



Institute of Technology of Cambodia



Department of Information and
Communication Engineering

Assignment TP10

TP10: VueJS, NodeJS Authentication (Continue)

Group: I4 GIC(C)

Lecturers

HOK TIN

Student's Name:

ROTHA DAPRAVITH

Subjects

TP Internet-Programming II(IP)

ID

e20190915

Academic Year: 2022 ~ 2023

Table of Contents

GitHub's Repository: https://github.com/Dapravith/TP_Internet_Programming/tree/TP10.....	1
EX1: Integrate the previous authentication APIs with VueJS.....	1
EX2: Continue implementing the authentication and user APIs	3

GitHub's Repository: https://github.com/Dapravith/TP_Internet_Programming/tree/TP10

EX1: Integrate the previous authentication APIs with VueJS.

- Result of Exercise1:
- Login

You did it!

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

Login | Register

Username
dapravith235

Password
.....

Login

Forget password?

- Register

Sign up

Please fill in this form to create an account.

Email
dapravith@gmail.com

Username
dapravith235

First name
dapravith

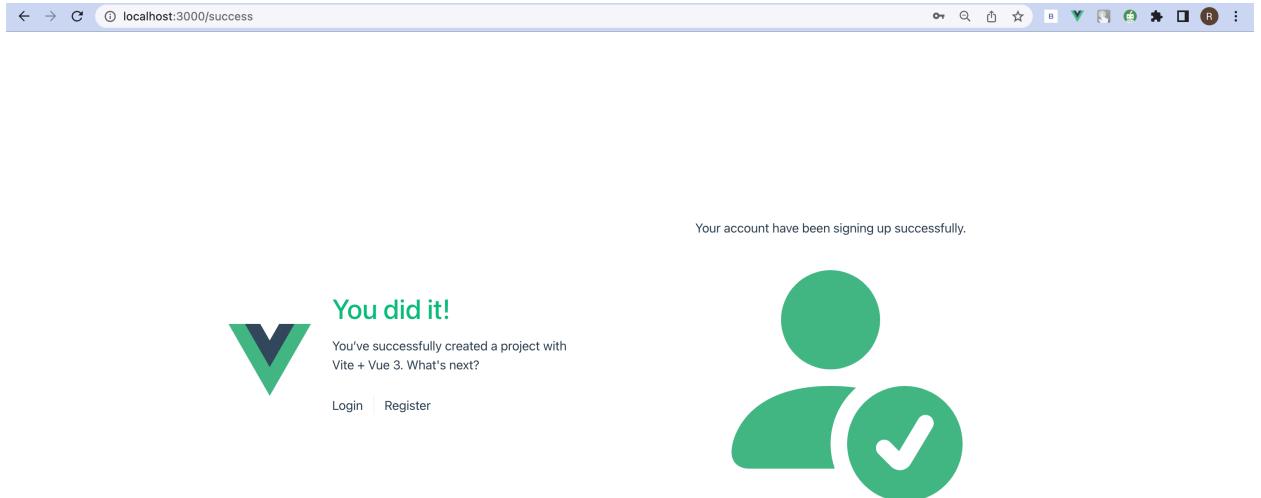
Last name
rotha

Password
.....

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

Sign up

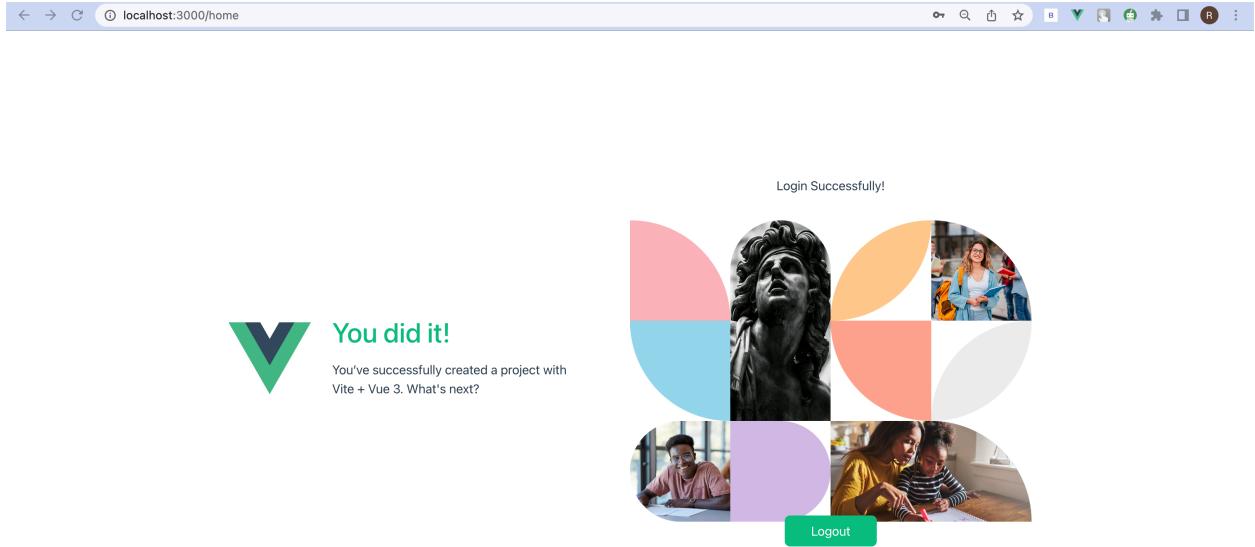
- Register account sucessfully



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

Key	Value	Type
» (1) 647dff4241e4142c226535f1	{ email : "virak@gmail.com" } (9 fields)	Document
_id	647dff4241e4142c226535f1	ObjectId
username	virak1244	String
updatedAt	6/5/2023, 10:29:06 PM - 8 minutes ago	Date
password	\$2a\$10\$Pj0jq2ZmvCaCqjXyAGZeyWKU/MiBz/phUp92OAgoeagolecWc4G	String
lastname	rotha	String
firstname	virak	String
email	virak@gmail.com	String
createdAt	6/5/2023, 10:29:06 PM - 8 minutes ago	Date
__v	0	Int32

- Home-page



EX2: Continue implementing the authentication and user APIs

- Result of Ex02:

- login user and get token:

POST /user/login

Params: none

Headers: Content-Type: application/json

Body (JSON)

```

1   {
2     "username": "dapravith",
3     "password": "12345"
4   }

```

Status: 200 OK Time: 16 ms Size: 779 B

- Register

The screenshot shows a Postman collection named 'TP10_IP' with a 'SignUp' request highlighted. The URL is `http://localhost:3001/user/register`. The 'Body' tab is selected, showing a JSON payload:

```
1
2
3   "username" : "dapravith235",
4   "email" : "dapravith5747@gmail.com",
5   "password": "12345"
6
7
```

The response status is 200 OK, with a user ID returned in the body:

```
1
2   "user": "647e960b8ae3548a6a10e634"
3
```

- Register already existed

The screenshot shows a Postman collection named 'TP10_IP' with a 'SignUp' request highlighted. The URL is `http://localhost:3001/user/register`. The 'Body' tab is selected, showing the same JSON payload as the previous screenshot.

The response status is 400 Bad Request, with an error message in the body:

```
1   Username is already existed!
```

- When no token

The screenshot shows a Postman collection named 'TP10_IP' with a 'GET Get User by ID' request. The 'Headers' tab is selected, showing the following configuration:

Key	Value	Description	... Bulk Edit	Presets
Cookie	access_token=s%3Al5ulMajTO7yZTMjOM8ltoOpNLI4...			
Postman-Token	<calculated when request is sent>			
Host	<calculated when request is sent>			
User-Agent	PostmanRuntime/7.32.2			
Accept	*/*			

The 'Test Results' tab shows a status of 401 Unauthorized with the message "Access Denied!".

- Invalid or wrong token

The screenshot shows the same Postman collection and request setup as the previous screenshot, but with a different header configuration. The 'Headers' tab is selected, showing the following configuration:

Key	Value	Description	... Bulk Edit	Presets
Host	<calculated when request is sent>			
User-Agent	PostmanRuntime/7.32.2			
Accept	*/*			
Accept-Encoding	gzip, deflate, br			
Connection	keep-alive			
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2...			

The 'Test Results' tab shows a status of 400 Bad Request with the message "Invalid Token".

- Access by username via correct token

The screenshot shows a Postman collection named 'TP10_IP' with various endpoints. The current request is a GET to `http://localhost:3001/user/me`. The Headers tab is selected, showing the following configuration:

Key	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.32.2
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJfaWQiOiI2...

The Body tab shows the response in Pretty JSON format:

```

1
2   "_id": "647e88bc7865ac136640631",
3   "username": "dapravith",
4   "firstname": "dapravith",
5   "lastname": "rotha",
6   "email": "dapravith@gmail.com",
7   "password": "$2a$10$Vp6RgHymzONj2piUy.HACOIJqfV34yjG19Evyp/8iPEjutsCI0In.",
8   "createdAt": "2023-06-06T01:15:40.292Z",
9   "updatedAt": "2023-06-06T01:15:40.292Z",
10
11   "__v": 0

```

The status bar at the bottom indicates: Status: 200 OK Time: 21 ms Size: 719 B.

- when update user with invalid email format:

The screenshot shows a Postman collection named 'TP10_IP' with various endpoints. The current request is a POST to `http://localhost:3001/user/update-user`. The Headers tab is selected, showing the following configuration:

Key	Value
Content-Type	application/json

The Body tab shows the request body in JSON format:

```

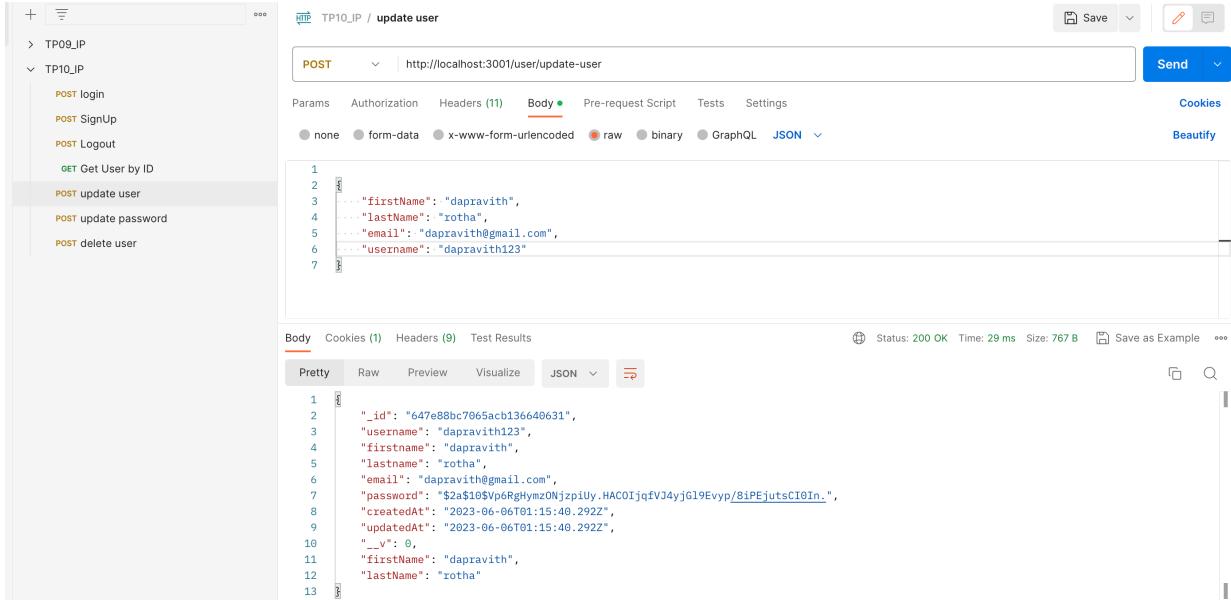
1
2 {
3   ...."firstName": "dapravith",
4   ...."lastName": "rotha",
5   ...."email": "dapravith@com",
6   ...."username": "dapravith123"
7 }

```

The status bar at the bottom indicates: Status: 400 Bad Request Time: 16 ms Size: 460 B.

The response message is: "email" must be a valid email.

- update user Successful:



The screenshot shows a Postman collection named 'TP10_IP' with several requests. The current request is a POST to 'http://localhost:3001/user/update-user'. The body is set to 'JSON' and contains the following data:

```

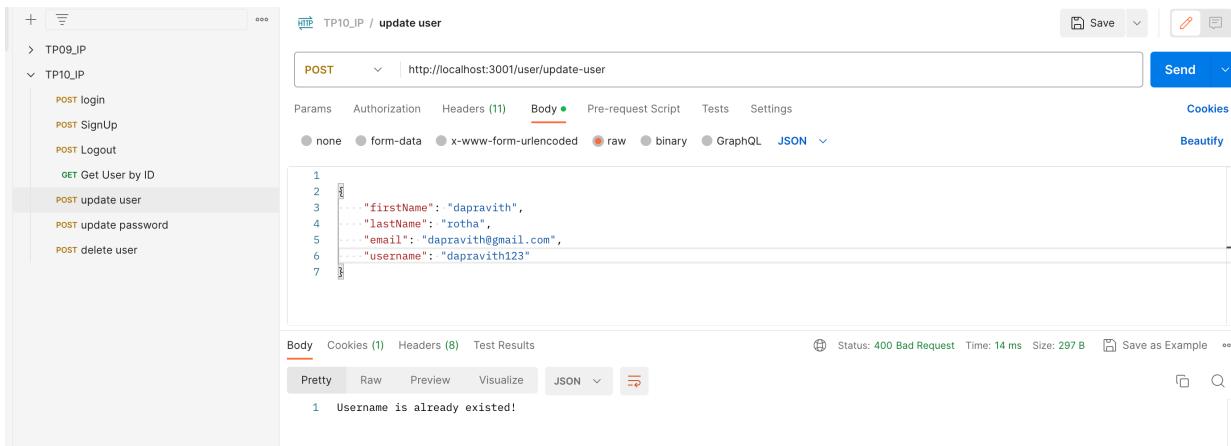
1
2
3   ...
4   ...
5   ...
6   ...
7
  
```

The response status is 200 OK, with a response body containing a user document:

```

1
2   "_id": "647e88bc7065ac136640631",
3   "username": "dapravith123",
4   "firstname": "dapravith",
5   "lastname": "rotha",
6   "email": "dapravith@gmail.com",
7   "password": "52a$105Vp6RgHymzONjzpiUY.HACOIJqfVJ4yjG19Evyp/B1PEjutsCI0In.",
8   "createdAt": "2023-06-06T01:15:40.292Z",
9   "updatedAt": "2023-06-06T01:15:40.292Z",
10  ...
11  ...
12  ...
13
  
```

- when update user with exist username account:



The screenshot shows the same Postman collection and request setup as the previous one, but the response status is 400 Bad Request. The response body is a single line:

```

1   Username is already existed!
  
```

- Invalid token for update password:

The screenshot shows the Postman application interface. On the left, there's a sidebar with project and collection navigation. The main area shows a POST request to `http://localhost:3001/user/update-password`. The Body tab is selected, showing a JSON payload with a single field `"password": "123456"`. The response at the bottom indicates a `400 Bad Request` status with the message `1 Invalid Token`.

- when update password successful:

The screenshot shows the Postman application interface. The setup is identical to the previous one, but the response at the bottom now indicates a `200 OK` status with the message `1 "status": "Password was updated successfully!"`.

- Delete User
- data before delete:

Key	Value	Type
647e87266e0d341a5c62c063	{ email : "dapravith@gmail.com" } (9 fields)	Document
_id	647e87266e0d341a5c62c063	ObjectId
username	dapravith	String
updatedAt	6/6/2023, 8:08:54 AM - an hour ago	Date
password	\$2a\$10\$BPJB1QDKoCslnJDxxieOfODkD7mJTBZuWp9PTBeQipnCF6FHMoPRq	String
lastname	rotha	String
firstname	dapravith	String
email	dapravith@gmail.com	String
createdAt	6/6/2023, 8:08:54 AM - an hour ago	Date
_v	0	Int32
647e86316e0d341a5c62c058	{ email : "virak@gmail.com" } (9 fields)	Document
_id	647e86316e0d341a5c62c058	ObjectId
username	virak124456	String
updatedAt	6/6/2023, 8:04:49 AM - an hour ago	Date
password	\$2a\$10\$AR72mBm68TELsFthuHWZOaVny/sBkz6SS361mJC0HDpNlpGKud2	String
lastname	rotha	String
firstname	virak	String
email	virak@gmail.com	String
createdAt	6/6/2023, 8:04:49 AM - an hour ago	Date
_v	0	Int32

- when delete user does not exist in database:

The screenshot shows the Postman interface with the following details:

- Collection:** TP10_IP
- Request:**
 - Method:** POST
 - URL:** http://localhost:3001/user/delete-user
 - Body:** JSON (raw) - id: "647e88bc7065acb136640632"
- Response:**
 - Status: 404 Not Found
 - Time: 9 ms
 - Size: 300 B
 - Body (Pretty): "error": "User not found"

- Delete user successfully.

TP10_IP / delete user

POST | http://localhost:3001/user/delete-user

Body (1) Headers (0) Test Results

```

1
2   "id": "647e960b8ae3548a6a10e634"
3
4
5

```

Status: 200 OK Time: 44 ms Size: 472 B Save as Example

- data after delete:

Key	Value	Type
1 (1) 647e93c018ef557bd0fc4c06	{ email : "test1@gmail.com", firstName : "dara", lastName : "vith" } (7 fields)	Document
_id	647e93c018ef557bd0fc4c06	ObjectId
username	test1	String
password	\$2b\$10\$WvTI7gxdsJeCYZq1PHpjXuANEVPWACUxAx2Cy5k9zmU7pHIA6Ed2	String
lastName	vith	String
firstName	dara	String
email	test1@gmail.com	String
__v	0	Int32
D (2) 647dff4241e4142c226535f1	{ email : "virak@gmail.com" } (9 fields)	Document
D (3) 647b43f51aa7ca3b36ad45ad	{ email : "dapravithrotha@gmail.com" } (9 fields)	Document
D (4) 6472a2836c943499b4b542a8	{ email : "sokdara1234@gmail.com" } (9 fields)	Document
D (5) 6472a017aabba63e02e1a7cb	{ email : "chavotey@gmail.com" } (9 fields)	Document
D (6) 6471c90e1ace404c0a8b7ce6	{ email : "SokVatey@gmail.com" } (9 fields)	Document
D (7) 6471926a5819ba802ce92d8d	{ email : "channararoth@gmail.com" } (9 fields)	Document
D (8) 64718dcf5819ba802ce92d58	{ email : "Guest@gmail.com" } (9 fields)	Document
D (9) 64718aa25819ba802ce92d3b	{ email : "admin@gmail.com" } (9 fields)	Document

- data after delete and display the result:

TP10_IP / delete user

POST | http://localhost:3001/user/delete-user

Body (1) Headers (0) Test Results

```

1
2   "id": "647e88bc7065acb136640632"
3

```

Status: 404 Not Found Time: 9 ms Size: 300 B Save as Example