MERN Stack Course

Express – quick web server

Cors – Cross Origin Resource Sharing: helps to make the right corrections with less errors.

MongoDB dependency – allows interaction with the mongodb databade

Dotenv – loads environmental variables from a dotenv file in the Process dotenv, which makes development simpler.

Nodemon – automatically refreshes the application made using Node.js when file changes are detected.

Bcrypt – used for encrypting the stored passwords

Express async handler – handles errors

JSON Web token – creates web token for users

Npm I -g(installs globally)

**Inside the package.json(right under “main”):**

{

"type": "module", (allows the use of Imports from ES6)

}

**Good practices when setting a route to an API:**

app.use("/api(the folder specifices that it is an API)/v1(the API version)/restaurants(the name of the API)")

app.use("\*", (req, res) => res.status(404).json({error: "not found"})) (**The “\*” in the route works as an Wildcard, so that if the user tries to access any routes that aren’t specified in the backend, it will return an error)**

**URI = Uniform Resource Identifier : works as an identifier for the content located in a server. Its childs are the URN(the name of the resource) and the URL(the locator of the resource, the most commonly used).**

**REACT**

-vDom (virtual DOM helps loading the web pages faster, as it pre-loads the UI of the page in your RAM memory; such as when you click certain links that redirects to different pages, it does so seemingly without any loading)

-Reusable Components (the concept of components helps for faster coding, as you can make a complicated component, and just replicate it as many times as you want, without having to code it all over again)

-The fastest framework available.

**Node.js**

-JS Runtime Environment built on Chrome’s V8 Engine

-Basically a scalable web server(the heart and brain of our applications; it manages the APIs, connects Front-Back-Database)

-Node Package Manager(makes coding simpler by making several code packages available to download and integrate)

-Develop real time systems

**Express**

-Node.js Framework

-Powerful routing API(application programming interface: used for carrying information from the back-end to the front-end)

-Detailed documentation

-High Performance

-Many third party plugins

**MongoDB**

-Cross-platform, No SQL and document-oriented

-Always on

-Highly Scalable

-Flexible Schema

**Commands:**

-GET (fetches data from the database or API) ex:/api/notes

-POST(sends data to the DB) ex:/api/notes

-PUT(works as an “update” command, when you want to change some pre-existing data, which can be done by providing the ID) ex:/api/notes/:id

-DELETE(it deletes data from the DB through an ID) ex:/api/notes/:id

**JSX**

-Part of React.js

-Similar to HTML

-Its function is to help integrate JS command lines with plain text, much alike HTML

***npm i react-bootstrap (installs React Bootstrap)***

**Importing a command:**

To auto-import a command from a Library, select the command that needs importing (by highlighting it) and press “CTRL + SPACE” then press ENTER.

**When using BrowserRouter:**

Always use the “element” command instead of “Component, as it has been deprecated and no longer works.

The routes also **need**  to be nested in a *<Routes>* tag, and have each element nested in a *React Fragment*, like this: **<a fragment />**  
Example:

<Routes>

        <Route path="/" element={<LandingPage />} exact />

        <Route path="/mynotes" element={<MyNotes />} />

</Routes>

**Mongoose notes:**

useNewUrlParser, useUnifiedTopology, useFindAndModify, and useCreateIndex are no longer supported options. Mongoose 6 always behaves as if useNewUrlParser, useUnifiedTopology, and useCreateIndex are true, and useFindAndModify is false. Please remove these options from your code.