



Institute of Technology of Cambodia

Departments of Information and Communication

Engineering



TP-07: JavaScript (VueJS, NodeJS)

TP: Internet Programming

Professor: HOK TIN

Student: SDEUNG VIRAKNON

Group: I4-GIC-C

ID: e20190961

2023 – 2024

Contents

TP09.1: Mongoose	3
TP09.2: NodeJS (Authentication Implementation)	4
TP09.3: NodeJS (Token, Cookie).....	5

TP09.1: Mongoose

For the previous authentication APIs, replace your user.json storage by a NoSql database, mongoose

```
1 db.users.find({})
2 .projection({})
3 .sort({_id:-1})
4 .limit(100)
```

Key	Value	Type
(1) 6471926a5819ba802ce92d8d	{ email : "channararothe@gmail.com" } (9 fields)	Document
(2) 64718dcf5819ba802ce92d58	{ email : "Guest@gmail.com" } (8 fields)	Document
(3) 64718aa25819ba802ce92d3b	{ email : "admin@gmail.com" } (9 fields)	Document
(4) 64717f1a5819ba802ce92cf9	{ email : "dapravithrotha@gmail.com" } (9 fields)	Document
_id	64717f1a5819ba802ce92cf9	ObjectId
username	Dapravith12345	String
updatedAt	5/27/2023, 10:55:06 AM - a day ago	Date
password	\$2a\$10\$jTWktZf2qyCATwmU2VGu93c1Dvb/BuWiqvYybc0LxPJU2Coy.Fy	String
lastname	Dapravith	String
firstname	Dapravith	String
email	dapravithrotha@gmail.com	String
createdAt	5/27/2023, 10:55:06 AM - a day ago	Date
__v	0	Int32
(5) 6471c90e1ace404c0a8b7ce6	{ email : "SokVatey@gmail.com" } (9 fields)	Document

- Find specific username

```
1 db.users.find({username: "Dapravith12345"})
2 .projection({})
3 .sort({_id:-1})
4 .limit(100)
```

Key	Value	Type
(1) 64717f1a5819ba802ce92cf9	{ email : "dapravithrotha@gmail.com" } (9 fields)	Document
_id	64717f1a5819ba802ce92cf9	ObjectId
username	Dapravith12345	String
updatedAt	5/27/2023, 10:55:06 AM - a day ago	Date
password	\$2a\$10\$jTWktZf2qyCATwmU2VGu93c1Dvb/BuWiqvYybc0LxPJU2Coy.Fy	String
lastname	Dapravith	String
firstname	Dapravith	String
email	dapravithrotha@gmail.com	String
createdAt	5/27/2023, 10:55:06 AM - a day ago	Date
__v	0	Int32

- List data in MongoDB

TP09_IP.users

7 DOCUMENTS 3 INDEXES















Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 7 of 7

users												
	_id	ObjectId	username	String	firstname	String	lastname	String	email	String	password	Str:
1		ObjectId('6471926a5819ba802...	"Channararoth"		"Channararoth"		"Channararoth"		"channararoth@gmail.com"		"\$2a\$10\$jtHw/	 
2		ObjectId('64718dcf5819ba802...	"Guest"		"Guest"		"Guest"		"Guest@gmail.com"		"\$2a\$10\$jtHw/	 
3		ObjectId('64718aa25819ba802...	"Admin"		"Admin"		"Admin"		"admin@gmail.com"		"\$2a\$10\$jtHw/	 
4		ObjectId('64717f1a5819ba802...	"Dapravith12345"		"Dapravith"		"Dapravith"		"dapravithrotha@gmail.com"		"\$2a\$10\$jtHw/	 
5		ObjectId('6471c90e1ace404c0...	"SokVatey3899"		"SokVatey"		"SokVatey"		"SokVatey@gmail.com"		"\$2a\$10\$jtHw/	 
6		ObjectId('6472a017aabb63e0...	"Votey"		"Votey"		"Chan"		"chanvotey@gmail.com"		"\$2a\$10\$jd58jv	 
7		ObjectId('6472a2836c943499b...	"SokDara"		"Dara"		"Sok"		"sokdara1234@gmail.com"		"\$2a\$10\$NIXgt	 

TP09.2: NodeJS (Authentication Implementation)

Ex2: Implement the authentication with the password encryption and request validation middleware.

• Result of Ex02:

- Encrypted user accounts

Key	Value	Type
(1) 6471926a5819ba802ce92d8d	{ email : "channararothe@gmail.com" } (9 fields)	Document
(2) 64718dcf5819ba802ce92d58	{ email : "Guest@gmail.com" } (8 fields)	Document
(3) 64718aa25819ba802ce92d3b	{ email : "admin@gmail.com" } (9 fields)	Document
(4) 64717f1a5819ba802ce92cf9	{ email : "dapravithrotha@gmail.com" } (9 fields)	Document
_id	64717f1a5819ba802ce92cf9	ObjectId
username	Dapravith12345	String
updatedAt	5/27/2023, 10:55:06 AM - a day ago	Date
password	\$2a\$10\$jtHw/ktZf2qyCATwmU2VGu93c1Dvb/BuWiqvYybc0LxPJU2Coy.Fy	String
lastname	Dapravith	String
firstname	Dapravith	String
email	dapravithrotha@gmail.com	String
createdAt	5/27/2023, 10:55:06 AM - a day ago	Date
_v	0	Int32
(5) 6471c90e1ace404c0a8b7ce6	{ email : "SokVatey@gmail.com" } (9 fields)	Document

- Request valid Email

POST http://localhost:3001/login

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

Body Cookies Headers (9) Test Results

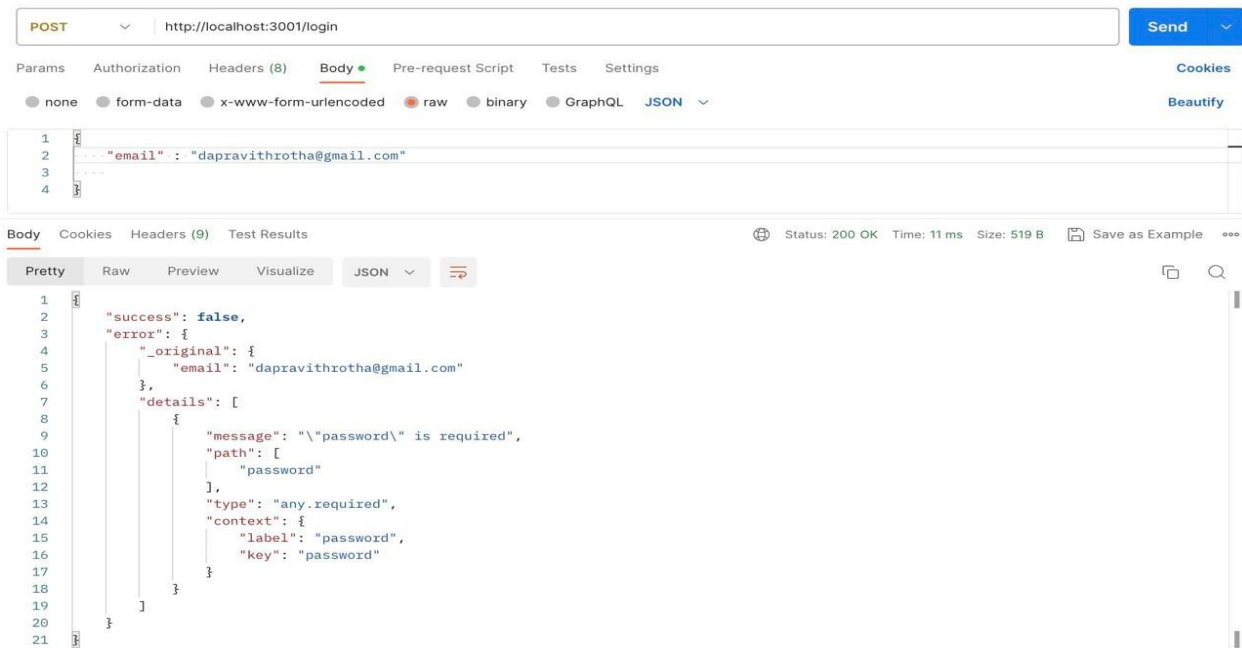
Status: 200 OK Time: 8 ms Size: 598 B Save as Example

```

1 {
2   "success": false,
3   "error": {
4     "original": {
5       "email": "dapravithrotha@gmail.com",
6       "password": "123456"
7     },
8     "details": [
9       {
10        "message": "\"email\" must be a valid email",
11        "path": [
12          "email"
13        ],
14        "type": "string.email",
15        "context": {
16          "value": "dapravithrotha@gmail.com",
17          "invalids": [
18            "dapravithrotha@gmail.com"
19          ],
20          "label": "email",
21          "key": "email"
22        }
23      }
24    ]
25  }
26 }

```

- Request valid Password



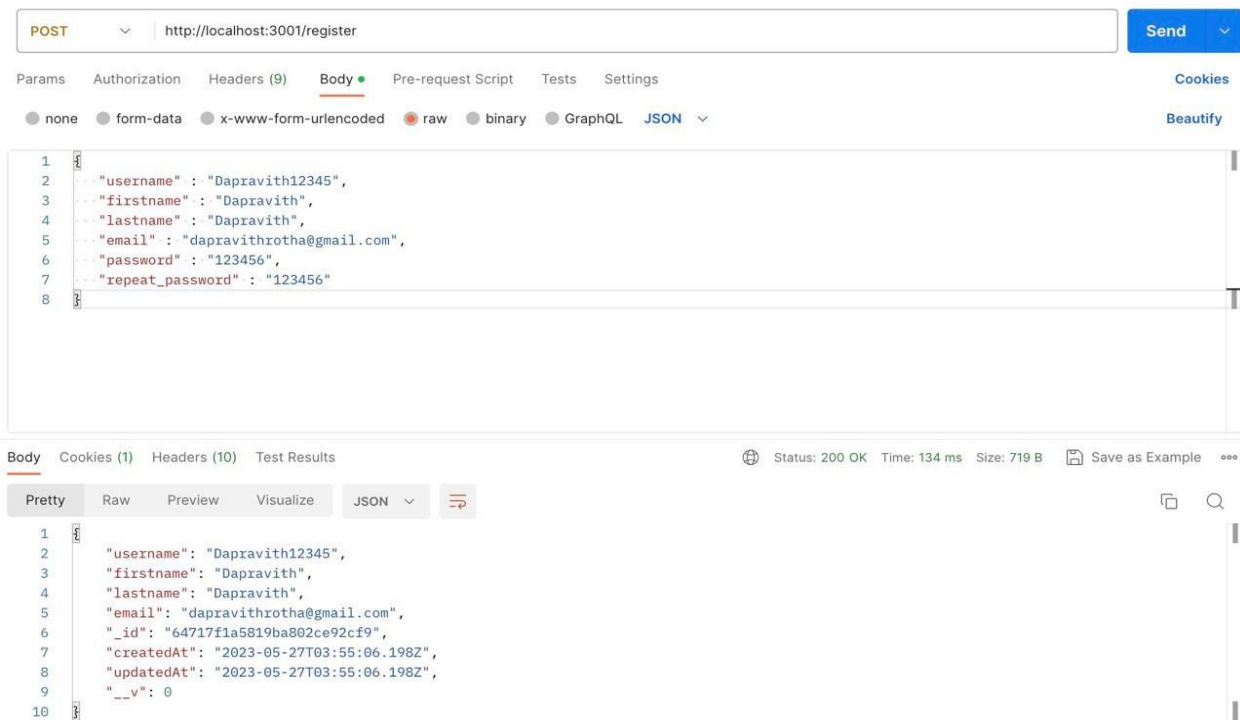
TP09.3: NodeJS (Token, Cookie)

Ex3: Create a token of an authenticated user and store it in the cookie for ensuring the authorized user.

The token validation method must be a middleware function.

Result of Ex3:

-Register User Account



The screenshot shows a REST client interface with a POST request to `http://localhost:3001/login`. The request body is a JSON object: `{ "email": "dapravithrotha@gmail.com", "password": "123456" }`. The response status is 200 OK, with a time of 6 ms and a size of 534 B. The response body is a JSON object: `{ "success": false, "error": "You already signed in-" }`.

```
POST http://localhost:3001/login

{
  "email": "dapravithrotha@gmail.com",
  "password": "123456"
}
```

Status: 200 OK Time: 6 ms Size: 534 B Save as Example

```
{
  "success": false,
  "error": "You already signed in-"
}
```

- Invalid login

The screenshot shows a REST client interface with a POST request to `http://localhost:3001/login`. The request body is a JSON object: `{ "email": "dapravithrotha@gmail.com", "password": "12345646" }`. The response status is 200 OK, with a time of 119 ms and a size of 547 B. The response body is a JSON object: `{ "success": false, "err": "Ther user's information is incorrect-" }`.

```
POST http://localhost:3001/login

{
  "email": "dapravithrotha@gmail.com",
  "password": "12345646"
}
```

Status: 200 OK Time: 119 ms Size: 547 B Save as Example

```
{
  "success": false,
  "err": "Ther user's information is incorrect-"
}
```

- Logout

The screenshot shows a REST client interface with a POST request to `http://localhost:3001/logout`. The request body is a JSON object: `{ "email": "dapravithrotha@gmail.com", "password": "123456" }`. The response status is 200 OK, with a time of 3 ms and a size of 317 B. The response body is a JSON object: `{ "success": true }`.

```
POST http://localhost:3001/logout

{
  "email": "dapravithrotha@gmail.com",
  "password": "123456"
}
```

Status: 200 OK Time: 3 ms Size: 317 B Save as Example

```
{
  "success": true
}
```

- Re-Logout

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

- Method:** POST
- URL:** http://localhost:3001/logout
- Body Type:** raw
- Request Body (JSON):**

```
{  "email": "dapravithrotha@gmail.com",  "password": "123456"}
```
- Status:** 200 OK
- Time:** 12 ms
- Size:** 529 B
- Response Body (JSON):**

```
{  "success": false,  "error": "You must sign In="}
```