Manual for Sync System

I Installation

Node

For Linux or MacOS it is recommended first to install the packet manager Homebrew, in order to install Node.

Once you have installed Homebrew use it to install Node. For Windows the process of installing Node is straight forward.

Git

Download and install Git in order to download the repository, I recommend the Git Bash but just installing Git should be sufficient.

Sync module

Download the module by entering the following commands on the desire directory.

```
>Git clone https://rresendez@bitbucket.org/rresendez/sync_module.git
```

Once downloaded, move to the repository and install the dependencies by typing the following command:

```
>Npm install
```

Make sure to execute that command on the folder where the `package.json` file is.

Upload folder

For Linux users make sure the upload folder has permission to write read and execute.

Database connection

Under the `config` folder there are several files. The database files, set up the connection for MySQL. The file named database.js holds the configuration for the database. You can switch in between databases by renaming the current file and renaming the desire file to `database.js`. The same logic applies for the passport.js file which handles local authentication.

Required Tables

There is a list of required tables in order for the program to work correctly. The schema for the tables is presented in the schema section below.

1) Local authentication table.

2) Temporary id table.

3) Csv log table.

II Schemas

tbl_log_csv

```
CREATE TABLE `tbl_log_csv` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` datetime DEFAULT CURRENT_TIMESTAMP,
  `username` varchar(45) NOT NULL,
  `create_entry` int(11) NOT NULL DEFAULT '0',
  `del_non_4` int(11) NOT NULL DEFAULT '0',
  `time_non_match` int(11) NOT NULL DEFAULT '0',
  `time_match` int(11) NOT NULL DEFAULT '0',
  `db_orphan` int(11) NOT NULL DEFAULT '0',
  `state_update` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

tbl_tmp_id

```
CREATE TABLE `tbl_tmp_id` (
  `unique_id` int(11) NOT NULL AUTO_INCREMENT,
  `id` int(11) NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`unique_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

tbl_users_serv

```
CREATE TABLE `tbl_users_serv` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `password` char(60) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `username_UNIQUE` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Authentication

The way the application is designed, it will only allow the first 3 users to create new users.

If you don't have any user created for the `tbl_users_serv` then you will have to create them.

In order to access sign up route you will have to go to `~/app/routes.js .`

Go to line **156** and delete the `isLoggedIn` argument.

Then go to line **160** change the if statement to TRUE.

Once you do this you can create users. To prevent users to further create users, go back to the `~/app/routes.js` and undo your changes.

Running the server

Once everything is in place and installed, you can go ahead and execute the server.

To do this go into root folder where package.json resides, and type the following command:

```
>node server
```

Alternately you can use nodemon which will reboot automatically in case of any failure or any applied change on the code .

```
>nodemon server
```

Once started the server will show the message "Magic happens on port 8081". If so, now you can go to your local server 8081 port in any browser and start using the application.

III Using the Application

Login

Once you have the server listening you can go ahead and visit the index page. Here you will find local login and local signup, unless you deactivated the isLoggedIn argument this won't allow you to access the route. Go to Local Login and enter your credentials. You won't be able to access any route unless you are logged in.

Main Menu

The main menu will show you first your ID number and Username then the instructions on how to use the application.

Administrators

If your user Id is less than 4 meaning that is one of the first 3 users created you will be able to see at the bottom a Create user button and a Second Stage button.

Second Stage Button

This is for test purposes, a regular user wouldn't need to skip to the second stage. But as an administrator you can do this.

First Stage

On first stage you will land on the upload route where you are supposed to upload the first csv file containing the schedule, click chose file select the correct file and then hit submit .

The application won't let you upload any file that doesn't correspond to the schedule column format.

Database Update

This page will show you the name of the file you just uploaded the size in Bytes and the type of file. This is to ensure you have uploaded the correct file, if so proceed to execute.

Database cleanup

This page will show only a logout button and a clean button. What the clean button does is use the information recollected form the first stage to now which user ID and dates where entered in the schedule, it will take those values and compare it against the database. If there is an orphan entry in the database in that range of dates, the cleaning process will delete it. Finally it will truncate the table so it can be reused.

Second Stage

This is the final stage of the sync module, here you will choose the csv file containing the states and upload it. Just like the first stage the application won't allow you to upload any file that doesn't match the state csv file columns.

Once uploaded go ahead and execute.

Pop up error

If any error related to the application process occurs, the system will show you a pop up box with an error message, generally the reason for the failure can be found in the console.

File structure III

Sync module actually follows node/express architecture, so if you are familiar with this kinds of framework the file structure should be very familiar.

On the root directory you can will find:

`.gitignore` – Contains the type of files gituhub should ignore while pushing.

`package.json` – Contains the dependencies needed for the application.

`server.js` – Contains the server configuration and the reference for the different routes, it also sets up the location for static file folder.

`/views` – Contains the ejs files which are similar to html and has the front end html code.

`/upload` – Contains the temporary files to be uploaded.

`/scripts` – Contains scripts to create the necessary tables.

`/public` – Contains the static files needed for the website.

`/node_modules` – Contains the modules installed for node.

`/export_test` – Its purpose is to hold test csv files.

`/config` – Contains the db configuration files and the passport configuration files, passport is the local authentication system it also handles password encryption.

`/app` – Contains most of the logic behind the routes and the processing software.

- `bupload.js` – Controls the routes for uploading the second stage document.
- `clean.js` – Controls the route that executes the cleaning.
- `routes.js` – Contains most of the routes used such as login, index, logs etc...
- `sql.js` - Contains the first stage logic and routes, it is the most complex file .
- `sql2.js`- Contains the second stage logic and routes, it is less complex than sql.js .

Contact IV

rresendez@rigoresendez.com