

Faker

To generate fake data

userId

username

email

password

```
let getUser = () => {  
  return [  
    faker.datatype.uuid(),  
    faker.internet.userName(),  
    faker.internet.email(),  
    faker.internet.password(),  
  ];  
};
```

snehagupta738

@faker-js/faker - npm

npmjs.com/package/@faker-js/faker

Notorious Puppy Memes

ProTe

npm

Search packages

Search

@faker-js/faker TS


8.0.2 • Public • Published 3 months ago

Readme

Code Beta

0 Dependencies

739 Dependents



Faker

Generate massive amounts of fake (but realistic) data for testing and development.

npm v8.0.2 downloads 11.9M/month CI passing codecov 100% chat discord backers 25 sponsors 48

Install

> npm i @faker-js/faker

Repository

github.com/faker-js/faker

Homepage

github.com/faker-js/faker

♥Fund this pack

Weekly Downloads

2,002,456

> SQLCLASS

> OUTLINE

> TIMELINE

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

○ shradhakhapra@Shradhas-MacBook-Air SQLClass % 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: (sqlclass)

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author: █

```
"author": "Shradha Khapra",  
"license": "ISC"  
}
```

Is this OK? (yes)

○ shradhakhapra@Shradhas-MacBook-Air SQLClass % `npm i @faker-js/faker`

EXPLORER

...

{ } package.json

JS index.js

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

JS index.js > createRandomUser

```
1  const { faker } = require('@faker-js/faker');
2
3  let createRandomUser = () => {
4    return {
5      ...{
6        userId: faker.datatype.uuid(),
7        username: faker.internet.userName(),
8        email: faker.internet.email(),
9        avatar: faker.image.avatar(),
10       password: faker.internet.password(),
11       birthdate: faker.date.birthdate(),
12       registeredAt: faker.date.past(),
13     };
14   }
```

snehagupta7385@gmail.com

- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will be removed in a future version. Use faker.string.uuid() instead.
{
 userId: '64a73cfa-fce9-46fc-8b1e-b45d29638fe1',
 username: 'Karen_Reinger3',
 email: 'Ivy_Bailey@yahoo.com',
 avatar: 'https://avatars.githubusercontent.com/u/31723747',
 password: 'hDDNw8pcaahm1yU',
 birthdate: 1999-05-27T15:11:44.347Z,
 registeredAt: 2023-07-27T04:19:52.761Z
}
- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will be removed in a future version. Use faker.string.uuid() instead.
{
 userId: 'e0a62458-83e1-4af0-a5f1-96be0ac1bff6',
 username: 'Mavis97',
 email: 'Russell98@gmail.com',
 avatar: 'https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFf',
 password: '5R_TOZgWLUANahR',
 birthdate: 1961-10-03T20:54:10.975Z,
 registeredAt: 2023-04-12T02:30:19.565Z
}
- shradhakhapra@Shradhas-MacBook-Air SQLClass %

EXPLORER

...

{ } package.json

JS index.js

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

```
JS index.js > [?] getRandomUser > [?] password
1  const { faker } = require("@faker-js/faker");
2
3  let getRandomUser = () => {
4    return {
5      userIdid: faker.datatype.uuid(),
6      username: faker.internet.userName(),
7      email: faker.internet.email(),
8      password: faker.internet.password()
9    };
10 };
11
12 console.log(createRandomUser());
13
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will be removed in a future version. Use faker.string.uuid() instead.
```

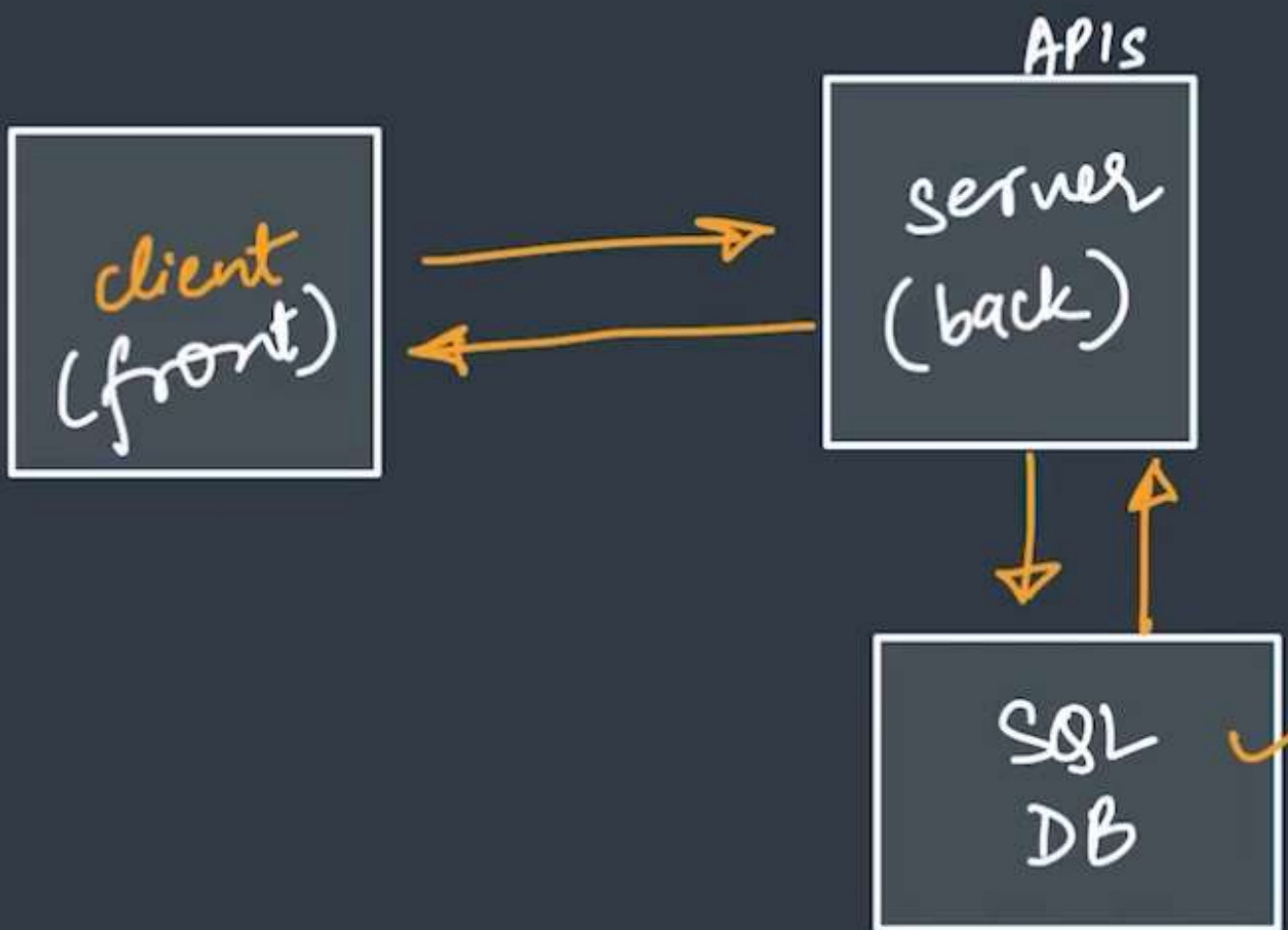
```
{  
  id: '12b4355a-a3fd-4231-bb56-b2bd622b2d9c',  
  username: 'Carmella_Mueller33',  
  email: 'Shyann35@yahoo.com',  
  password: '03DYhh5X0BFz2'  
}
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```


MySQL2 Package

To connect Node with MySQL

```
connection.end(); // to close connection
```





Search packages

Search

mysql2

3.6.0 • Public • Published a month ago

 Readme

 Code Beta

 8 Dependencies

 3,830 Dependents

 157 Versions

MySQL 2

 [Move to Snyk](#)  [v3.6.0](#)  [6.7M/month](#)  [>= 8.0](#)  [passing](#)  [MIT](#)


English | [简体中文](#) | [Português \(BR\)](#)

MySQL client for Node.js with focus on performance. Supports prepared statements, non-utf8 encodings, binary log protocol, compression, ssl **much more**.

Install

```
> npm i mysql2
```

Repository

 github.com/sidorares/node-mysql2

Homepage

 github.com/sidorares/node-mysql2

```
// get the client
const mysql = require('mysql2');

// create the connection to database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  database: 'test'
});

// simple query
connection.query(
  'SELECT * FROM `table` WHERE `name` = "Page" AND `age` > 45',
  function(err, results, fields) {
    console.log(results); // results contains rows returned by server
    console.log(fields); // fields contains extra meta data about results,
  }
);
```

Delta Connection - Warning - not supported

Administration Schemas Query 1

SCHEMAS

Filter objects

- delta_app
 - Tables
 - temp
 - Views
 - Stored Procedures
 - Functions
 - sys

```
1 • CREATE DATABASE delta_app;
2
3 • USE delta_app;
4
5 • CREATE TABLE temp (
6   id INT PRIMARY KEY
7 );
```

Limit to 1000 rows

snehagupta7385@gmail.com

```
1  const { Taker } = require( '@taker-js/taker' );
2  const mysql = require('mysql2');
3
4  const connection = mysql.createConnection({
5    host: 'localhost',
6    user: 'root',
7    database: 'delta_app',
8    passowrd: 'mysql@123'
9  });
```

```
0
1 try {
2     connection.query("SHOW TABLES", (err, result) => {
3         if (err) throw err;
4         console.log(result);
5     });
6 } catch (err) {
7     console.log(err);
8 }
```

19

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
    at Socket.Readable.push (node:internal/streams/readable:228:10) {  
  code: 'ER_ACCESS_DENIED_ERROR',  
  errno: 1045,  
  sqlState: '28000',  
  sqlMessage: "Access denied for user 'root'@'localhost' (using password:  
  sql: undefined,  
  fatal: true  
}
```

```
○ shradhakhapra@Shradhas-MacBook-Air SQLClass %
```


Ln 16, Col 16, Spaces: 4, UTF-8, LF, {} Java

Using SQL from CLI

/usr/local/mysql/bin/mysql -u root -p ✓

Create schema.sql

source schema.sql // in CLI

JS index.js >  connection.query("SHOW TABLES") callback


```
9   });
10
11   try {
12     connection.query("SHOW TABLES", (err, result) => {
13       if (err) throw err;
14       console.log(result);
15     });
16   } catch (err) {
17     console.log(err);
18   }
19
20   let getRandomUser = () => {
21     return {
22       id: faker.datatype.uuid(),
23       username: faker.internet.userName(),
24       email: faker.internet.email().
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

○ shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }]



```

9   });
10
11   try {
12     connection.query("SHOW TABLES", (err, result) => {
13       if (err) throw err;
14       console.log(result);
15     });
16   } catch (err) {
17     console.log(err);
18   }
19   
20   connection.end();
21
22   let getRandomUser = () => {
23     return {
24       id: faker.datatype.uuid(),

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

- ✖ shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }]
^C
- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }]
- shradhakhapra@Shradhas-MacBook-Air SQLClass %

Using SQL from CLI

```
/usr/local/mysql/bin/mysql -u root -p
```

Create schema.sql

```
source schema.sql // in CLI
```

EXPLORER

...

{ } package.json

JS index.js

schema.sql X

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

schema.sql

1 SHOW TABLES;

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> source schema.sql;

Tables_in_delta_app
temp

CREATE Table user

```
CREATE TABLE user (  
    userId VARCHAR(50) PRIMARY KEY,  
    username VARCHAR(50) UNIQUE,  
    email VARCHAR(50) UNIQUE NOT NULL,  
    password VARCHAR(50) NOT NULL  
);
```

EXPLORER

...

JS index.js

schema.sql X

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

schema.sql

```
1 CREATE TABLE user (  
2     id VARCHAR(50) PRIMARY KEY,  
3     username VARCHAR(50) UNIQUE,  
4     email VARCHAR(50) UNIQUE NOT NULL,  
5     password VARCHAR(50) NOT NULL  
6 );
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

Database changed

mysql> source schema.sql;

+-----+

| Tables_in_delta_app |

+-----+

| temp |

+-----+

1 row in set (0.00 sec)

mysql> source schema.sql;

Query OK, 0 rows affected (0.01 sec)

mysql> █

> OUTLINE

> TIMELINE

EXPLORER

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js

schema.sql

JS index.js > ...

```
10
11 try {
12   connection.query("SHOW TABLES", (err, result) => {
13     if (err) throw err;
14     console.log(result);
15   });
16 } catch (err) {
17   console.log(err);
18 }
19
20 connection.end();
21
22 let getRandomUser = () => {
23   return {
24     id: faker.datatype.uuid(),
25     username: faker.internet.userName(),
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
^C
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' } ]
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```


EXPLORER

...

JS index.js

●

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > ...

```
3
4  const connection = mysql.createConnection({
5    host: "localhost",
6    user: "root",
7    database: "delta_app",
8    password: "mysql@123",
9  });
10
11  try {
12    connection.query("SHOW TABLES", (err, result) => {
13      if (err) throw err;
14      console.log(result);
15      console.log(result.length);
16      console.log(result[0]);
17      console.log(result[1]);
18    });
19  } catch (err) {
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' }]
2
- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' }]
2
{ Tables_in_delta_app: 'temp' }
{ Tables_in_delta_app: 'user' }

```
7     database: "delta_app",
8     password: "mysql@123",
9 });
10
11 let q = "SHOW TABLES";
12
13 try {
14     connection.query(q, (err, result) => {
15         if (err) throw err;
16         console.log(result);
17         console.log(result.length);
18         console.log(result[0]);
19         console.log(result[1]);
20     });
21 }
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

- shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[{ Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' }]
2
{ Tables_in_delta_app: 'temp' }
{ Tables_in_delta_app: 'user' }
○ shradhakhapra@Shradhas-MacBook-Air SQLClass % █

INSERT User

using **placeholders**

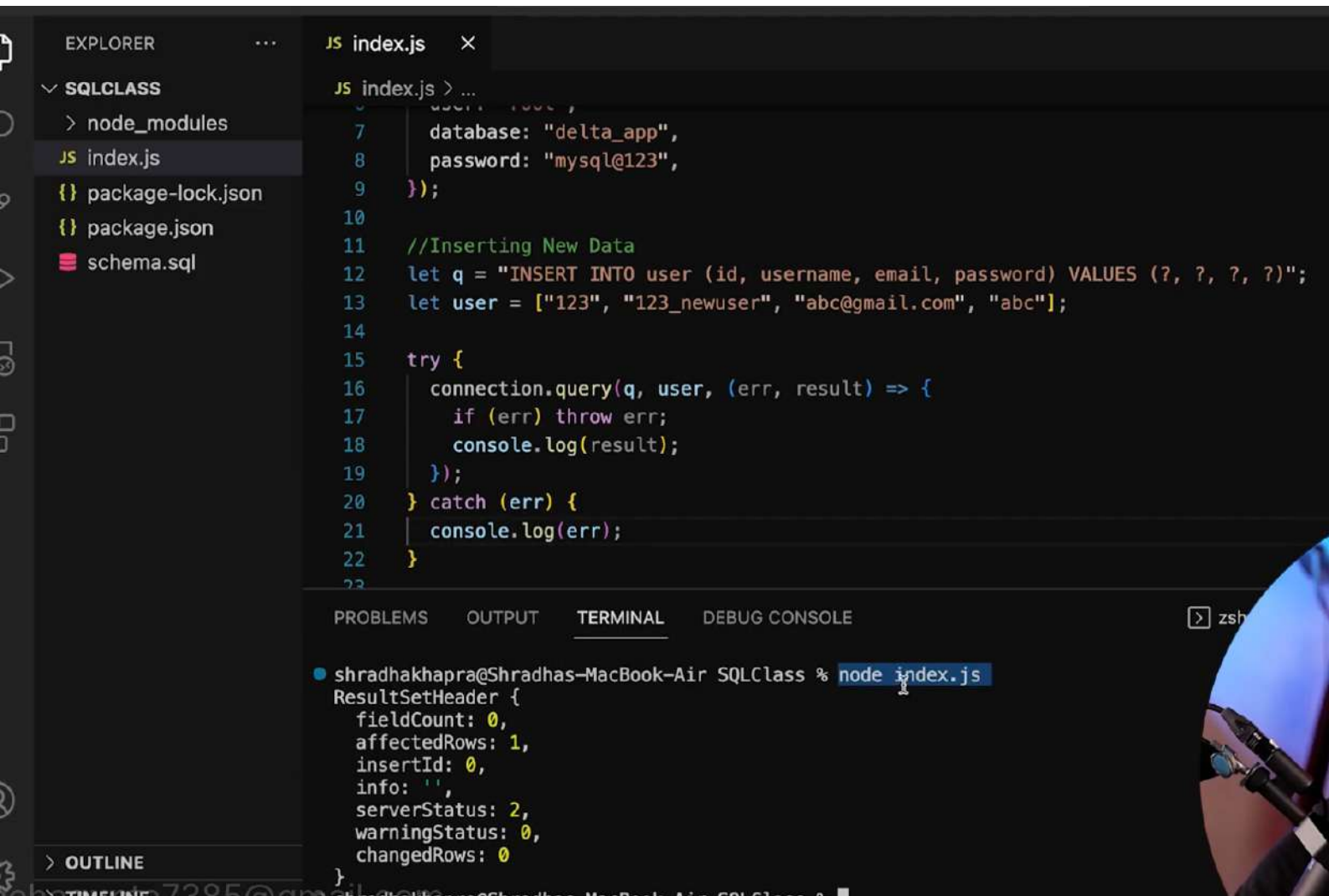
```
let user = ["123@abc2", "random_user2", "random@gmail.com2", "random@123"];

connection.query(
  `INSERT INTO user (userId, username, email, password) VALUES (?, ?, ?, ?)`,
  user,
  function (err, results) {
    if (err) throw err;
    console.log(results);
  }
);
```

```
});

// simple query
connection.query(
  'SELECT * FROM `table` WHERE `name` = "Page" AND `age` > 45',
  function(err, results, fields) {
    console.log(results); // results contains rows returned by server
    console.log(fields); // fields contains extra meta data about results,
  }
);

// with placeholder
connection.query(
  'SELECT * FROM `table` WHERE `name` = ? AND `age` > ?',
  ['Page', 45],
  function(err, results) {
    console.log(results);
  }
);
```



2 rows in set (0.00 sec)

mysql> SELECT * FROM user;

id	username	email	password
123	123_newuser	abc@gmail.com	abc

1 row in set (0.00 sec)

mysql>

SQLCLASS
> node_modules
js index.js
{ } package-lock.json
{ } package.json
schema.sql

```
9  });  
10  
11  //Inserting New Data  
12  let q = "INSERT INTO user (id, username, email, password) VALUES (?, ?, ?, ?)";  
13  let users = [  
14    ["123b", "123_newuserb", "abc@gmail.comb", "abcb"],  
15    ["123c", "123_newuserc", "abc@gmail.comc", "abcc"],  
16  ];  
17  
18  try {  
19    connection.query(q, users, (err, result) => {  
20      if (err) throw err;  
21      console.log(result);  
22    });  
23  } catch (err) {  
24    console.log(err);  
25  }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

zsh

2 rows in set (0.00 sec)

mysql> SELECT * FROM user;

id	username	email	password
123	123_newuser	abc@gmail.com	abc

1 row in set (0.00 sec)

mysql>

> OUTLINE

> TIMELINE

snehagup

EXPLORER

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > [q]

9 };

10

11 💡 Inserting New Data

12 let q = `INSERT INTO user (id, username, email, password) VALUES ?`;

13 let users = [

14 ["123b", "123_newuserb", "abc@gmail.comb", "abcb"],

15 ["123c", "123_newuserc", "abc@gmail.comc", "abcc"],

16];

17

18 try {

19 connection.query(q, [users], (err, result) => {

20 if (err) throw err;

21 console.log(result);

22 });

23 } catch (err) {

24 console.log(err);

25 }

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

> zsh

2 rows in set (0.00 sec)

mysql> SELECT * FROM user;

id	username	email	password
123	123_newuser	abc@gmail.com	abc

1 row in set (0.00 sec)

mysql>

> OUTLINE

> TIMELINE

snehagupta

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
● shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 2,
  insertId: 0,
  info: 'Records: 2 Duplicates: 0 Warnings: 0',
  serverStatus: 2,
  warningStatus: 0,
  changedRows: 0
}
○ shradhakhapra@Shradhas-MacBook-Air SQLClass %
```

Ln 15, Col 25 (29 selected) Spaces: 4 UTF-8 LF {}

```
mysql> SELECT * FROM user;
```

id	username	email	password
123	123_newuser	abc@gmail.com	abc
123b	123_newuserb	abc@gmail.comb	abcb
123c	123_newuserc	abc@gmail.comc	abcc

```
3 rows in set (0.00 sec)
```

```
mysql>
```

INSERT in Bulk

using **faker**

```
let data = [];  
for (let i = 1; i <= 100; i++) {  
  data.push(getUser());  
}  
  
console.log(data);  
let q = `INSERT INTO user (userId, username, email, password) VALUES ?`;  
  
connection.query(q, [data], function (err, results) {  
  if (err) throw err;  
  console.log(results);  
});
```

```
28
29 let getRandomUser = () => {
30   return [
31     faker.datatype.uuid(),
32     faker.internet.userName(),
33     faker.internet.email(),
34     faker.internet.password(),
35   ];
36 };
37
```

```
//Inserting New Data
```

```
let q = "INSERT INTO user (id, username, email, password) VALUES ?";
```

```
let data = [];
```

```
for (let i = 1; i <= 100; i++) {
```

```
  console.log(getRandomUser());
```

```
}
```

```
20 //Inserting New Data
21 let q = "INSERT INTO user (id, username, email, password) VALUES ?";
22
23 let data = [];
24 for (let i = 1; i <= 100; i++) {
25     data.push(getRandomUser());
26 }
27
```

EXPLORER

...

JS index.js

●

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > ...

```
14   faker.internet.userName(),
15   faker.internet.email(),
16   faker.internet.password(),
17   ];
18 };
19
20 //Inserting New Data
21 let q = "INSERT INTO user (id, username, email, password) VALUES ?";
22
23 let data = [];
24 for (let i = 1; i <= 100; i++) {
25   data.push(getRandomUser()); // 100 fake
26 }
27
28 try {
29   connection.query(q, [data], (err, result) => {
30     if (err) throw err;
31     console.log(result);
32   });
33 } catch (err) {
34   console.log(err);
35 }
```

```
[@faker-js/faker]: faker.datatype.uuid() is deprecated s
e use faker.string.uuid() instead.
[@faker-js/faker]: faker.datatype.uuid() is deprecated s
e use faker.string.uuid() instead.
```

```
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 100,
  insertId: 0,
  info: 'Records: 100 Duplicates: 0 Warnings: 0',
  serverStatus: 2,
  warningStatus: 0,
  changedRows: 0
}
```

```
○ shradhaKhapra@Shradhas-MacBook-Air SQLClass %
```


123b	123_newuserb	abc@gmail.comb
123c	123_newuserc	abc@gmail.comc
12f1c618-e24e-4b5b-8624-db04f2a8d14c	Trisha95	Sonia.Bednar-Abernathy99@hotmail.com
151e72cb-1098-42fb-be53-c470e09d7a96	Odie35	Mackenzie_Blanda6@gmail.com
17e1087b-9d73-46a6-ba77-441da71066f0	Ulisess55	Clotilde.Gibson@yahoo.com
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Esther54	Gabe_Buckridge@yahoo.com
1fc0bed1-d300-4893-94eb-9c4ead370d88	Pat74	Mina_Bode37@gmail.com
22eb4210-992f-4e13-91e8-134fbe441a89	Cydne73	Maddison_Beer@yahoo.com
255c4f3d-d08d-4493-be2f-d3bfc431404b	Theo61	Jalen_Feil5@hotmail.com
25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1	Mathew_Fadel55	Christa34@hotmail.com
28ffb4c4-04d7-425c-a4d7-9516a4959318	Micheal_Schuppe65	Tiana_OConner@hotmail.com
2aa39c8d-59a2-45e1-a390-9bcb95ccbf2a	Damon.Beahan-0rn	Erna_Mayer57@hotmail.com
2b775ca4-2d08-429b-a572-1c173913ece9	Reynold57	Earline96@yahoo.com
36ef8227-dbd4-4041-9474-e4328bba6e35	Tremayne51	Jasper.Doyle@yahoo.com
3807fb66-7a9a-4afd-bf4a-dc772c86a2d6	Penelope_MacGyver55	Alycia_Aufderhar40@gmail.com
3ac2df49-fb2c-46c9-81bc-60826dd82b23	Santino.Doo1ey194	Raven74@hotmail.com
3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82	Leslie_DAmore	Coralie.Hahn@gmail.com
42f0bc83-eba4-44b6-9309-1f14c2821027	Carleton_Wisoky3	Jakob.Sanford96@gmail.com
4c219478-9615-4cb4-b813-cd0461956600	Troy74	Justice.Dicki65@hotmail.com

DB - 100

Routing

REST

GET / → show no. of user in DB

GET /user → show users(email, id, username) *ejs*

PATCH /user/:id → username edit

~~ADD~~ POST /user → new user

DELETE /user/:id → user delete

EXPLORER

...

JS index.js X

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > ...

```
32     if (err) throw err;
33     console.log(result);
34   });
35 } catch (err) {
36   console.log(err);
37 }
38
39 connection.end();
40
41 app.get("/", (req, res) => {
42   res.send("welcome to home page");
43 });
44
45 app.listen("8080", () => {
46   console.log("server is listening to port 8080");
47 });
48
```

index.js > ...

```
const { faker } = require("@faker-js/faker");  
const mysql = require("mysql2");  
const express = require("express");  
const app = express();
```

Setting up Express App

GET /

Fetch & show total number of users on our app

```
app.get("/", (req, res) => {  
  let q = `SELECT count(*) FROM user`;  
  try {  
    connection.query(q, (err, result) => {  
      if (err) throw err;  
      let count = result[0]["count(*)"];  
      res.render("home.ejs", { count });  
    });  
  } catch (err) {  
    res.send("some error occurred");  
  }  
});
```

EXPLORER

SQLCLASS

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js

JS index.js > app.get("/") callback > connection.query() callback

```
21
22 app.get("/", (req, res) => {
23   let q = `SELECT count(*) FROM user`;
24   try {
25     connection.query(q, (err, result) => {
26       if (err) throw err;
27       console.log(result[0]["count(*)"]);
28       res.send("success");
29     });
30   } catch (err) {
31     console.log(err);
32     res.send("some error in DB");
33   }
34 });
35
36 app.listen("8080", () => {
37   console.log("server is listening to port 8080");
38 });
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

103
express deprecated res.send(status): Use res.sendStatus(status) instead in
103
103
103



EXPLORER

...

JS index.js X

SQLCL... [Icons]

> node_modules

views

JS index.js

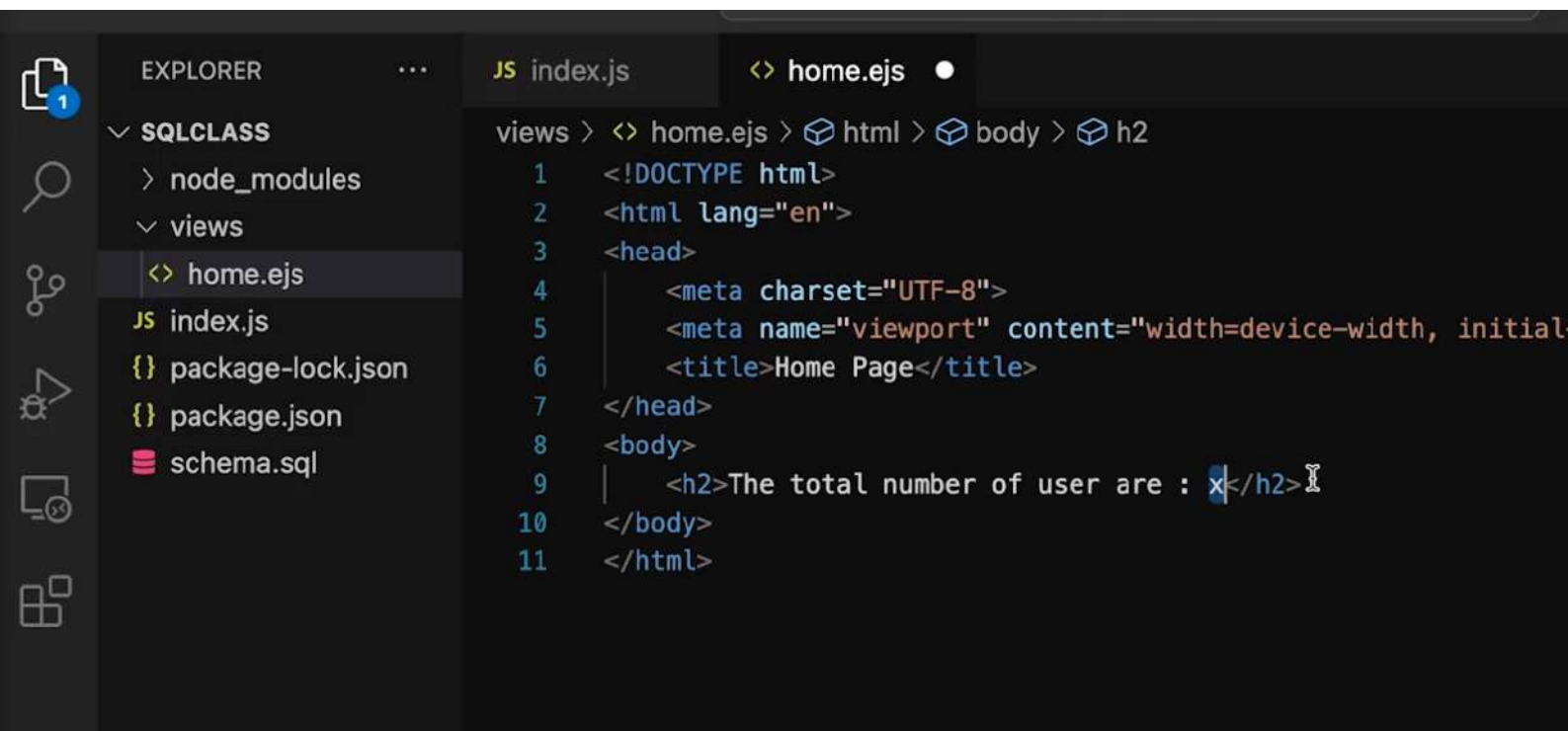
{ } package-lock.json

{ } package.json

schema.sql

JS index.js > ...

```
1  const { faker } = require("@faker-js/faker");
2  const mysql = require("mysql2");
3  const express = require("express");
4  const app = express();
5  const path = require("path");
6
7  app.set("view engine", "ejs");
8  app.use("views", path.join(__dirname, "/views"));
9
10 const connection = mysql.createConnection({
11   host: "localhost",
12   user: "root",
13   database: "delta_app",
14   password: "mysql@123",
15 });
16
17 let getRandomUser = () => {
18   return [
19     faker.datatype.uuid(),
20     faker.internet.userName(),
21     faker.internet.email(),
22     faker.internet.password(),
23   ];
24 }
```





EXPLORER


- SQLCLASS
 - node_modules
 - views
 - home.ejs
- index.js
- package-lock.json
- package.json
- schema.sql


views > <> home.ejs > html > body > h2 > ?


```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Home Page</title>
7   </head>
8   <body>
9     <h2>The total number of user are : <%= count %></h2>
10    <button>Join us today!</button>
11  </body>
12 </html>
13
```

















EXPLORER

SQLCLASS

> node_modules

> views

<> home.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js

<> home.ejs

JS index.js > app.get("/") callback > connection.query() callback

26 app.get("/", (req, res) => {

27 let q = `SELECT count(*) FROM user`;

28 try {

29 connection.query(q, (err, result) => {

30 if (err) throw err;

31 let count = result[0]["count(*)"];

32 res.render("home.ejs", { count });

33 });

34 } catch (err) {

35 console.log(err);

36 res.send("some error in DB");

37 }

38 });

39

40 app.listen("8080", () => {

41 console.log("server is listening to port 8080");

42 });

43

44 //try {

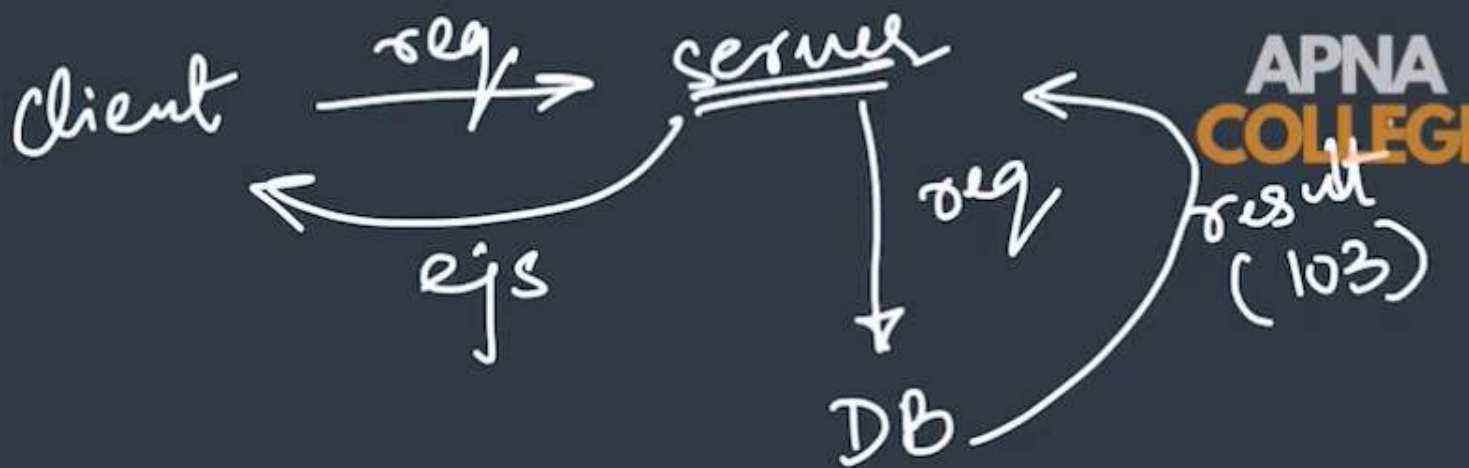
45

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE



Creating Our Routes

GET /user

Fetch & show (userId, username, email) of all users



```
app.get("/user", (req, res) => {  
  let q = `SELECT * FROM user`;  
  try {  
    connection.query(q, (err, result) => {  
      if (err) throw err;  
      let data = result;  
      console.log(data);  
      res.render("users.ejs", { data });  
    });  
  } catch (err) {  
    res.send("some error occurred");  
  }  
});
```

...

JS index.js

×

<> home.ejs

JS index.js >  app.get("/user") callback >  connection.query()

41 //Show Route

42 app.get("/user", (req, res) => {

43 let q = `SELECT * FROM user`;

44

45 try {

46 connection.query(q, (err, result) => {

47 if (err) throw err;

48 console.log(result);

49 res.send(result);

50 });

51 } catch (err) {

52 console.log(err);

53 res.send("some error in DB");

54 }

55 });

1

EXPLORER

...

SQLCLASS

> node_modules

> views

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js

<> showusers.ejs

<> home.ejs

views > <> showusers.ejs > html > body > table > tr

1 <!DOCTYPE html>

2 <html lang="en">

3 <head>

4 <meta charset="UTF-8">

5 <meta name="viewport" content="width=device-width, initial-scale=1.0">

6 <title>All Users</title>

7 </head>

8 <body>

9 <table>

10 <tr>

11 <th>Id</th>

12 <th>Email</th>

13 <th>Username</th>

14 </tr>

15

16 </table>

17 </body>

18 </html>

EXPLORER

SQLCLASS

- node_modules
- views
 - home.ejs
 - showusers.ejs

JS index.js

- package-lock.json
- package.json
- schema.sql

JS index.js

showusers.ejs X

home.ejs

views > showusers.ejs > html > body > table > tr > td

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

```
<title>All Users</title>
</head>
<body>
  <table>
    <tr>
      <th>Id</th>
      <th>Email</th>
      <th>Username</th>
    </tr>
    <tr>
      <td>123</td>
      <td>abc@gmail.com</td>
      <td>abc</td>
    </tr>
  </table>
</body>
</html>
```



```
40
41 //Show Route
42 app.get("/user", (req, res) => {
43     let q = `SELECT * FROM user`;
44
45     try {
46         connection.query(q, (err, result) => {
47             if (err) throw err;
48             // res.send(result);
49             res.render("showusers.ejs");
50         });
51     } catch (err) {
52         console.log(err);
53         res.send("some error in DB");
54     }
55 });
56
57 app.listen("8080", () => {
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

SQLCLASS

> node_modules

views

<> home.ejs

<> showusers.ejs

js index.js

{ } package-lock.json

{ } package.json

schema.sql

views > <> showusers.ejs > html > head > style > table > th

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>All Users</title>
7     <style>
8       table,
9       th,
10      td {
11        border: 1px solid black;
12      }
13    </style>
14  </head>
15  <body>
16    <table>
17      <tr>
18        <th>Id</th>
19        <th>Email</th>
```

EXPLORER

JS index.js

<> showusers.ejs ●

SQLCLASS

> node_modules

views

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

views > <> showusers.ejs > html > body > table > tr > th

```
11     border: 1px solid black;
12   }
13   </style>
14 </head>
15 <body>
16   <table>
17     <tr>
18       <th>Id</th>
19       <th>Email</th>
20       <th>Username</th>
21     </tr>
22     <% for(user of users) { %>
23       <tr>
24         <td><%= user.id %></td>
25         <td><%= user.email %></td>
26         <td><%=user.username%></td>
27       </tr>
28     <% } %>
29   </table>
30 </body>
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening to port 8080
[nodemon] restarting due to changes...
```

Id	Email	Username
02fd3f59-d408-4414-b6e7-6da573aa1ede	Lysanne18@hotmail.com	Michaela15
07a0ab83-221e-4b43-874a-5a9f69895ced	Francis_Krajcik@yahoo.com	Lauriane.Mertz3
09198186-4202-45bb-a7d7-76170bd70490	Marley_Metz87@gmail.com	Tianna71
0a6b3c30-e064-463c-b1b2-0bfd9196e31d	Mckayla37@gmail.com	Virgie62
0b20452f-b0c9-4bcb-a12f-457137e35ac3	Anita.Wilderman69@yahoo.com	Davin_Gutkowski
123	abc@gmail.com	123_newuser
123b	abc@gmail.comb	123_newuserb
123c	abc@gmail.comc	123_newuserc
12f1c618-e24e-4b5b-8624-db04f2a8d14c	Sonia.Bednar-Abernathy99@hotmail.com	Trisha95
151e72cb-1098-42fb-be53-c470e09d7a96	Mackenzie_Blanda6@gmail.com	Odie35
17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	Pat74
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61
25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1	Christa34@hotmail.com	Mathew_Fadel55
28ffb4c4-04d7-425c-a4d7-9516a4959318	Tiana_OConner@hotmail.com	Micheal_Schuppe65
2aa39c8d-59a2-45e1-a390-9bcb95ccb2a	Erna_Mayer57@hotmail.com	Damon.Beahan-Orn
2b775ca4-2d08-429b-a572-1c173913ece9	Earline96@yahoo.com	Reynold57
36ef8227-dbbc-4041-9474-e4328bba6e35	Jasper.Doyle@yahoo.com	Tremayne51
3807fb66-7a9a-4afd-bf4a-dc772c86a2d6	Alycia_Aufderhar40@gmail.com	Penelope_MacGyver55
3ac2df49-fb2c-46c9-81bc-60826dd82b23	Raven74@hotmail.com	Santino.Dooley19
3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82	Coralie.Hahn@gmail.com	Leslie_DAmore
42f0bc83-eba4-44b6-9309-1f14c2821027	Jakob.Sanford96@gmail.com	Carleton_Wisoky3
4c219478-9615-4cb4-b813-cd0461956600	Justice.Dicki65@hotmail.com	Troy74
4d4fc797-f789-472a-a06e-99fb14b8f09e	Patricia_Prosacco@gmail.com	Rodger_Wilderman84
4dc89a71-a84a-4825-82f5-52a2a31cd4fe	Sibyl25@yahoo.com	Elvis.Wisozk
4f1c9ad5-5435-4739-814c-77c364e55855	Bonita_Paucek@yahoo.com	Trey24

Creating Our Routes

GET /user/:id/edit

To get form to edit the username, based on id

This form will require a password

PATCH /user/:id

To edit username, if correct password was entered in form

SQLCLASS

> node_modules

views

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

views > <> showusers.ejs > html > body > table > ? > tr > td

```
14 </thead>
15 <body>
16   <table>
17     <tr>
18       <th>Id</th>
19       <th>Email</th>
20       <th>Username</th>
21     </tr>
22     <% for(user of users) { %>
23       <tr>
24         <td><%= user.id %></td>
25         <td><%= user.email %></td>
26         <td><%=user.username%></td>
27         <td>
28           <form method="GET" action="/user/<%=user.id%>/edit">
29             <button>Edit username</button>
30           </form>
31         </td>
32       </tr>
33     <% } %>
34   </table>
```


Id	Email	Username	
02fd3f59-d408-4414-b6e7-6da573aa1ede	Lysanne18@hotmail.com	Michaela15	Edit username
07a0ab83-221e-4b43-874a-5a9f69895ced	Francis_Krajcik@yahoo.com	Lauriane.Mertz3	Edit username
09198186-4202-45bb-a7d7-76170bd70490	Marley_Metz87@gmail.com	Tianna71	Edit username
0a6b3c30-e064-463c-b1b2-0bfd9196e31d	Mckayla37@gmail.com	Virgie62	Edit username
0b20452f-b0c9-4bc9-a12f-457137e35ac3	Anita.Wilderman69@yahoo.com	Davin_Gutkowski	Edit username
123	abc@gmail.com	123_newuser	Edit username
123b	abc@gmail.comb	123_newuserb	Edit username
123c	abc@gmail.comc	123_newuserc	Edit username
12f1c618-e24e-4b5b-8624-db04f2a8d14c	Sonia.Bednar-Abernathy99@hotmail.com	Trisha95	Edit username
151e72cb-1098-42fb-be53-c470e09d7a96	Mackenzie_Blanda6@gmail.com	Odie35	Edit username
17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55	Edit username
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54	Edit username
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	Pat74	Edit username
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73	Edit username
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61	Edit username
25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1	Christa34@hotmail.com	Mathew_Fadel55	Edit username
28ffb4c4-04d7-425c-a4d7-9516a4959318	Tiana_OConner@hotmail.com	Micheal_Schuppe65	Edit username
2aa39c8d-59a2-45e1-a390-9bcb95ccbf2a	Erna_Mayer57@hotmail.com	Damon.Beahan-Orn	Edit username
2b775ca4-2d08-429b-a572-1c173913ece9	Earline96@yahoo.com	Reynold57	Edit username
36ef8227-dbd9-4041-9474-e4328bba6e35	Jasper.Doyle@yahoo.com	Tremayne51	Edit username
3807fb66-7a9a-4afd-bf4a-dc772c86a2d6	Alycia_Aufderhar40@gmail.com	Penelope_MacGyver55	Edit username
3ac2df49-fb2c-46c9-81bc-60826dd82b23	Raven74@hotmail.com	Santino.Dooley19	Edit username
3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82	Coralie.Hahn@gmail.com	Leslie_DAmore	Edit username
42f0bc83-eba4-44b6-9309-1f14c2821027	Jakob.Sanford96@gmail.com	Carleton_Wisoky3	Edit username
4c219478-9615-4cb4-b813-cd0461956600	Justice.Dicki65@hotmail.com	Troy74	Edit username

Cannot GET /user/02fd3f59-d408-4414-b6e7-6da573aa1ede/edit

JS index.js

<> edit.ejs



views > <> edit.ejs > html > body > h2

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Edit Username</title>
7   </head>
8   <body>
9     <h2>You are about to edit this user</h2>
10    </body>
11  </html>
12
```

abc Username

EXPLORER

...

JS index.js

<> edit.ejs

SQLCLASS

> node_modules

views

<> edit.ejs

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > app.get("/user/:id/edit") callback

```
44
45     try {
46         connection.query(q, (err, users) => {
47             if (err) throw err;
48             res.render("showusers.ejs", { users });
49         });
50     } catch (err) {
51         console.log(err);
52         res.send("some error in DB");
53     }
54 });
55
56 //Edit Route
57 app.get("/user/:id/edit", (req, res) => {
58     res.render("edit.ejs")
59 })
60
61 app.listen("8080", () => {
62     console.log("server is listening to port 8080");
63 });
```

```
55
56 //Edit Route
57 app.get("/user/:id/edit", (req, res) => {
58   let { id } = req.params;
59   res.render("edit.ejs");
60 });
61
```

You are about to edit this user

```
54     });
55
56     //Edit Route
57     app.get("/user/:id/edit", (req, res) => {
58         let { id } = req.params;
59         let q = `SELECT * FROM user WHERE id='${id}'`;
60
61         try {
62             connection.query(q, (err, result) => {
63                 if (err) throw err;
64                 res.render("edit.ejs");
65             });
66         } catch (err) {
67             console.log(err);
68             res.send("some error in DB");
69         }
70     });
71
```

SQLCLASS

> node_modules

views

<> edit.ejs

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > app.get("/user/:id/edit") callback > connection.query() callback

```
58 let { id } = req.params;
59 let q = `SELECT * FROM user WHERE id='${id}'`;
60
61 try {
62   connection.query(q, (err, result) => {
63     if (err) throw err;
64     console.log(result);
65     res.render("edit.ejs");
66   });
67 } catch (err) {
68   console.log(err);
69   res.send("some error in DB");
70 }
71 });
72
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
[nodemon] starting `node index.js`
server is listening to port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening to port 8080
[
  {
    id: '09198186-4202-45bb-a7d7-76170bd70490',
    username: 'Tianna71',
    email: 'Marley_Metz87@gmail.com',
    password: 'zeEmhGnsACCWLHs'
  }
]
```

OUTLINE

TIMELINE

```
74     });  
75  
76     //UPDATE (DB) Route  
77     app.patch("/user/:id", (req, res) => {  
78         res.send("updated");  
79     });  
80  
81     app.listen("3000", () => {
```

EXPLORER

...

JS index.js

X

<> edit.ejs

SQLCLASS

> node_modules

views

<> edit.ejs

<> home.ejs

<> showusers.ejs

JS index.js

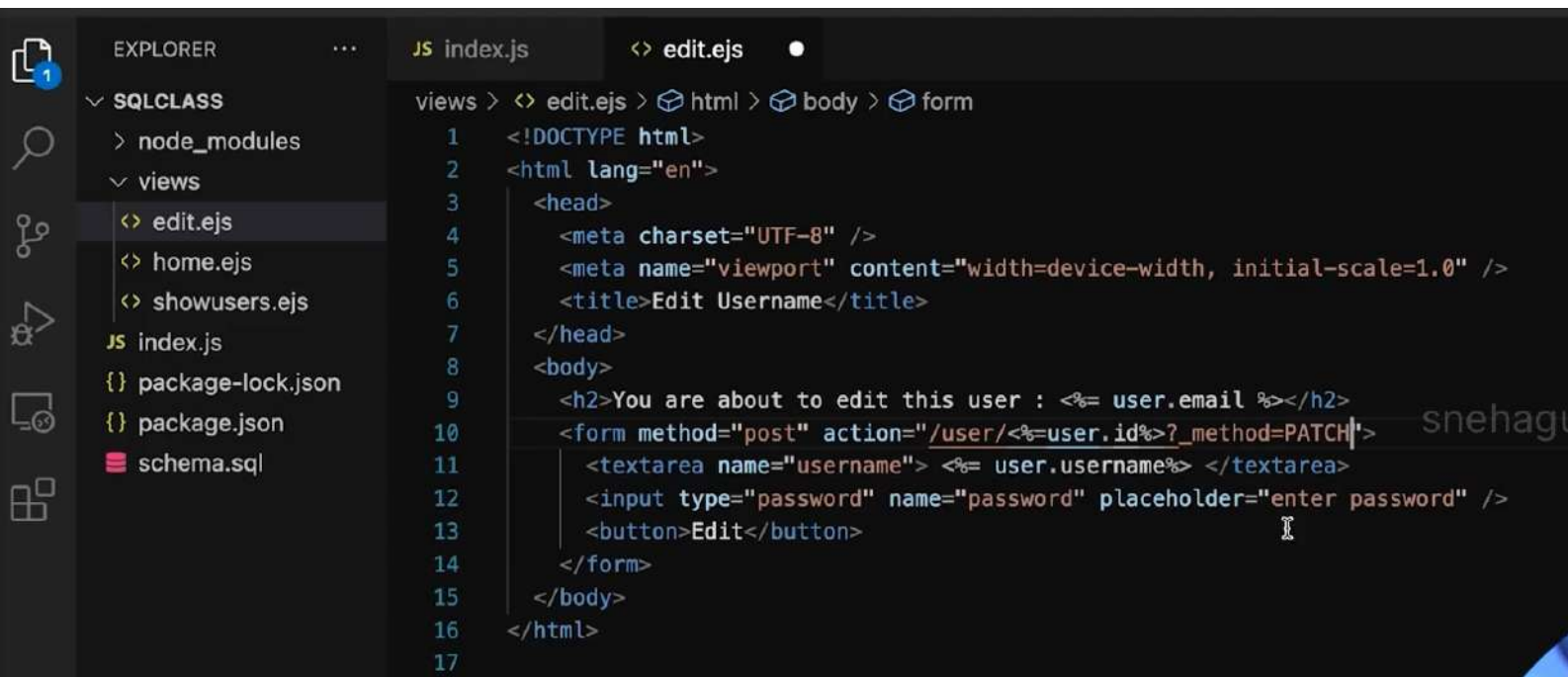
{ } package-lock.json

{ } package.json

schema.sql

JS index.js > ...

```
1  const { faker } = require("@faker-js/faker");
2  const mysql = require("mysql2");
3  const express = require("express");
4  const app = express();
5  const path = require("path");
6  const methodOverride = require("method-override");
7
8  app.use(methodOverride("_method"));
9  app.use(express.urlencoded({ extended: true }));
10 app.set("view engine", "ejs");
11 app.set("views", path.join(__dirname, "/views"));
12
13 const connection = mysql.createConnection({
14   host: "localhost",
15   user: "root",
16   database: "delta_app",
17   password: "mysql@123",
18 });
19
```



The image shows a VS Code editor window with the Explorer sidebar on the left. The Explorer sidebar shows a project named 'SQLCLASS' with a 'views' folder containing 'edit.ejs', 'home.ejs', and 'showusers.ejs'. The 'edit.ejs' file is selected and its content is displayed in the main editor area. The breadcrumb navigation at the top of the editor shows the path: 'views > edit.ejs > html > body > form'. The code in 'edit.ejs' is an HTML template for editing a user's information. It includes a form with a text area for the username and a password input field. The form is styled with a dark background and white text. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Edit Username</title>
7   </head>
8   <body>
9     <h2>You are about to edit this user : <%= user.email %></h2>
10    <form method="post" action="/user/<%=user.id%>?_method=PATCH">
11      <textarea name="username"> <%= user.username%> </textarea>
12      <input type="password" name="password" placeholder="enter password" />
13      <button>Edit</button>
14    </form>
15  </body>
16 </html>
17
```


Creating Our Routes

GET /user/:id/edit

To get form to edit the username, based on id

This form will require a password

- ✓ search user based on id
- 2) check if (form.password = pass)
 - 3) update username

PATCH /user/:id

To edit username, if correct password was entered in form

Form
↓
pass
New username
id }



EXPLORER

...

JS index.js x <> edit.ejs

SQLCLASS

> node_modules

v views

<> edit.ejs

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js > app.patch("/user/:id") callback

```
74 });
75
76 //UPDATE (DB) Route
77 app.patch("/user/:id", (req, res) => {
78   let { id } = req.params;
79   let { password: formPass, username: newUsername } = req.body;
80   let q = `SELECT * FROM user WHERE id='${id}'`;
81
82   try {
83     connection.query(q, (err, result) => {
84       if (err) throw err;
85       let user = result[0];
86       if (formPass !== user.password) {
87         res.send("WRONG password");
88       } else {
89         let q2 = `UPDATE user SET username='${newUsername}' WHERE id='${id}'`;
90         connection.query(q2, (err, result) => {
91           if (err) throw err;
92           res.send(result);
93         });
94       }
95     });
96   } catch (err) {
97     console.log(err);
```

mysql2 - npm



localhost:8080/user/1fc0bed1-

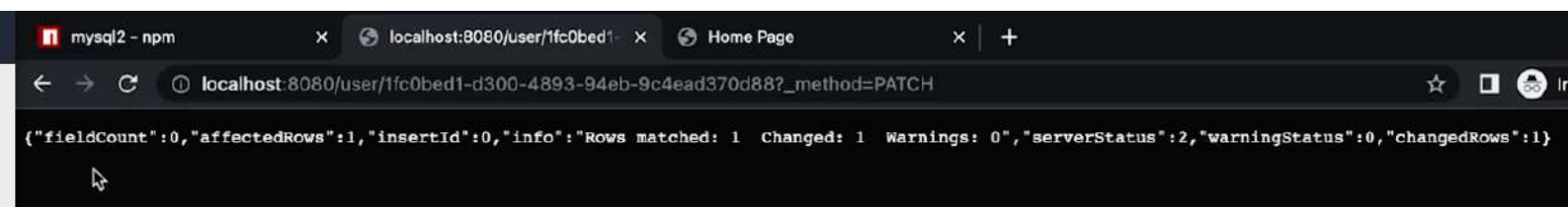


Home Page



localhost:8080/user/1fc0bed1-d300-4893-94eb-9c4ead370d88?_method=PATCH

WRONG password



17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55	Edit username
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54	Edit username
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	shradha	Edit username
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73	Edit username
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61	Edit username

EXPLORER

SQLCLASS

> node_modules

> views

<> edit.ejs

<> home.ejs

<> showusers.ejs

JS index.js

{ } package-lock.json

{ } package.json

schema.sql

JS index.js

JS index.js > app.patch("/user/:id") callback > connection.query() callback > connection.query()

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

try {

connection.query(q, (err, result) => {

if (err) throw err;

let user = result[0];

if (formPass !== user.password) {

res.send("WRONG password");

} else {

let q2 = `UPDATE user SET username='\${newUsername}' WHERE id='\${id}'`;

connection.query(q2, (err, result) => {

if (err) throw err;

res.redirect("/user");

});

}

});

} catch (err) {

console.log(err);

res.send("some error in DB");

}

}

```
75
76 //UPDATE (DB) Route
77 app.patch("/user/:id", (req, res) => {
78   let { id } = req.params;
79   let { password: formPass, username: newUsername } = req.body;
80   let q = `SELECT * FROM user WHERE id='${id}'`;
81
82   try {
83     connection.query(q, (err, result) => {
84       if (err) throw err;
85       let user = result[0];
86       if (formPass !== user.password) {
87         res.send("WRONG password");
88       } else {
89         let q2 = `UPDATE user SET username='${newUsername}' WHERE id='${id}'`;
90         connection.query(q2, (err, result) => {
91           if (err) throw err;
92           res.redirect("/user");
93         });
94       }
95     });
96   } catch (err) {
97     console.log(err);
98     res.send("some error in DB");
99   }
100 });
101
102 app.listen("8080", () => {
```

Homework Tasks

1) Create Form to Add a new user to the Database

Create Form to Delete a user from Database if they enter correct email id & password.