

# API DOCUMENTATION

## FOR PERFORMING REST API REQUESTS ON LAPTOP

### STORE DATABASE V1.0

University of Belize



Prepared by

**Tadeo Bennett**

Prepared for

**Mr. Lester Pech**

**University of Belize**

**CMPS4191: Advanced Web-  
Technologies**

**October 1, 2023**

# Table of Contents

<a href="#">Introduction</a>	<a href="#">1</a>
<a href="#">Getting Started</a>	<a href="#">1</a>
<a href="#">Endpoints</a>	<a href="#">2</a>
<a href="#">GET Get All Users</a>	<a href="#">2</a>
<a href="#">GET Get Users by ID</a>	<a href="#">2</a>
<a href="#">GET Get Employees</a>	<a href="#">3</a>
<a href="#">GET Get Customers</a>	<a href="#">3</a>
<a href="#">POST Add New User</a>	<a href="#">4</a>
<a href="#">PUT Update User By ID</a>	<a href="#">5</a>
<a href="#">DELETE Delete a User By ID</a>	<a href="#">5</a>
<a href="#">Response Format</a>	<a href="#">6</a>
<a href="#">API Structure</a>	<a href="#">6</a>
<a href="#">HTTP Response Codes Usage</a>	<a href="#">6</a>
<a href="#">Response Codes and Their Meanings</a>	<a href="#">7</a>
<a href="#">Common Errors</a>	<a href="#">7</a>
<a href="#">GET Requests Errors</a>	<a href="#">7</a>
<a href="#">POST Request Errors</a>	<a href="#">7</a>
<a href="#">PUT Request Errors</a>	<a href="#">7</a>
<a href="#">DELETE Request Errors</a>	<a href="#">7</a>
<a href="#">Success Response Codes</a>	<a href="#">7</a>
<a href="#">Support</a>	<a href="#">8</a>
<a href="#">Changelog</a>	<a href="#">8</a>
<a href="#">Appendix</a>	<a href="#">9</a>
<a href="#">GET - Get All Users Example Response</a>	<a href="#">9</a>
<a href="#">GET - Get Users by ID Example Response</a>	<a href="#">10</a>
<a href="#">GET - Get Employees Example Response</a>	<a href="#">10</a>
<a href="#">GET - Get Customers Example Response</a>	<a href="#">11</a>
<a href="#">POST - Add New User Example Response</a>	<a href="#">11</a>
<a href="#">PUT - Update User Record by ID Example Response</a>	<a href="#">12</a>
<a href="#">DELETE - Delete User Record by ID Example Response</a>	<a href="#">12</a>

## Introduction

This API enables developers to perform operations such as GET, POST, PUT, and DELETE on the users of the laptopstore database. Its primary purpose is to enable developers to understand the requests' makeup and perform operations that yield expected results from the database. In addition, any error during the process is communicated to the developer to allow easy troubleshooting. This API is designed for any developer with knowledge of REST APIs, PHP, and MySQL.

These request examples can also be found in the [online documentation](#).

## Getting Started

If permitted, you can [clone the repo](#) from GitHub. In the extras directory, the file laptopstore\_sql\_dump must be used to create the laptopstore database. First, you'll need to create the "laptopstore" database and then run the file's contents in that database. Include the necessary login credentials for your mysql database in the index.php file. If Apache is used, I recommend having the root directory set up as the parent directory of the repo. It should look like the following:

```
root/CMPS4191_adv_web_RESTAPI/...
```

If required, ensure that your port is provided in the URI. It should look like the following:

```
127.0.0.1:8036/CMPS4191_adv_web_RESTAPI/...
```

## Endpoints

### GET Get All Users

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users`

This request returns all information for all users in the laptopstore database.

A successful request for all users will result in an *HTTP 200* status code.

Example Request:

```
curl --location '127.0.0.1/CMPS4191_adv_web_RESTAPI/users'
```

Example Response

[See Appendix Page 9](#)

### GET Get Users by ID

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users/{id}`

This request returns all information for a user with a certain ID in the laptopstore database.

A successful request for a user will result in an *HTTP 200* status code.

Example Request:

```
curl --location '127.0.0.1/CMPS4191_adv_web_RESTAPI/users/3'
```

Example Response

[See Appendix Page 10](#)

## GET Get Employees

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users/employees`

The request returns all information for employees in the laptopstore database.

A successful request for all employees will result in an *HTTP 200* status code.

Example Request:

```
curl --location '127.0.0.1/CMPS4191_adv_web_RESTAPI/users/employees'
```

Example Response

[See Appendix Page 10](#)

## GET Get Customers

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users/customers`

The request returns all information for customers in the laptopstore database.

A successful request for all customers will result in an *HTTP 200* status code.

Example Request:

```
curl --location '127.0.0.1/CMPS4191_adv_web_RESTAPI/users/customers'
```

Example Response

[See Appendix Page 11](#)

## **POST Add New User**

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users`

The request creates a new user in the laptopstore database with the details sent in the request body.

A successful request to create a user using the provided details will result in an *HTTP 200* status code.

Example Request:

```
curl --location '127.0.0.1/CMPS4191_adv_web_RESTAPI/users' \  
--form 'firstname="Mary"' \  
--form 'lastname="Jane"' \  
--form 'age="25"' \  
--form 'email="mary@gmail.com"' \  
--form 'username="MJane"' \  
--form 'member="1"'
```

Example Response

[See Appendix Page 11](#)

## **PUT Update User By ID**

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users/{id}`

The request updates an existing user record using an existing user's ID provided in the URI, and details to update in the request body.

A successful request to update a user using the provided ID will result in an *HTTP 200* status code.

### Example Request

```
curl --location --request PUT '127.0.0.1/CMPS4191_adv_web_RESTAPI/users/5' \  
--form 'firstname="Mary"' \  
--form 'lastname="Jane"' \  
--form 'age="25"' \  
--form 'member="1"' \  
--form 'email="mary@gmail.com"'
```

### Example Response

[See Appendix Page 12](#)

## **DELETE Delete a User By ID**

`http://127.0.0.1/CMPS4191_adv_web_RESTAPI/users/{id}`

The request deletes an existing user record using an existing user's ID provided in the URI. A successful request to delete a user using the provided ID will result in an *HTTP 200* status code.

### Example Request

```
curl --location --request DELETE '127.0.0.1/CMPS4191_adv_web_RESTAPI/users/5'
```

### Example Response

[See Appendix Page 12](#)

# Response Format

## API Structure

**Example Response**

```
json
{
  "rc": 2.1,
  "message": "Success",
  "data": [
    {
      "id": 3,
      "firstname": "Lucas",
      "lastname": "Films",
      "age": 0,
      "address": "San Ignacio",
      "contact_number": "",
      "email": "Lucas",
      "legal_doc": null,
      "degree": "ASSC",
      "status": null
    }
  ]
}
```

"rc" is the **response code**. This code uniquely tracks the status of any request being handled. A negative value indicates an error, and a positive value indicates success.

"message" is a description of the response. Can be referred to as the status of the request.

"data" is an array of objects. These objects are student records returned from the database by a request.

## HTTP Response Codes Usage

Two response codes are used in the current version of the API:

### HTTP Response Code 200

- 200 OK: The request was successful.

### HTTP Response Code 400

- The client's request is malformed.



## Response Codes and Their Meanings

### Common Errors

- 1: Invalid Request. Unknown resource Requested; expecting “users”.
- 2: Unsupported Request Method. Could not find the request method GET, POST, PUT, or DELETE.
- 3: No Database connection; Database connection was lost

### GET Requests Errors

- 11: Default; Invalid GET Request
- 12: Default; Users’ details not found
- 13: STMT Execution Error when getting all users
- 14: Default; User details not found with the ID provided
- 15: STMT Execution Error when getting user with the ID provided
- 16: Default; Employees’ details not found
- 17: STMT Execution Error when getting all employees
- 18: Default; Customers’ details not found
- 19: STMT Execution Error when getting all customers

### POST Request Errors

- 20: Default; Invalid POST Request
- 21: STMT Execution Error when creating a new user

### PUT Request Errors

- 22: Default; Invalid PUT Request
- 23: STMT Execution Error when updating an existing user’s details

### DELETE Request Errors

- 24: Default; Invalid DELETE Request
- 25: STMT Execution Error when deleting a user

### Success Response Codes

- 50: Successful GET request for all users’ details
- 51: Successful GET request for user with the provided ID
- 52: Successful GET request for all employees’ details
- 53: Successful GET request for all customers’ details

54: Successful POST request for creating a new user

55: Successful PUT request for updating a user with provided ID

56: Successful DELETE request for deleting a user with provided ID

100: Test function is working.

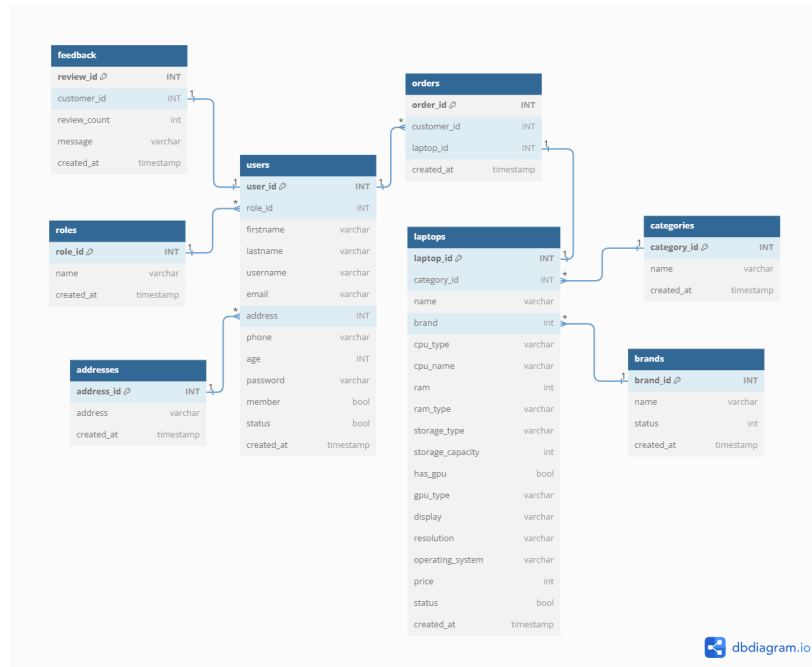
## Support

If there are unhandled errors, kindly email me at [2021154344@ub.edu.bz](mailto:2021154344@ub.edu.bz). Ensure you have proper descriptions of the encountered problem and the events leading up to it. If you can, include pictures to demonstrate further.

## Changelog

API updates, new features, and bug fixes are communicated in the commits of the git repository.

## Appendix



Database Entity Relationship Diagram (ERD)

## GET - Get All Users Example Response

```

1  {
2    "rc": 50,
3    "message": "Success",
4    "data": [
5      {
6        "user_id": 1,
7        "role_id": 1,
8        "firstname": "Tadeo",
9        "lastname": "Bennett",
10       "username": "TBennett",
11       "email": "tadeo@gmail.com",
12       "address": 4,
13       "phone": "343-8371",
14       "age": 21,
15       "password": "Tadeo2002",
16       "member": 0,
17       "status": 1,
18       "created_at": "2023-09-30 17:30:43"
19     },
20     {
21       "user_id": 2,
22       "role_id": 2,
23       "firstname": "William",
24       "lastname": "Locario",
25       "username": "WLocario",
26       "email": "william@gmail.com",
27       "address": 6,
28       "phone": "837-3421",
29       "age": 22,
30       "password": "William2002",
31       "member": 1,
32       "status": 1,
33       "created_at": "2023-09-30 17:30:43"
34     },
35     {
36       "user_id": 3,
37       "role_id": 2,
38       "firstname": "Victor",
39       "lastname": "Castillo",
40       "username": "VCastillo",
41       "email": "victor@gmail.com",
42       "address": 2,
43       "phone": "998-3234",
44       "age": 19,
45       "password": "Victor2002",
46       "member": 0,
47       "status": 1,
48       "created_at": "2023-09-30 17:30:43"
49     }
50   ]
51 }

```

## GET - Get Users by ID Example Response

```
1  {
2    "rc": 51,
3    "message": "Success",
4    "data": [
5      {
6        "user_id": 3,
7        "role_id": 2,
8        "firstname": "Victor",
9        "lastname": "Castillo",
10       "username": "VCastillo",
11       "email": "victor@gmail.com",
12       "address": 2,
13       "phone": "998-3234",
14       "age": 19,
15       "password": "Victor2002",
16       "member": 0,
17       "status": 1,
18       "created_at": "2023-09-30 17:30:43"
19     }
20   ]
21 }
```

## GET - Get Employees Example Response

```
1  {
2    "rc": 52,
3    "message": "Success",
4    "data": [
5      {
6        "user_id": 1,
7        "role_id": 1,
8        "firstname": "Tadeo",
9        "lastname": "Bennett",
10       "username": "TBennett",
11       "email": "tadeo@gmail.com",
12       "address": 4,
13       "phone": "343-8371",
14       "age": 21,
15       "password": "Tadeo2002",
16       "member": 0,
17       "status": 1,
18       "created_at": "2023-09-30 17:30:43"
19     }
20   ]
21 }
```

## GET - Get Customers Example Response

```
1 {
2   "rc": 53,
3   "message": "Success",
4   "data": [
5     {
6       "user_id": 2,
7       "role_id": 2,
8       "firstname": "William",
9       "lastname": "Locario",
10      "username": "WLocario",
11      "email": "william@gmail.com",
12      "address": 6,
13      "phone": "837-3421",
14      "age": 22,
15      "password": "William2002",
16      "member": 1,
17      "status": 1,
18      "created_at": "2023-09-30 17:30:43"
19    },
20    {
21      "user_id": 3,
22      "role_id": 2,
23      "firstname": "Victor",
24      "lastname": "Castillo",
25      "username": "VCastillo",
26      "email": "victor@gmail.com",
27      "address": 2,
28      "phone": "998-3234",
29      "age": 19,
30      "password": "Victor2002",
31      "member": 0,
32      "status": 1,
33      "created_at": "2023-09-30 17:30:43"
34    }
35  ]
36 }
```

## POST - Add New User Example Response

```
1 {
2   "rc": 54,
3   "message": "Successful user creation",
4   "new_user_id": 5
5 }
```

## PUT - Update User Record by ID Example Response

```

1  {
2    "rc": 51,
3    "message": "Success",
4    "data": [
5      {
6        "user_id": 5,
7        "role_id": 2,
8        "firstname": "Luke",
9        "lastname": "Skywalker",
10       "username": "MJane",
11       "email": "luke@gmail.com",
12       "address": null,
13       "phone": "",
14       "age": 35,
15       "password": "password",
16       "member": 0,
17       "status": 1,
18       "created_at": "2023-09-30 23:52:45"
19     }
20   ],
21   "newdata": [
22     {
23       "user_id": 5,
24       "role_id": 2,
25       "firstname": "Mary",
26       "lastname": "Jane",
27       "username": "MJane",
28       "email": "mary@gmail.com",
29       "address": null,
30       "phone": "",
31       "age": 25,
32       "password": "password",
33       "member": 1,
34       "status": 1,
35       "created_at": "2023-10-01 00:26:34"
36     }
37   ]
38 }

```

## DELETE - Delete User Record by ID Example Response

```

1  {
2    "rc": 2,
3    "message": "Success. User deleted with id 5",
4    "data": [
5      {
6        "user_id": 5,
7        "role_id": 2,
8        "firstname": "Mary",
9        "lastname": "Jane",
10       "username": "MJane",
11       "email": "mary@gmail.com",
12       "address": null,
13       "phone": "",
14       "age": 25,
15       "password": "password",
16       "member": 1,
17       "status": 1,
18       "created_at": "2023-10-01 00:26:34"
19     }
20   ]
21 }

```