Project Implementation Navigation

The purpose of this document is to help the readers of my code understand what on earth I have written, and most importantly, how I have gone around writing it

Overview

The project, movies reviewing site, is on the MERN stack. The project uses bootstrap (front-end), ExpressJS (middleware), NodeJS (server) and MongoDB (database). Moreover, a templating engine, called handlebars, has also been used (which introduces all the complexity in the project and compels me to document it)

Remember, app.js is the entry point for the project

Serving Bootstrap files – Easy Difficulty

This part is very easy; using ExpressJS framework, we serve the bootstrap files. The basic syntax used is: app.use('/css', express.static('./public/home/css')) for the home and app.use('/css', express.static('./public/admin/css')) for the admin. Hence, we created a middleware and served all the files (css, js, img, icon, fonts) statically

Be noted that the template sends request to the server (from the browser) for these static files, and the files get served automatically. You don't have to arrange any paths or anything else. Just put all the template files (js, fonts, css, img, icon) in /public/admin and in /public/home

At one point, the static files were not being served. This consistently happened for all sub-routes on a main route. For example, after going to localhost:64000/admin and localhost:64000/home files are served perfectly. But on the sub-routes, like localhost:64000/admin/view-reviews and localhost:64000/admin/add-genre, static files were not served. All you have to do here is add absolute path (not relative) request to the server. You do this by prepending just one, single forward slash ("/") to the lines, like <script src = "js/main/jquery.3.5.min)> so that they become <script src = "/js/main/jquery.3.5.min)>. Moreover, you do the same to the css file requests in the href of the same files (in the head of the html)

Managing Routes – Medium Difficulty

This one is relatively trickier but not so much. The complexity here is added by both ExpressJS as well as handlebars

Using ExpressJS, we make sure that home and admin routes are managed separately in separate files at routes/admin/main.js and routes/home/main.js by the help of Express router that can be included by writing ExpressJS.Router(). It is also exported from those files

Handlebars is a little tricky. The following points must be kept in mind while using handlebars

- **Views** – Handlebars framework always looks (by default) into views directory. So, we create that at first. Inside there, we have layouts directory (use exact same name), partials directory (use exact same name), admin directory and home directory
- **Layouts** – Handlebars framework makes you create layouts. When creating routes, on top of all three route files (app.js, routes/admin/main.js and routes/home/mian.js), you have to inform what the default layout of all the routes (that follow in this file) is going to be. That layout is basically the code that will be hardcoded to all the files that are set up on this route

  For example, in the home side of the projects, all the files inherit navigation and footer. This is because navigation and footer are set-up in view/layouts/index-home.handlebars. In this file, where you see {{> *file-name*}} means that handlebars should peek into views/partials/file-name and include that file's code here (navigation code / footer code etc.). Whereas, if you see {{{ body }}} in this file, it means that handlebars should not hardcode anything here (like footer or navigation) from some file, rather code will be added here dynamically (as we will serve the file when somebody comes on this route). Hence, to add code dynamically in place of {{{ body }}}, we simply serve some file on this route

  For example, on localhost:64000/home, we serve page-content.handlebars file (from views/home/page-content) and on localhost:64000/faq, we serve faq.handlebars file (from

views/home/faq). The content dynamically served on these files will be added in place of {{{ body }}} of the layouts file

Remember, the main file of the template (index.html) has to set-up in views/layouts folder for the admin as well as for the home

- **Partials** – These are the code snippets that you want to include (hardcode) in every file on a specific route. For example, view/partials/admin-sidebar.handlebars is added to all the files of the admin/ route