/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ITE5315 – Assignment 2

I declare that this assignment is my own work in accordance with Humber Academic Policy.

\* No part of this assignment has been copied manually or electronically from any other source (including web sites) or distributed to other students.

\*

\* Name: Sandeep Das

Student ID: N01472825

Date: 29th October 2022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Step 1:

Graphical user interface, application

Description automatically generated

Step 2:

Text

Description automatically generated

Step 3:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Step 4:

Text

Description automatically generated

Setting the root path for the app

1. Firstly in the app we are setting the static root path as the public folder inside our app.

Then main.hbs is located inside the /public.

The default behaviour of handlebars is that whenever a handlebar is called, the control first goes to main.hbs located inside our layout folder and the variables passed are injected into the hbs document. In case we rename main.hbs to anything else. The app wont work as expected.

Then the {{body}} is replaced by the handlebar passed as another parameter after that passed handlebar is substituted with all the variables.

The variables passed as a JS object keys should match the handlebar variables exactly to work.

1. Error is matched with a wildcard route at the end after defining all valid routes. The message parameter gets injected into the error handlebar and title parameter gets injected to the main handlebar. The {{body} of main.hbs is substituted by error.hbs
2. If we comment out the below line. The CSS will not load as the static content of CSS to be served remains unknown to our app

app.use(express.static(path.join(\_\_dirname, 'public')));

var express = require('express');                               //Fetch express module to application

var path = require('path');                                     //Fetch path module to application

var app = express();                                            //Initialize app variable with express

const exphbs = require('express-handlebars');                   //Fetch express-handlebars module for application

const port = process.env.port || 3000;                          //Serve web app of env variable port if defined else on 3000

app.use(express.static(path.join(\_\_dirname, 'public')));        //Configure web app to use application root path appended with public as the folder to serve static content

app.engine('.hbs', exphbs.engine({ extname: 'hbs' }))           //setiing the app engine with .hbs key and value as exphbs.engine({ extname: 'hbs' })

app.set('view engine', 'hbs');                                  //Set app parameter view engine to hbs value

app.get('/', function (req, res) {                              //Set root route

    res.render('index', { title: 'Express' });                  //Redirect to index handlebar and pass title as express and inject the value of title in all relevant handlebars

});

app.get('/users', function (req, res) {                         //Set /users route

    res.send('respond with a resource');                        //Just pass text as response

});

app.get('\*', function (req, res) {                              //Set route for undefined path

    res.render('error', { title: 'Error', message: 'Wrong Route' });        //Call the error handlebar and pass message attribute for handlebars

});

app.listen(port, () => {

    console.log(`Example app listening at http://localhost:${port}`)        //initialize the server

})

Step 5:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Step 6:

/data route

Logo, company name

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

/data/isbn/:index

Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

'/data/search/title'

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, text, application, Word

Description automatically generated

Step 7:

Graphical user interface

Description automatically generated

Step 8:

Text

Description automatically generated

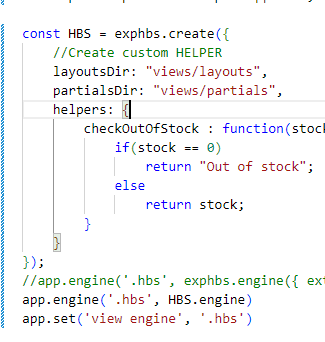
Step 9:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, website

Description automatically generated



Step 10:

a.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

b.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

C. I did find partials useful for rendering static content that does not change as our main content changes.