenshot shows a POST request to `http://localhost:3001/user/update-password`. The request body is a JSON object with a single field `"password": "123456789"`. The response status is 200 OK, and the response body is `"status": "Password was updated successfully!"`.

| Body | Cookies (1) | Headers (9) | Test Results |
|---|-------------|-------------|--|
| <code>"status": "Password was updated successfully!"</code> | | | Status: 200 OK Time: 34 ms Size: 476 B Save as Example |

- Delete User
- Data before delete:

The screenshot shows a list of 10 user documents in MongoDB Compass. Each document contains fields: _id, username, email, firstName, lastName, and password. The password field is heavily redacted.

| Key | Value | Type |
|-------------------------------|---|----------|
| _id | <code>647fee424fc980a9dfb54d2</code> | ObjectId |
| username | admin | String |
| email | admin@gmail.com | String |
| firstName | admin | String |
| lastName | admin | String |
| password | <code>\$2b\$10\$wmylwVQ6K4hxmtgRSpAi.VNJ61wVEMN1tw2xUJkAP00UQyB29bUu</code> | String |
| __v | 0 | Int32 |
| (2) 647fede64fc980a9dfb54cd | { email : "tonghy@gmail.com", firstName : "Tonghy", lastName : "Yorng" } (7 fields) | Document |
| (3) 647febdb34fc980a9dfb54bf | { email : "tonghy999@gmail.com" } (5 fields) | Document |
| (4) 6472a2836c943499b4b542a8 | { email : "sokdara1234@gmail.com" } (9 fields) | Document |
| (5) 6472a017aabba63e02e1a7cb | { email : "chanvotey@gmail.com" } (9 fields) | Document |
| (6) 6471c90e1ace404c0a8b7ce6 | { email : "SokVatey@gmail.com" } (9 fields) | Document |
| (7) 6471926a5819ba802ce92d8d | { email : "channararoth@gmail.com" } (9 fields) | Document |
| (8) 64718dcf5819ba802ce92d58 | { email : "Guest@gmail.com" } (9 fields) | Document |
| (9) 64718aa25819ba802ce92d3b | { email : "admin@gmail.com" } (9 fields) | Document |
| (10) 64717f1a5819ba802ce92cf9 | { email : "yorngtonghy@gmail.com" } (9 fields) | Document |

- When delete user does not exist in database.

The screenshot shows a POST request to `http://localhost:3001/user/delete-user`. The request body contains the JSON object `{"id": "6471926a5819ba802ce92d8d"}`. The response status is 404 Not Found, with the message `"error": "User not found"`.

- Delete user successfully.

The screenshot shows a POST request to `http://localhost:3001/user/delete-user`. The request body contains the JSON object `{"id": "647ea1e1e4956a5e2c5a1531"}`. The response status is 200 OK, with the message `"status": "User was deleted successfully!"`.