

**Implemented the following functionalities:**

SP Share authenticates each user and Multiple concurrent users can share “items” such as pictures, text, executables (other format “binary” (raw) files), small videos.  
There are three types of SPS “users”: general, admin and root.  
Root will grant all permissions to admin, and admin can approve any user.

General user will be registered with the actual name other than login name along with password and the group name.

All the General users who have been authenticated may View, Put, Get, and Remove, (as well as edit) items: pictures, videos, text, and other format “binary” (raw) files.

View – selects all the items in all groups that a user is a member of and display them to the user.

Get – This functionality will extract the item (meta data of the files along with its file id(if flat-null else fileid), item id, timestamp, item description of all the files the user has uploaded).

Remove – Drops the entire item entry from mysql database. This privilege is given only for the item creator or the admin.

Put – Inserts the data into the respective item table in the mysql database. (item description as creator name, timestamp etc)

Download – Downloads the file for the file id mentioned to the user.

Group names and general users must be accepted by the admin before being activated (asynchronously).

The admin must accept (allow) user (who have previously registered), requested groups, be able to create new groups, and be able to set some limits of space consumption: limits per individual user, limits per group, and limits per item.

Users (as well as admin) may operate independently (asynchronously).  
Items (group files) are stored in a mysql database.

All passwords, protection, etc must be “secure”. – We are hashing the password and generating a token through which the user is authenticated the next time he logs in.

All the users will be signedup(registered) with their unique loginname, password and actual name.

All the other mentioned description for each filed names is taken into consideration.

Deployed in AWS. <http://18.222.143.83:3000/>

## References: StackOverFlow, Wikipedia



POST localhost:3000/auth/signup

localhost:3000/auth/signup

POST http://18.222.143.83:3000/auth/signup

Body

```
1> {
2   "loginname": "root",
3   "password": "root",
4   "actualname": "root user"
5 }
```

Hit the Send button to get a response.

```

44 router.post('/signup', function (req, res) {
45   console.log(req);
46   let saltRounds = 10;
47   let originalPwd = connection.escape(req.body.password);
48   let loginname = connection.escape(req.body.loginname);
49   let actualname = connection.escape(req.body.actualname);
50
51   if (loginname === null) {
52     connection.query("SELECT * FROM sp_share_new.userz WHERE upper(userz.loginname) = upper('" + loginname + "')", function (err, result, fields) {
53       if (result.length === 0) {
54         bcrypt.hash(originalPwd, saltRounds, function (error, hashedPwd) {
55           connection.query("Insert into sp_share_new.userz (actualname, loginname, password) VALUES (" + actualname + "," + loginname + "," + hashedPwd + ")");
56           if (error) {
57             res.sendStatus(409);
58           } else {
59             res.sendStatus(200);
60             usage_monitor.initUserUsage(loginname);
61           }
62         });
63       } else {
64         res.sendStatus(409);
65       }
66     });
67   }
68 }
69 });
70 });
71 });
72 });

```

The password sent will be hashed based on salts(number of salts used is 10). Used bcrypt method. Also created a jwt token which expires in 24 hours. This will be updated only when is\_Activated is true(only for active users). Attached the method screenshot below.

```

module.exports = function (connection, bcrypt, jwt, config, shortid) {
  router.post('/signin', function (req, response) {
    let password = req.body.password;
    let loginname = req.body.loginname;
    if (loginname === null && password === null) {
      connection.query("SELECT password FROM sp_share_new.userz WHERE loginname = " + connection.escape(loginname), function (err, result) {
        console.log(result);
        if (result) {
          var hash = result[0]['password'];
          bcrypt.compare(password, hash, function (err, res) {
            console.log(res);
            if (res) {
              // Create a token
              var token = jwt.sign({
                loginname: loginname,
                config.secret,
                expiresIn: 86400 // expires in 24 hours
              },
              function (err, token) {
                connection.query("UPDATE sp_share_new.userz SET jwt = " + connection.escape(token) + ", istokengenerated = true WHERE loginname = " + connection.escape(loginname));
                response.status(200).send({
                  auth: true,
                  token: token
                });
                console.dir(this);
              });
            }
          });
        }
      });
    }
  });
}

```

Based on the loginname, if it is root, the user type will be "R"(root) else it will be "U"(general user) or "A"(admin). In Database, under userz table, an entry will be created. Sample screenshot attached below.

The screenshot shows the MySQL Workbench interface. On the left, the 'Object Info' sidebar is open, showing the 'Tables' section for the 'sp\_share\_new' schema, which contains tables like 'files', 'group\_membership', 'groups', 'items', 'item\_sharing', 'userz', and 'sys'. The main pane displays a query editor with the following SQL command:

```
SELECT * FROM sp_share_new.userz;
```

The results grid shows two rows of data:

	actualname	loginname	password	jwt	istokengenerated	invalidated	timestamp	userstype
1	swetha3	swetha	\$2b\$10\$B1Wy/T9TMZ2P4n1RLvSpQ8H1u6...	eyJhbGciOiJzI1NisInR5cI6IkpxVCJ9.eyJ...	001	000	2019-11-15 13:45:19	U
2	root	root	\$2b\$10\$7dSBVqJkA9hAVOrSpV3yGexNC4BK...	...	000	000	2019-11-17 04:16:17	R

Below the results grid, the 'Logs' tab is selected, showing the following log entries:

Time	Action	Response	Duration / Fetch Time
3 22:16:10	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.042 sec / 0.000017...
4 22:17:08	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.00003...
5 22:17:37	UPDATE `sp_share_new`.`userz` SET `userstype` = 'R' WHERE (`loginname` = 'root')	1285: Data truncated for column 'userstype' at row 1	
6 22:18:41	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.045 sec / 0.000019...
7 22:42:29		Error Code: 2013 Lost connection to MySQL server at...	

Once the user is signed up, the same user can be logged in to the system. In the screenshot below, you can see we can give loginname, password, actual name is optional here and, in the response, we will get a token generated only when the authentication is done.

The screenshot shows the Postman application interface. The left sidebar lists collections: 'Postman Echo' (38 requests) and 'swetha-app' (8 requests). The main workspace shows a POST request to 'http://18.222.143.83:3000/auth/signup' with the following JSON body:

```
[{"loginname": "root", "password": "root", "actualname": "root user"}]
```

The response status is 200 OK, with a duration of 174 ms and a size of 390 B. The response body is:

```
{"auth": true, "token": "eyJhbGciOiJIUzI1NiIsInR5cI6IkpxVCJ9.eyJsb2dpbmShbWl0Ijyb290IiwiZWFIjoxNTQyNDI5ODI5LCJleHAiOjE1NDI1MTYyMjI1.XtndxjZNklaRIP_fjHkg9d0mSSVDr_moZrsqGLd0"}
```

Once we get the bearer token, I will match the login name obtained and stored loginname from db and if it matches then the user is authenticated else Failed to authenticate token message is displayed.

```
1 module.exports = function (connection, bcrypt, jwt, config, token, req, res) {
2     return function(callback){
3         if (!token) return res.status(401).send({
4             auth: false,
5             message: 'No token provided.'
6         });
7         jwt.verify(token, config.secret, function (err, decoded) {
8             if (err) return res.status(500).send({
9                 auth: false,
10                 message: 'Failed to authenticate token.'
11             });
12             if (err) {
13                 connection.query("SELECT loginname FROM sp_share_new.userz WHERE loginname = " + connection.escape(decoded.loginname) + " AND jwt = " + connection.escape(toke
14             if(result) {
15                 if(typeof callback == 'function'){
16                     req.local.loginname = result[0];
17                     req.local.loginname = decoded.loginname;
18                     // res.local = {};
19                     // res.local.loginname = decoded.loginname;
20                     callback();
21                 }
22             }
23         });
24     });
25 };
26
27
28 }
29 }
```

Line 20, Column 40      Spaces: 4      JavaScript

Once the user is authenticated, root can make any user as admin described in makeUserAdmin, and in the db, we can see first `go3swetha` had a user type 'U' now reflected to 'A'.

```
6 router.post('/makeUserAdmin', function (req, res) {
7     let username = connection.escape(req.body.username);
8     let xx = req.header('authorization');
9     let tokenArr = xx.split(" ");
10    let token = TokenArr[1];
11    let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
12    _bav(function () {
13        let login_name = connection.escape(req.local.loginname);
14        if (String(req.body.username).toLowerCase() == 'root') {
15            res.sendStatus(401);
16        } else {
17            connection.query("select userz.loginname from userz where userz.usertype='R' and userz.loginname =" + login_name, function (error, result) {
18                if (error && result.length > 0) {
19                    connection.query("UPDATE `sp_share_new`.`userz` SET `usertype`='A' WHERE `loginname`=" + username, function (error, result) {
20                        if (error) {
21                            res.send(200);
22                        }
23                    });
24                }
25            });
26        }
27    })
28 }
29 return router;
30 }
```

Postman Screenshot:

- Header:** POST http://18.222.143.83:3000/auth/makeUserAdmin
- Body:** JSON (application/json)
 

```
{
        "username": "go3swetha"
      }
```
- Response:**
  - Status: 200 OK
  - Time: 100 ms
  - Size: 206 B
  - Content: OK

MySQL Workbench Screenshot:

- Schema:** sp\_share\_new
- Table:** userz
- Query:** SELECT \* FROM sp\_share\_new.userz;
- Result Grid:**

actualname	loginname	password	jwt	istokengenerated	isvalidated	timestamp	usertype
go3swetha	go3swetha	\$2b\$10\$B1WyT9TMZPR4n1RlvSp08H61U6...eyJhbGciOiUz1NilsInR5cI6IkpxVCQj9 eyJsb...	eyJhbGciOiUz1NilsInR5cI6IkpxVCQj9 eyJsb...	001	000	2018-11-15 13:45:19	A
root user	root	\$2b\$10\$MAHPH4glcfqC2FLxLjOKaBceCAF...eyJhbGciOiUz1NilsInR5cI6IkpxVCQj9 eyJsb...	eyJhbGciOiUz1NilsInR5cI6IkpxVCQj9 eyJsb...	001	000	2018-11-17 04:43:26	R
- Logs:**

Time	Action	Response	Duration / Fetch Time
14 23:28:16	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.00003...
15 23:28:22	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.000012...
16 23:34:00	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.000015...
17 23:34:02	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.065 sec / 0.000012...
18 23:34:41	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.000019...

POST http://18.222.143.83:3000/groups/new

Body (2) `{"group_name": "movies"}`

Status: 200 OK Time: 93 ms Size: 219 B

```
i 1 'movies' created
```

group_name	datetime	group_creator	isActivated
NULL	NULL	NULL	NULL

Action Log:

Time	Action	Response	Duration / Fetch Time
16 23:34:00	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.000015...
17 23:34:02	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.065 sec / 0.000012...
18 23:34:41	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	2 row(s) returned	0.040 sec / 0.000019...
19 23:58:03	SELECT * FROM sp_share_new.userz LIMIT 0, 1000	3 row(s) returned	0.040 sec / 0.000018...
20 00:11:10	SELECT * FROM sp_share_new.groups LIMIT 0, 1000	1 row(s) returned	0.041 sec / 0.000015...

```

router.post('/new', function (req, res, next) {
  var xx = req.header('authorization');
  let TokenArray = xx.split(" ");
  var token = TokenArray[1];
  const _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
  _bav(function (loginname) {
    next();
  })
}, function (req, res, next) {
  let group_name = connection.escape(req.body.group_name);
  let loginname = connection.escape(req.local.loginname);
  connection.query("INSERT INTO sp_share_new.`groups`(`group_name`, `group_creator`) VALUES (" + group_name + ", " + loginname + ")",
    function (err, result) {
      if (!err) {
        res.status(200).send(group_name + 'created');
        usage_monitor.initGroupUsage(group_name);
      } else {
        res.status(406).send(group_name + ': cannot be created');
      }
    });
});

```

The Root will activate the admin, in turn admin will activate the user (The admin must accept (allow) user (who have previously registered), requested groups)

Who ever is activating the general user, their key will be provided in the bearer token and loginname will be the new user who is yet to get activated. This is implemented under makeUserActive route.

POST http://18.222.143.83:3000/auth/makeUserActive

Params Headers (2) Body Pre-request Script Tests Cookies Code

TYPE Bearer Token

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJsb2dpbm5hbWUiOiJyb290liwiWF0ljoxNTQy...

Preview Request

Status: 200 OK Time: 86 ms Size: 206 B Download

Body Pretty Raw Preview Auto

1 OK

```

5 router.post('/makeUserAdmin', function (req, res) {
6     let username = connection.escape(req.body.username);
7     let xx = req.header('authorization');
8     let TokenArray = xx.split(" ");
9     let token = TokenArray[1];
10    let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
11    _bav(function () {
12        let login_name = connection.escape(req.local.loginname);
13        if (String(req.body.username).toLowerCase() == 'root') {
14            res.sendStatus(401);
15        } else {
16            connection.query("select userz.loginname from userz where userz.usertype='R' and userz.loginname =" + login_name, function (error, result) {
17                if (error && result.length > 0) {
18                    connection.query("UPDATE `sp_share_new`.`userz` SET `usertype`='A' WHERE `loginname`=" + username, function (error, result) {
19                        if (error) {
20                            res.send(200)
21                        }
22                    })
23                }
24            })
25        }
26    })
27}
28
29 return router;
30

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there are two items: 'Postman Echo' (38 requests) and 'swetha-app' (11 requests). The main area displays a POST request to 'http://18.222.143.83:3000/auth/makeUserActive'. The 'Body' tab is selected, showing a JSON payload: { "loginname": "gozsueet" }. Below the body, the response status is shown as 'Status: 200 OK'.

The group is created, and it will be stored in groups table in the db

Postman Echo

History

swetha-app

POST http://18.222.143.83:3000/groups/new

Params Authorization Headers (2) Body Pre-request Script Tests Cookies Code

Body (JSON application/json)

```
{ "group_name": "documents" }
```

Status: 200 OK Time: 88 ms Size: 223 B Download

Pretty Raw Preview HTML

i 1 | 'documents' created

MANAGEMENT

INSTANCE

PERFORMANCE

SCHEMAS

Table: groups

Columns:

group_name	datetime	group_creator	isActivated
documents	2018-11-17 07:26:23	root	NULL
movies	2018-11-17 06:11:31	goSujeet	NULL
	HULL	HULL	HULL

Time Action Response Duration / Fetch Time

19 23:58:03 SELECT \* FROM sp\_share\_new.userz LIMIT 0,1000 3 row(s) returned 0.040 sec / 0.000018...

20 00:11:10 SELECT \* FROM sp\_share\_new.groups LIMIT 0,1000 1 row(s) returned 0.041 sec / 0.000016...

21 00:12:46 SELECT \* FROM sp\_share\_new.userz LIMIT 0,1000 3 row(s) returned 0.044 sec / 0.000016...

22 01:17:29 Error Code: 2013 Lost connection to MySQL server at...

23 01:26:01 SELECT \* FROM sp\_share\_new.groups LIMIT 0,1000 2 row(s) returned 0.040 sec / 0.000021...

Once the group is created, the group has to be approved by the root or admin.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there are two entries: 'Postman Echo' (38 requests) and 'swetha-app' (13 requests). The main area displays a POST request to 'http://18.222.143.83:3000/groups/makeGroupActive'. The 'Body' tab is selected, showing the JSON payload: { "group\_name": "documents" }. Below the request, the response status is '200 OK', time '98 ms', and size '206 B'. The response body contains the string 'OK'.

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for 'MANAGEMENT', 'INSTANCE', 'PERFORMANCE', and 'SCHEMAS'. Under 'SCHEMAS', 'sp\_share\_new' is expanded, showing 'Tables' like 'files', 'group\_membership', 'groups', 'items', 'item\_sharing', 'userz', 'Views', 'Stored Procedures', 'Functions', and 'sys'. The central area shows the 'userz' schema with two tables: 'groups' and 'userz'. The 'groups' table has columns: group\_name, datetime, group\_creator, and isActivated. The data shows three rows: 'documents' (root, 2018-11-17 07:26:23, root, 001), 'movies' (go2sujet, 2018-11-17 06:11:31, go2sujet, 001), and 'NULL' (NULL, NULL, NULL, NULL). The 'userz' table is also visible.

Eventhough user has created the group, not necessarily he needs to join. So I have an route which joins user to the provided `group\_name`. This needs admin or root approval.

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for 'New', 'Import', 'Runner', and 'Invite'. The title bar says 'My Workspace' and 'No Environment'. Below the title bar, there is a search bar labeled 'Filter' and a 'Collections' section with 'History' and 'swetha-app' (14 requests). The main workspace shows a POST request to 'http://18.222.143.83:3000/groups/join'. The 'Body' tab is selected, showing a JSON payload: { "group\_name": "documents" }. The 'Params', 'Authorization', 'Headers', 'Pre-request Script', and 'Tests' tabs are also visible. Below the request, there is a preview of the response: 'Status: 200 OK Time: 95 ms Size: 241 B'. The response body contains the message 'Added to the Group, Pending Approval'. At the bottom of the screen, there is a code editor window displaying Node.js server code for a 'join' route.

```

router.post('/join', function (req, res, next) {
  var xx = req.header('authorization');
  let TokenArray = xx.split(" ");
  var token = TokenArray[1];
  let gm_id = connection.escape('GM-' + shortid.generate());
  const _bav = new BAV(connection, bcrypt, jwt, config, token, req);
  let group_name = connection.escape(req.body.group_name);
  let member_type = connection.escape(req.body.member_type);
  _bav(function (Loginname) {
    connection.query("SELECT group_name FROM sp_share_new.groups` WHERE group_name = " + connection.escape(req.body.group_name), function (err, result) {
      if (!err) {
        connection.query("SELECT group_name FROM sp_share_new.groups` WHERE group_name = " + connection.escape(req.body.group_name), function (error, result) {
          if (!error) {
            connection.query("INSERT INTO sp_share_new.group_membership (member_name,group_name,gm_id,member_type) VALUES (" + connection.escape(req.local.log
              if (!err) {
                res.status(200).send("Added to the Group, Pending Approval");
                usage_monitor.initGroupUserUsage(connection.escape(req.local.loginname),connection.escape(connection.escape(group_name)));
              }
            })
          }
        });
      }
    });
  });
});

```

Once user is requested to join a specific group by providing a group\_name, this will be stored in group\_membership table. Since the user is not yet approved, the isAdded field is by default false.

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation panels for MANAGEMENT, INSTANCE, PERFORMANCE, SCHEMAS, and OBJECT INFO. The SCHEMAS panel shows the current schema is 'sp\_share\_new'. The main area has a query editor titled 'userz' containing the following SQL code:

```
1 • SELECT * FROM sp_share_new.groups;
2 •
3 • SELECT * FROM sp_share_new.userz;
4 •
5 •
6 • SELECT * FROM sp_share_new.group_membership;
```

Below the query editor is a results grid titled 'Result Grid' for the 'group\_membership' table. The grid displays one row of data:

group_name	member_name	isAdded	timestamp	gm_id
documents	root	NULL	2018-11-17 07:42:48	GM-qvr2yA_c7

At the bottom, there is a log of actions and a summary of the response.

Now the user has to be approved under a group name requested by admin or root. This is done under makeGroupActive

Sequel Pro - MySQL 5.7.24 - localhost/sp\_share\_new/group\_membership

Tables:

- files
- group\_membership
- groups
- item\_sharing
- items
- userz

Structure Content Relations Triggers Table Info Query

Search: group\_name =

group_name	member_name	isAdded	timestamp	gm_id	member_type
documents	go3swetha	1	2018-11-17 13:30:15	dasdasdasda	A
movies	go4swetha	0	2018-11-17 17:54:27	GM-zoF5AU_5b	U

TABLE INFORMATION

- created: 11/17/18
- updated: 11/18/18
- engine: InnoDB
- rows: 2
- size: 16.0 Kib
- encoding: utf8mb4

Postman Screenshot:

- Header:** POST http://18.222.143.83:3000/groups/makeGroupMemberActive
- Body (JSON):**

```
1 - {"group_name": "movies",
2   "member_name": "go4swetha"
3 }
```
- Response Headers:** Status: 200 OK, Time: 106 ms, Size: 206 B

Now the isValidated column will be 001.

Sequel Pro Screenshot:

	group_name	member_name	isAdded	timestamp	gm_id	member_type
1	documents	go3swetha	0	2018-11-17 13:30:15	dadsdadasda	A
2	movies	go4swetha	0	2018-11-17 17:54:27	GM-zoF5AU_5b	U

TABLE INFORMATION

- created: 11/17/18
- updated: 11/18/18
- engine: InnoDB
- rows: 2
- size: 16.0 KiB
- encoding: utf8mb4

Uploading (File based)

My Workspace

POST http://18.222.143.83:3000/items/upload

Body

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> filename	Choose Files Screen Shot 2018-11-18 at 12.15.40 AM	screenshot regarding project
Key	Value	Description

Hit the Send button to get a response.

Build Browse

Screen Shot 2018-11-18 at 12.15.40 AM

Screen Shot 2018-11-18 at 12.15.40 AM

Cancel Open

Hit the Send button to get a response.

The image shows two separate instances of the Postman application interface, both displaying a collection named "swetha-app".

**Request 1 (Top):**

- Method: POST
- URL: http://18.222.143.83:3000/items/upload
- Body tab selected.
- Form-data:

  - filename: Choose Files (ss)
  - screenshot: swetha raghavan

- JSON response (Pretty):

```
1 < {  
2   "fileID": "FU-7-ViiZywK"  
3 }
```

**Request 2 (Bottom):**

- Method: POST
- URL: http://18.222.143.83:3000/items/upload
- Body tab selected.
- Form-data:

  - filename: Choose Files (Cartoon-Mario.jpg)
  - screenshot: swetha raghavan

- JSON response (Pretty):

```
1 < {  
2   "fileID": "FU-ggQmlohes"  
3 }
```

File is uploaded. Along with a short id.

Sequel Pro

File Edit View Database Table Bundles Window Help

(MySQL 5.7.24-Oubuntu0.18.04.1) localhost/sp\_share\_new/files

SSH Connected

Select Database Structure Content Relations Triggers Table Info Query

Table History Users Console

TABLES

Search: file\_name

file_name	file_id	timestamp	original_file_name
FU-7-ViiZywK-ss.png	FU-7-ViiZywK	2018-11-18 06:55:02	ss.png
FU-c8MPTTwmm-btc qr.png	FU-c8MPTTwmm	2018-11-18 06:45:53	btc qr.png
FU-gqQm1ohe-Cartoon-Mario.jpg	FU-gqQm1ohe	2018-11-18 06:53:43	Cartoon-Mario.jpg
FU-Hiup_cvL_btc qr.png	FU-Hiup_cvL	2018-11-18 06:52:49	btc qr.png
FU-lhuaAxvZj-btc qr.png	FU-lhuaAxvZj	2018-11-18 06:45:46	btc qr.png
FU-s5VgkEnXq-btc qr.png	FU-s5VgkEnXq	2018-11-18 06:45:59	btc qr.png
FU-wMuDheASM-btc qr.png	FU-wMuDheASM	2018-11-18 06:45:34	btc qr.png
FU-WXuIEqWcL-btc qr.png	FU-WXuIEqWcL	2018-11-18 06:46:33	btc qr.png
FU-XbfSLzzto-btc qr.png	FU-XbfSLzzto	2018-11-18 06:46:30	btc qr.png
FU-zVedr6qnU-btc qr.png	FU-zVedr6qnU	2018-11-18 06:45:56	btc qr.png

TABLE INFORMATION

- created: 11/15/18
- updated: 11/19/18
- engine: InnoDB
- rows: 10
- size: 16.0 KiB
- encoding: utf8mb4

10 rows in table; 1 row selected

Open Connection Quick Connect Action Refresh Edit

Disconnect

/home/ubuntu/app

Search

Filename	Size	Modified
commons	62 B	11/10/18, 1:39 AM
config.js	2.8 KB	11/16/18, 10:40 PM
index.js		-- 11/15/18, 7:14 AM
node_modules		
package-lock.json	48.0 KB	11/12/18, 1:04 PM
package.json	984 B	11/12/18, 1:04 PM
README.md	3.6 KB	11/13/18, 10:52 PM
routes		-- Yesterday, 11:30 AM
auth.js	5.7 KB	Yesterday, 1:03 AM
groups.js	5.6 KB	Today, 12:15 AM
items.js	11.2 KB	Today, 12:41 AM
uploads		-- Today, 12:55 AM
FU-7-ViiZywK-ss.png	1.8 MB	Today, 12:55 AM
FU-A4rGqIvb-btc qr.png	17.9 KB	11/16/18, 1:27 AM
FU-c8MPTTwmm-btc qr.png	17.9 KB	Today, 12:45 AM
FU-gqQm1ohe-Cartoon-Mario.jpg	629.7 KB	Today, 12:53 AM
FU-Hiup_cvL_btc qr.png	17.9 KB	Today, 12:52 AM
FU-klkB7TX9z-btc qr.png	17.9 KB	11/16/18, 1:23 AM
FU-lhuaAxvZj-btc qr.png	17.9 KB	Today, 12:45 AM
FU-s5VgkEnXq-btc qr.png	17.9 KB	Today, 12:45 AM
FU-wMuDheASM-btc qr.png	17.9 KB	Today, 12:45 AM
FU-WXuIEqWcL-btc qr.png	17.9 KB	Today, 12:46 AM
FU-XbfSLzzto-btc qr.png	17.9 KB	Today, 12:45 AM

24 Files

File if file type not given, considered as flat. This is implemented in put route

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there are two entries: 'Postman Echo' (38 requests) and 'swetha-app' (19 requests). The main area displays a POST request to 'http://18.222.143.83:3000/items/put'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2     "description": "by g333_new Swetha",  
3     "share-with": "documents",  
4     "fileid": "FU-7-VlZyWk"  
5 }
```

Below the body, the response status is 200 OK, Time: 95 ms, Size: 206 B. The response body is 'OK'.

The screenshot shows the Sequel Pro application interface. The title bar indicates it's connected to MySQL 5.7.24-Oubuntu0.18.04.1 on localhost. The database is 'sp\_share\_new'. The 'item\_sharing' table is selected. The table structure includes columns: item\_id, group\_name, and creation\_timestamp. A single row is visible:

item_id	group_name	creation_timestamp
IT-Hj2kFjDd	documents	2018-11-18 07:35:20

On the left, the 'Tables' pane lists tables: files, group\_membership, groups, item\_sharing, items, and userz. At the bottom, 'TABLE INFORMATION' provides details: created: 11/15/18, updated: 11/18/18, engine: InnoDB, rows: 2, size: 16.0 KiB, encoding: utf8mb4.

```

78 // Share Schema
79 // Iless: pictures, videos, text, and other format "binary" (raw) files.
80
81 router.post('/put', function (req, res) {
82   let xx = req.headers['content-type'];
83   let token = xx.split(';')[0];
84   let tokenArr = token.split(' ');
85   let token = tokenArr[0];
86   let itemBody = req.body;
87
88   let item_description = connection.escape(itemBody.description);
89   let item_id = itemBody.fileid ? 'FILE' : 'FLAT';
90   let file_id = connection.escape(itemBody.fileid);
91   let item_id = connection.escape('IT-' + shortid.generate());
92   let shared_group = connection.escape(itemBody.shareWith);
93
94
95   let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
96   _bav(function () {
97     let group_name = connection.escape(req.local.username);
98     if (item_type == 'FLAT') {
99       item_type = connection.escape(item_type);
100      connection.query("INSERT INTO `sp_share_new`.items` ('item_id', 'item_type', 'item_description', 'item_creator') VALUES (" + item_id + ", " + item_type + ", " + item_description + "," + group_name + ")", function (error, result) {
101        if (error) {
102          connection.query("select group_name from sp_share_new.groups where groups.group_name = " + shared_group, function (error, result) {
103            if (result.length > 0) {
104              connection.query("INSERT INTO `sp_share_new`.item_sharing` ('item_id', 'group_name') VALUES (" + item_id + "," + shared_group + ")", function (error, result) {
105                if (error) {
106                  res.sendStatus(200);
107                }
108              }
109            }
110          }
111        }
112      })
113    } else {
114      item_type = connection.escape(item_type);
115      connection.query("INSERT INTO `sp_share_new`.items` ('item_id', 'item_type', 'file_id', 'item_description', 'item_creator') VALUES (" + item_id + ", " + item_type + ", " + file_id + ", " + item_description + "," + group_name + ")", function (error, result) {
116        if (error) {
117          connection.query("select group_name from sp_share_new.groups where groups.group_name = " + shared_group, function (error, result) {
118            if (result.length > 0) {
119              connection.query("INSERT INTO `sp_share_new`.item_sharing` ('item_id', 'group_name') VALUES (" + item_id + "," + shared_group + ")", function (error, result) {
120                if (error) {
121                  res.sendStatus(200);
122                }
123              }
124            }
125          }
126        }
127      })
128    }
129  })
130})

```

Download with file id we can download the respective file.

The screenshot shows two applications side-by-side. On the left, the Postman interface is displayed. A GET request is made to `http://18.222.143.83:3000/items/download/FU-7-ViiZywK`. The 'Authorization' tab is selected, showing a 'Bearer Token' input field with a placeholder: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.' Below it is a 'Token' input field containing a long string of characters. The 'Preview Request' button is visible at the bottom of the request panel. On the right, the Sequel Pro MySQL client is open, connected to the 'sp\_share\_new' database. The 'Tables' section lists 'files', 'group\_membership', and 'GROUPS'. The 'group\_membership' table is currently selected, showing its structure with columns: group\_name, member\_name, isAdded, and timestamp. A single row is visible with the values: go3swetha, 1, 2018-11-17.

```

router.get('/download/:file_id', function (req, res) {
    let xx = req.header('authorization');
    let TokenArray = xx.split(" ");
    let token = TokenArray[1];
    let file_id = connection.escape(req.params.file_id);
    let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
    _bav(function () {
        let login_name = connection.escape(req.local.loginname);
        connection.query("select group_membership.member_name from group_membership where group_membership.group_name in (select item_creator from sp_share_new.items where item_creator = " + login_name + ")");
        if (!error) {
            connection.query("select * from files where files.file_id=" + file_id, function (error, result) {
                let file_path = path.resolve("./uploads/" + result[0].file_name);
                res.download(file_path);
            })
        }
    })
})
}

```

Get will extract all meta data of the files along with its file id(if flat-null else fileid), item id, timestamp, item description of all the files the user has uploaded.

```

router.get('/get', function (req, res) {
    let xx = req.header('authorization');
    let TokenArray = xx.split(" ");
    let token = TokenArray[1];
    let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
    _bav(function (params) {
        let loginname = connection.escape(req.local.loginname);
        connection.query("select item_id AS 'ID',item_type AS 'TYPE',item_description AS 'DESCRIPTION',timestamp AS 'TIMESTAMP',file_id AS 'FILE_ID' from sp_share_new.items where sp_share_new.items.item_creator = " + loginname,
function (error, result) {
        if (!error) {
            res.send(JSON.parse(JSON.stringify(result)));
        }
    })
})
}

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there's a folder named 'swetha-app' containing 20 requests. The main area shows a request configuration for a 'GET' method to 'http://18.222.143.83:3000/items/get'. The 'Body' tab displays a JSON response with 30 items, each having fields like ID, TYPE, DESCRIPTION, TIMESTAMP, and FILE\_ID. The status bar at the bottom indicates a 200 OK status with a 97 ms time and 725 B size.

```

router.get('/get/:item_id', function (req, res) {
  let xx = req.header('authorization');
  let item_id = connection.escape(req.params.item_id);
  let TokenArray = xx.split(" ");
  let token = TokenArray[1];
  let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
  _bav(function (params) {
    let loginname = connection.escape(req.local.loginname);
    connection.query("select item_id AS 'ID',item_type AS
'TYPE',item_description AS 'DESCRIPTION',timestamp AS 'TIMESTAMP' from
sp_share_new.items where sp_share_new.items.item_creator = " + loginname + " and
sp_share_new.items.item_id =" + item_id, function (error, result) {
      if (!error) {
        res.send(JSON.parse(JSON.stringify(result)));
      }
    });
  });
})
}

```

My Workspace

POST http GET http POST http + ... No Environment

GET http://18.222.143.83:3000/items/get/IT-6TvnOKD1A

Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

**TYPE**: Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJsb2dpbm5hbWUiOjnbRz...

Preview Request

Body Cookies Headers (6) Test Results Status: 200 OK Time: 44 ms Size: 324 B Download

Pretty Raw Preview JSON

```

1 [
2   {
3     "ID": "IT-6TvnOKD1A",
4     "TYPE": "FLAT",
5     "DESCRIPTION": "by g333_new Swetha",
6     "TIMESTAMP": "2018-11-18T07:31:49.000Z"
7   }
8 ]

```

Build Browse

View(if a user belongs to a particular group and this view shows all the files he has shared in that group)

My Workspace

POST http POST http POST http GET http POST http GET http GET http GET http + ... No Environment

GET http://18.222.143.83:3000/items/view

Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

**TYPE**: Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJsb2dpbm5hbWUiOjnbRz...

Preview Request

Body Cookies Headers (6) Test Results Status: 200 OK Time: 86 ms Size: 620 B Download

Pretty Raw Preview JSON

```

1 [
2   {
3     "item_id": "IT-Hj2ZkFj0d",
4     "item_type": "FILE",
5     "file_id": "FU-7-ViiZywK",
6     "item_description": "by g333_new Swetha",
7     "item_creator": "g04swetha",
8     "timestamp": "2018-11-18T07:35:20.000Z",
9     "group_name": "documents"
10   },
11   {
12     "item_id": "IT-INSpUnBdY",
13     "item_type": "FILE",
14     "file_id": "FU-7-ViiZywK",
15     "item_description": "by g333_new Swetha",
16     "item_creator": "g01swetha",
17     "timestamp": "2018-11-18T08:00:00.000Z",
18     "group_name": "documents"
19   }
20 ]

```

Build Browse

```

router.get('/view', function (req, res) {
  let xx = req.header('authorization');
}

```

```
let item_id = connection.escape(req.params.item_id);
let TokenArray = xx.split(" ");
let token = TokenArray[1];
let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
_bav(function (params) {
    let loginname = connection.escape(req.local.loginname);
    connection.query("select items.*,item_sharing.group_name from
group_membership join item_sharing on item_sharing.group_name =
group_membership.group_name join items on item_sharing.item_id = items.item_id where
group_membership.member_name = " + loginname, function (error, result) {
        if (!error) {
            res.send(JSON.parse(JSON.stringify(result)));
        }
    })
});
```

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Remove' (highlighted), 'New', 'Import', 'Runner', and other icons. The main area has tabs for 'My Workspace' and 'Invite'. Below the tabs, there's a search bar with 'Filter' and a 'No Environment' dropdown.

The left sidebar lists a collection named 'Collections' containing the following requests:

- POST localhost:3000/auth/signup
- POST localhost:3000/groups/join
- POST localhost:3000/items/upload
- GET localhost:3000/items/get
- POST localhost:3000/groups/new
- POST http://18.222.143.83:3000/auth/signin
- POST localhost:3000/items/put
- POST localhost:3000/items/remove
- POST http://18.222.143.83:3000/auth/makeUserAdmin
- POST http://18.222.143.83:3000/auth/signup
- POST http://18.222.143.83:3000/auth/signin
- POST http://18.222.143.83:3000/auth/makeUserActive
- POST http://18.222.143.83:3000/groups/new
- POST http://18.222.143.83:3000/groups/makeGroupActive
- POST http://18.222.143.83:3000/groups/join
- POST http://18.222.143.83:3000/groups/makeGroupActive
- POST http://18.222.143.83:3000/groups/makeGroupMemberActive
- POST http://18.222.143.83:3000/items/upload
- GET http://18.222.143.83:3000/items/download/FU-7-VliZywK
- POST http://18.222.143.83:3000/items/put
- GET http://18.222.143.83:3000/items/get
- GET http://18.222.143.83:3000/items/get/IT-6TvnOKD1A
- GET http://18.222.143.83:3000/items/view

The right side shows a detailed view of a POST request to 'http://18.222.143.83:3000/items/remove'. The 'Body' tab is selected, showing a JSON payload: `{"item_id": "IT-6TvnOKD1A"}`. The response section shows a status of 200 OK with a size of 206 B.

Sun 2:04 AM

(MySQL 5.7.24-0ubuntu0.18.04.1) localhost/sp\_share\_new/items

SSH Connected

Select Database Structure Content Triggers Table Info Query

Table History Users Console

TABLE INFORMATION

created: 11/18/18 updated: 11/18/18 engine: InnoDB rows: 4 size: 16.0 KiB encoding: utf8mb4

4 rows in table

```

134     router.post('/remove', function (req, res) {
135         let xx = req.header('authorization');
136         let tokenArray = xx.split(" ");
137         let token = TokenArray[1];
138         let itemBody = req.body;
139         let item_id = connection.escape(req.body.item_id);
140
141         let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
142         _bav(function () {
143             let loginname = connection.escape(req.local.loginname);
144             connection.query("SELECT items.file_id,files.file_name FROM sp_share_new.items items, sp_share_new.files files WHERE items.file_id = files.file_id AND items.item_id = " + item_id, function (error, result) {
145                 if (!error) {
146                     var file_res = result;
147                     connection.query("delete from item_sharing where item_sharing.item_id =" + item_id, function (error, result) {
148                         if (!error) {
149                             connection.query("delete from items where item_id =" + item_id, function (error, result) {
150                                 if (!error) {
151                                     if (file_res.length > 0) {
152                                         var file_id = result[0].file_id;
153                                         var file_name = result[0].file_name;
154                                         connection.query("Delete files from files where files.file_id =" + connection.escape(file_id), function (error, result) {
155                                             res.sendStatus(200);
156                                             let file_path = "/uploads/" + file_name;
157                                             fs.unlink(file_paths, function (err) {
158                                                 if (err && err.code == 'ENOENT') {
159                                                     // file doesn't exist
160                                                     console.info("file doesn't exist, won't remove it.");
161                                                 } else if (err) {
162                                                     // other errors, e.g. maybe we don't have enough permission
163                                                     console.error("Error occurred while trying to remove file");
164                                                 } else {
165                                                     console.info('removed');
166                                                 }
167                                             });
168                                         }
169                                     } else {
170                                         res.sendStatus(200);
171                                     }
172                                 }
173                             }
174                         }
175                     }
176                 }
177             }
178         );
179     }
180 
```

## Limit on space and file consumption

1. limit-group-usage(Group size at max)
2. limit-user-usage(User max data )
3. limit-file-usage(File Size max)

## 1. limit-group-usage(Group size at max)

Postman screenshot showing a POST request to http://18.222.143.83:3000/groups/new with the body:

```
1 {"group_name": "exams"}
```

MySQL Workbench screenshot showing the entity\_storage\_rules\_with\_usage table:

group_name	user_name	limit	usage	entity_type
NULL	go3swetha	001000	000000	U
documents	go3swetha	000100	000010	GU
documents	NULL	001020	000000	G
NULL	NULL	000011	000100	F
exams	NULL	001000	000000	G

MySQL Workbench Table Information:

- created: 11/18/18
- updated: 11/18/18
- engine: InnoDB
- rows: 5
- size: 16.0 KiB
- encoding: utf8mb4

The screenshot shows a developer's desktop environment with three main windows open:

- Postman:** A tool for making API requests. The left sidebar lists various API endpoints with their methods and URLs. The right panel shows a POST request to `http://18.222.143.83:3000/storage/limit-group-usage` with a JSON body containing `{"group_name": "exams", "group_limit": "2000"}`. The response status is 200 OK.
- Sequel Pro:** A MySQL database management tool. The top menu bar includes File, Edit, View, Database, Table, Bundles, Window, Help, and a status bar showing "Sun 12:58 PM". The main pane displays the contents of the `entity_storage_rules_with_usage` table, which has columns: group\_name, user\_name, limit, usage, and entity\_type. The data shows rows for documents, exams, and other entities with specific values for each column.
- Browser:** A standard web browser window showing a page with a table of information. The table includes columns like created, updated, engine, rows, size, and encoding. The "rows" value is listed as 5 rows in the table.

```

module.exports = function (connection, bcrypt, jwt, config, shortid) {
  router.post('/limit-group-usage', function (req, res) {
    let xx = req.header('authorization');
    let TokenArray = xx.split(" ");
    let token = TokenArray[1];
    let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
    let group_name = req.body.group_name;
    let group_limit = req.body.group_limit;
    _bav(function (params) {
      connection.query("UPDATE `sp_share_new` `entity_storage_rules_with_usage` SET `limit`=" + connection.escape(group_limit) + " WHERE `group_name`=" + connection.escape(group_name));
      if (error) {
        res.sendStatus(200);
      } else {
        res.sendStatus(500);
      }
    });
  });
}

```

## 2. limit-user-usage(User max data )

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', and 'Invite' buttons. The main workspace shows a list of recent requests on the left and a detailed view of a current request on the right.

**Request Details:**

- Method: POST
- URL: <http://18.222.143.83:3000/auth/signup>
- Headers: (1)
- Body (raw, JSON):

```

1 {
2   "loginname": "newuser3",
3   "password": "new",
4   "actualname": "user"
5 }
```

**Response View:**

- Status: 200 OK
- Time: 265 ms
- Size: 206 B
- Body (Pretty):

```
1  OK
```

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', and 'My Workspace'. The workspace title is 'My Workspace' with an 'Invite' button. The left sidebar has a 'Filter' search bar and tabs for 'History' and 'Collections', with 'Collections' currently selected. A list of API requests is visible on the left.

The main area displays a POST request to `http://18.222.143.83:3000/storage/limit-user-usage`. The request method is set to 'POST'. The 'Body' tab is active, showing the following JSON payload:

```
1. {"user_name": "newuser3",  
2. "user_limit": "3000"}
```

Below the request, the 'Body' tab is selected, showing the raw JSON response: `i 1 OK`.

Sequel Pro - MySQL 5.7.24-Dubuntu0.18.04.1 localhost/sp\_share\_new/entity\_storage\_rules\_with\_usage

Tables:

group_name	user_name	limit	usage	entity_type
NULL	go3swetha	001000	000000	U
documents	go3swetha	000100	000010	GU
documents	NULL	001020	000000	G
NULL	NULL	000011	000100	F
exams	NULL	002000	000000	G
NULL	newuser3	003000	000000	U

TABLE INFORMATION

- created: 11/18/18
- updated: 11/18/18
- engine: InnoDB
- rows: 6
- size: 16.0 KiB
- encoding: utf8mb4

6 rows in table

```
router.post('/limit-user-usage', function (req, res) {
  let xx = req.header('authorization');
  let TokenArray = xx.split(" ");
  let token = TokenArray[1];
  let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
  let user_name = req.body.user_name;
  let user_limit = req.body.user_limit;
  _bav(function (params) {
    connection.query("UPDATE `sp_share_new`.`entity_storage_rules_with_usage` SET `limit`=" + connection.escape(user_limit) + " WHERE `user_name`=" + connection.escape(user_name));
    if (!error) {
      res.sendStatus(200);
    } else {
      res.sendStatus(500);
    }
  });
});
```

### 3 limit-file-usage(File Size max)

```
router.post('/limit-file-usage', function (req, res) {
  let xx = req.header('authorization');
  let TokenArray = xx.split(" ");
  let token = TokenArray[1];
  let _bav = new BAV(connection, bcrypt, jwt, config, token, req, res);
  let file_limit = req.body.file_limit;
  _bav(function (params) {
    connection.query("UPDATE `sp_share_new`.`entity_storage_rules_with_usage` SET `limit`=" + connection.escape(file_limit) + " WHERE `group_name` IS NULL AND `user_name`=" + connection.escape(user_name));
    if (!error) {
      res.sendStatus(200);
    } else {
      res.sendStatus(500);
    }
  });
});
```

return router;

Sequel Pro

(MySQL 5.7.24-Dubuntu0.18.04.1) localhost/sp\_share\_new/entity\_storage\_rules\_with\_usage

Tables:

group_name	user_name	limit	usage	entity_type
NULL	go3swetha	001000	000000	U
documents	go3swetha	001010	000010	GU
documents	NULL	001020	000000	G
NULL	NULL	000011	000100	F
exams	NULL	002000	000000	G
NULL	newuser3	003000	000000	U

TABLE INFORMATION

- created: 11/18/18
- updated: 11/18/18
- engine: InnoDB
- rows: 6
- size: 16.0 KiB
- encoding: utf8mb4

6 rows in table; 1 row selected

New Import Runner

My Workspace

POST http://18.222.143.83:3000/storage/limit-file-usage

Body

```
{
  "file_limit": "3000"
}
```

Status: 200 OK Time: 136 ms Size: 206 B

Body Cookies Headers (6) Test Results

Pretty Raw Preview Auto

After

Sequel Pro File Edit View Database Table Bundles Window Help (MySQL 5.7.24-Dubuntu0.18.04.1) localhost/sp\_share\_new/entity\_storage\_rules\_with\_usage Sun 1:17 PM SSH Connected

Select Database Structure Content Relations Triggers Table Info Query Table History Users Console Filter

entity\_storage\_rules\_with\_usage

group_name	user_name	limit	usage	entity_type
NULL	go3swetha	001000	000000	U
documents	go3swetha	000100	000010	GU
documents	NULL	001020	000000	G
NULL	NULL	003000	000100	F
exams	NULL	002000	000000	G
NULL	newuser3	003000	000000	U

TABLE INFORMATION

- created: 11/18/18
- updated: 11/18/18
- engine: InnoDB
- rows: 6
- size: 16.0 KiB
- encoding: utf8mb4

6 rows in table; 1 row selected

The screenshot shows the Sequel Pro application interface for MySQL. The main window displays the 'entity\_storage\_rules\_with\_usage' table with the following data:

group_name	user_name	limit	usage	entity_type
NULL	go3swetha	001000	000000	U
documents	go3swetha	000100	000010	GU
documents	NULL	001020	000000	G
NULL	NULL	003000	000100	F
exams	NULL	002000	000000	G
NULL	newuser3	003000	000000	U

Below the table, the 'TABLE INFORMATION' section provides details about the table's creation and structure:

- created: 11/18/18
- updated: 11/18/18
- engine: InnoDB
- rows: 6
- size: 16.0 KiB
- encoding: utf8mb4

The status bar at the bottom indicates there are 6 rows in the table and 1 row is selected.