**TEST CASE**

**Tech stack**

**DB**: mysql

**Programming Language**: Node.js, Express

**Libraries**: fs-extra, jwt, multer, bcrypt

{{local}} = <http://127.0.0.1:889/api>

* Because this type of system will use a lot of memory and server space. Cache is default managed from the fs, fs-extra. Make sure you are not interrupted on cache issue.
* Because I have less time, thus I make it to work more than I make it to sustainable.

1. **Register**

* Register is a must part in this project. Simple information
* Automatically sync user profile after success unless user has no space on server
* Automatically login after success

Path: {{local}}/register

Method: Post

PayLoad:

{

    "userName":"test4",

    "userEmail": "test4@ymail.la",

    "userPassword": "123",

    "passwordConfirm":"123"

}

1. **Login**

* Login is a required.

Path: {{local}}/login

Method: POST

Payload:

{

    "userEmail":"test2@ymail.la",

    "Password":"123"

}

* Response return all necessary user information including user profile the important part.
* If user profile is lost, user will not go to first page because don’t has user space on server.
* Frontend need to prompt to user and do a manual sync the profile.

"userProfileId": 96,

"userProfile": "VSHARE202410534132",

* If this profile lost will be null.
* Profile ID will required on query information or query my files.

1. **Sync userProfile**

* Sync profile is purpose to reserve the server space.

Path: {{local}}/sync-profile

Method: GET

Payload: None

Token: required

1. **Create Folder / Directory**

* Make directory on user’s space on server.
* The directory will be empty after created

Path: {{local}}/create-folder

Method: POST

Token: required

Payload: {

    "folderName":"Test2",

    "parentKey":108

}

* parentKey: is the parent node where user create folder / directory.
* If user did not has any folder or director, that means user still stay at profile space. Use the profile Id instead.

1. **Rename Folder / Directory**

* Rename or change folder name if unsatisfied

Path: {{local}}/rename-folder

Method: PUT

Token: required

Payload: {

    "folderId":82,

    "folderNewName":"Test5"

}

* Enter the newname and folderId which want to rename it.
* Rename a directory will affect all children and path directories to each child (I already handle). Make sure it is fully completed.

1. **Delete Folder / Directory**

* Delete unwanted folder
* Delete folder will affect all children lost.
* Frontend handle folderId to delete. Put to the params. (Number)

Path: {{local}}/delete-folder/:folderId

Method: DELETE

Payload: NONE

Token: required

1. **Create File**

* Create file mean, user is already access to a specific folder or directory.
* Frontend already know where the user is.
* Frontend handle the parentId and put to the params. (parentId is a number)

Path: {{local}}/upload-file/:parentId

Method: POST

Token: required

Payload: Use multiparts/form-data

File key pair = filename

* Support 1 or more files.

1. **Rename File**

* User can rename the file if unsatisfied

Path: {{local}}/rename-file

Method: PUT

Token: required

Paylaod:

{

    "fileId":50,

    "newFileName":"ModifyName"

}

1. **Delete File**

* User can remove unwanted file
* Delete file will affect lost permanently
* Put fileId on the param (Number)

Path: {{local}}/delete-file/:fileId

Method: DELETE

Payload: none

Token: required

1. **Download File**

* User can use browser to download the file specifically

Path: {{local}}/download/:fileId

Method: GET

Payload: none

Token: required

1. **Query My Files**

* Get my file to display on monitor
* User should be locate in a specific location or space such like inside a folder or directory or even in the user profile space.
* Files or subdirectories will display to the right parent node only.
* User access to a subfolder, It mean user change location or move to new space. The files or subDirectories will display if any.
* Frontend handle parentKey and put to the params to get new files or dirs base on the right node.

Path: {{local}}/get-files/:parentKey

Method: GET

Payload: none

Token: required

* File or dirs. Will show in remarked between my file and file shared with me.
* My file will display: sharing members if any

1. **Share Folder / Directory / Files**

Rule:

1. A share to B, C,…N (All member can see file from A)
2. B,C,…N created new file or directory. (Only specific member & A can see new file)
3. B,C,…N share link from A to Z. Z can see only file from share user (shared\_by) and files from original A.
4. A share to Y in finally, Y can see only file from A.

Practice:

A: can see an empty folder or has content. Children will automatic shared base on parent

A: can share to B,C,…N correct.

B,C,..N: created new file will automatic sync to A. View flow is : (A,C), (A,B),..(A,N)

A: share to Y. file display correctly. View flow is (A,Y)

Path: {{local}}/share

Method: POST

Token: required

Payload: {

    "fileType":0,

    "typeId":108,

    "shareTo":33

}

Explain:

fileType: 0|1|2. (0 is folder, 1 is File, 2 less use. 1% usage file shared by someone to this user.

fileId: if user shares on the folder, it is folder ID, if user shares file, it is file ID

shareTo: member will see this folder.

1. **UnShare Folder / Directory / Files**

* User can stop sharing the file created by himself.
* If user stop sharing a folder will affect all children stopped. Children created by himself.

Path: {{local}}/unshare

Method: POST

Token: required

Payload:

{

    "fileType": 1,

    "typeId": 50

}