Shopping Cart Project

- Create a folder for project
- express --hbs :- to download essential things
  npm install :- to install all dependencies.
  npm install nodemon :- to install nodemon server, that automatically restarts when we make changes to the code.
  we can start the project by
  npm start -
  nodemon bin/www
  or change node to nodemon in the file "package.json" and type npm start.

Creating/editing "index.hbs" file

From bootstrap, copy the starter template.

- add a navbar.
  edit navbar according to your preference
- After that add cards for items in your shopping site.
- On the <header> ... </header> part add navbar there.

default ആയിട്ട് stylesheet ൽ body ക്ക് 50px padding കൊടുത്തിട്ടുണ്ട്. If you don't need it we can remove it. (when we create project using express --hbs)

* below that header add a new <section> for listing items
* ~~add a container~~
* create a row.
* ~~Create~~ inside that 'row' div. add another div. <class="col-md-3"> to list 4 items/row. (3*4=12).
* Add a container div (which helps to give some space on both (left & right side), also gave 'mt-5' to give ~~top~~ margin-top.

* Manually its impossible to add 100's of cards/lists in a site, so we need to use hbs. (using hbs for loop).

* For that, on 'index.js' create an array
After creating that array, pass it to the index.hbs page.
After reaching the index.hbs, ~~using for~~ loop display it using loop.

* Setting layout like navbar and other to a common file (because its not practical to create navbar for each and every page, for that it is created in a common file and calls where its needed.
* for that create a layout folder in views folder and inside layout folder create layout.hbs.
* Also create a folder called partials in the views folder for ~~partial layouts~~
* And all the partial layouts are called in "layout.hbs" and this "layout.hbs" is called in pages wherever needed.
* Then to set template engine in app.js. → hbs.engine

app.engine('hbs', hbs({extname: 'hbs', defaultLayout: 'layout', layoutsDir: __dirname + '/views/layout/', partialsDir: __dirname + '/views/partials/'}))

Dont need to byheart just ~~go~~ google it.

* Also install hbs module using npm

npm install express-handlebars.

* And use the library by assigning it into the app.js by require().

var hbs = require('express-handlebars').

~~* We don't need all html~~

* We only need the essential section of the code in files. All other common codes/layouts are written in layout.hbs. and we call the body in that file using {{{ body }}}

* After that the header from layout is removed and individual (partials) header is created for admin & user and then calls it in the layout.hbs.

'{{> admin-header}}'

will display the admin-header, but we need to use ~~the~~ an if-condition to check whether we need to use admin/user header.

* passing a ~~value~~ parameter 'admin: true' in index.js to ~~check~~ get the value for checking if condition. (this done just to check the working, actually it is passed from backend)

14/9/23

* Setting route for admin & user
* In this project index.js is changed to user.js & user.js to admin.js also don't forget to change it in the app.js file.
* Then render an HTML page for admin
* Changing the navbar contents to necessary options.
* On admin.js, add a variable for {admin:true}

For admin

* Create page for listing, adding products. Create it in the folder admin in views.
* For admin, we use table view instead of card view to list products.
* We pass values to admin.js (like we add product variable in user.js.
* then on view-products.hbs, we add a loop to display that data from admin.js to the page.
* We need to give image in <img> tag
* Adding a button for adding new products which contain a link to page called add-product.
* On admin.js, set route for add-product.
* create add-product page
* While uploading image, we use input type='file'. To

To upload these image to server, we need to give an attribute to form called enctype and choose multipart/form-data.

* set method="post", and action="/admin/add-product"

* set another route method for POST on admin.js for getting the datas we upload on add-product.

Note: * In normal case req.body will return the datas.

* But if a file is there, we need to use a module called for fileupload called "express-fileupload"

* Don't forget to require('express-fileupload') on app.js and to use on the middleware i.e app.use(fileUpload).