**MERN-FULL STACK**

**Training by Arun (20-Nov-2023 to 19-Dec-2023)**

Code Base : [arunprabu/mern-stack-bosch-nov2023 (github.com)](https://github.com/arunprabu/mern-stack-bosch-nov2023)

**Day 1:**

======

Introduction

------

1. Course Plan

MERN Stack (80 Hours)

-

Full Stack JavaScript

Front End (10 Half Days)

React JS

Back End (10 Half Days)

NodeJS + Express JS + MongoDB / MySQL

Hands-on

-------

\* Explain => Coding => Demo => Try it out => (Tiny Todos) => Review

3 Projects

-------

1 App React JS

2 Apps NodeJS

2. Softwares Required

Node JS v 16.15+

Visual Studio Code Editor

Browser (Chrome / FF / Edge / Opera / Brave)

REST API Client (Postman / ARC )

Mongo DB Community Edition v 6.0.11

MySQL Community Edition v 8+

3. Skillset

HTML, CSS

JS

4. Materials

1. Codebases of all 3 projects in github.com

2. 2 E-Books in PDF format - FREE

--------------

Learning Curve

-------------

Essential JS (2 Hours)

Essential ES2015 (~1.5 Hours )

NodeJS Ecosystem (~45 mins)

React App Development (38 Hours)

NodeJS, ExpressJS, MongoDB + MySQL (40 Hours)

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Webapps

--------

  1. Server Side Rendering Apps (SSR) / Traditional Webapps

      \* Pages will refresh / reload

      Codebase

      ---------

        index.php

          HTML, CSS, JS + Back End (PHP) + DB Query (SQL)

        about.php

          HTML, CSS, JS + Back End (PHP) + DB Query (SQL)

      Server

    -------

        1 Server

        \* port number 8080

        \* localhost:8080/index.php

        Response: HTML, CSS, JS + Data

      Adv

      ----

        1. Search Engine Friendly

        2. Initial Loading will NOT take more time

      Disadv

      -----

        1. Server has to do the heavy-lifting of going thru front end codes also

        2. Slower Webapps

      3. Bandwidth consumption is more

        4. Expensive

        5. Bad Developer Experience

        6. Bad User Experience

  2. Client Side Rendering Apps (CSR) / Modern Webapps = SPA + RESTful API

      \* Pages will NOT refresh / reload

      Codebase

      ---------

        2 Apps

        -------

        1 Front End App a.k.a Single Page Apps (SPA)

            HTML, CSS, JS

          1 Back End App a.k.a RESTful API

              Back End (NodeJS, ExpressJS) + DB Query (MongoDB / SQL)

      Servers

      -------

        1 Server for the Front End App

        \* port number: 4000

      \* Requesting: localhost:4000   (publicly exposed)

        \* Response: HTML, CSS, JS, (without Data)

        1 Server for the Back End App

          \* port number: 8081

          \* Requesting: localhost:8081/users

          \* Response: Data (JSON)

    Adv

      ----

        1. Workload is shared between 2 servers.

        2. Faster Webapps

        3. Bandwidth consumption is less

        4. Less Expensive

        5. Better Developer Experience

        6. Better User Experience

      Disadv

      ----

        1. NOT Search Engine Friendly (can be fixed -- We Won't do)

        2. Initial Loading will take longer (can be fixed -- We will do)

Browser

-------

  \* HTML Renderer

  \* CSS Parser

  \* JS Engine

Single Page Apps (SPA)

----------------------

  7 Characteristics

  ------------------

    1. Pages should NOT reload

    2. URL should change

    3. History should be maintained w/o Page refresh

    4. Header and Footer should NOT be loaded again and again

    5. In b/w header and footer -- contextual updates should show up

    6. Active Menu should be set

    7. Page title should change

  SPA Frameworks / Libraries

  ---------------------------

    #1 React JS (library)

    #2 Angular 2+ (Framework)

    #3 VueJS (Framework)

    #4 Svelte (Framework)

Course Projects

------

  App #1 in React JS

  App #2 in Redux Toolkit

  App #3 in NodeJS (SSR)

  App #4 in NodeJS (RESTful API)

  App #5 in NodeJS (Video Conferencing App + Chat App)

  2 Books

=================================

2 Types of Languages by Data Typing

------------------------

  1. Strictly-Typed Languages

      Examples: Java, C#

      Syntax:

        String myName = "Arun";

        myName = "John"; // valid

myName = 100; // error

        int x = 10;

x = "Arun"; // error

  2. Loosely-Typed Languages

Examples: JavaScript, PHP, Python

Syntax in JS

--------

var myName = "Arun"; //valid

myName = "John"; // valid

myName = 10; // valid

myName = true; //valid

myName = undefined; //valid

myName = null; //valid

myName = function() {} //valid

Data Types in JS

----------

1. Primitive Data Types

string, number, boolean, undefined, null, symbol (ES2015), big int (ES2017)

2. Non-Primitive Data Types / Reference Types

Object, Array, RegEx

**Day 2:**

======

3 Layers to develop website / webapps

----

  1. Structural Layer

      HTML

  2. Presentation Layer

      CSS

  3. Interaction Layer

      JavaScript

HTML

-----

\* Standard

\* Maintained by W3C

CSS

----

\* Standard

\* Maintained by W3C

ECMAScript

-------

\* Standard

\* Grammar

\* Maintained by ECMA International

\* Implementations: JavaScript, JScript, ActionScript, Google AppsScript, QtScript

ES Before June 2015

-------

\* It was NOT Object Oriented Programming Language

\* no class, no extends, no import, no export

ES After June 2015 (ES2015)

-------

\* It has become Object Oriented Programming Language

\* class, extends, import, export

------------------

Todos

=====

1. Learn about JSON stringify and JSON parse

2. Learn about duplicating object

Ref: https://code.tutsplus.com/the-best-way-to-deep-copy-an-object-in-javascript--cms-39655a

3. Have an object in const and try changing its' properties

4. Have an array in const and try changing its' indices

5. Learn about using spread operator in an object to duplicate

6. Learn about promise

Refer: https://leanpub.com/understandinges6/read

Refer: The Best Way to Deep Copy an Object in JavaScript | Envato Tuts+ (tutsplus.com)

#####################################################

Content covered

notepad.pw / BoschMERNStackNov | The napkin of the internet.

JS Bin - Collaborative JavaScript Debugging

**Day 3:**

======

NPM Packages

--

  \* default packages

  \* third-party packages

  Package Management with NPM

  -----

    1. Search for packages

    2. Installing with command

    2. Updating with command

    3. Uninstalling with command

  3 Players

---------

1. Package Developers

2. npmjs Marketplace for Package

3. Web Developers

-------

NodeJS Ecosystem

---------

Dependencies

------------

1. Project Dependencies

\* some packages are needed for the functionality of the app to work

\* Examples: jquery, bootstrap, react, axios, dropzone, datatables.net

\* How to Install?

npm i jquery (or)

npm install jquery (or)

npm i jquery --save (or)

npm i jquery -S

Can be seen in package.json

"dependencies": {

"dropzone": "^6.0.0-beta.2",

"jquery": "^3.7.1"

},

2. Development Dependencies

\* some packages are needed for the development of the app

\* Examples: typescript, sass, less, webpack

\* How to install?

npm i webpack --save-dev  (or)

npm install webpack --save-dev  (or)

npm i webpack -D  (or)

Can be seen in package.json

"devDependencies": {

"webpack": "^5.89.0"

}

--------------------------------

Proxy related commands to try

---

npm config rm proxy

npm config rm https-proxy

#############################################################

https://felixgerschau.com/javascript-memory-management/

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow\_functions

https://jsbin.com/boxocuv/1/edit?js,console

https://jsbin.com/kidedur/edit?js,console

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/map

https://jsbin.com/hovefes/edit?js,console

https://jsbin.com/dufewik/edit?js,console

------------------------------------------

About to write to C:\Users\OLJ1COB\mern-stack-course-2023\npm-demo\package.json:

{

  "name": "npm-demo",

  "version": "1.0.0",

  "description": "\"my first npm app\"",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "author": "Leander Jones",

  "license": "ISC"

}

Is this OK? (yes) yes

npm notice

npm notice New minor version of npm available! 10.1.0 -> 10.2.4

npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.4

npm notice Run npm install -g npm@10.2.4 to update!

npm notice

----------------------------------------------------

**Day 4, 5, 6:**

=======

Webapps

Desktop Apps

Mobile Apps

Tablet Apps

TV Apps

Smartwatch Apps

Installable Webapps = Progressive Web App (PWA)

---

\* Installable in Mac or Windows or Linux

\* Installable in Android Mobiles and Tablets

\* Not Installable in iPhone and iPad

=====

Think in React

-------------

1. Identifying the number of components

2. Identifying the hierarchy of the components

3. ...

4. ...

Components

---------

\* Reusable block of code made with JS, JSX and CSS (optional)

1. Functional Components (RECOMMENDED)

\* made with JS functions

\* were called as dumb components / presentational components (before react v 16.8)

\* were called as stateless components (before react v 16.8)

\* no longer called as stateless/presentational/dumb components after hooks introduced

1.1 FC with Named Fn [DONE]

1.2 FC with Anonymous Fn [DONE]

1.3 FC with Arrow Fn [DONE] (RECOMMENDED)

2. Class Components (NOT RECOMMENDED -- We will NOT write. We will NOT LEARN)

\* made with JS class

\* were called as smart components / containers (before react v 16.8)

\* were called as stateful components (before react v 16.8)

\* no longer called as stateful/containers/smart components

JSX Limitations

-------

\* Will look like HTML. But really not.

\* All tags must be self-closed or separately-closed

\* class attribute in element should become className

\* JSX expressions must have one parent element.

Project Structure

---------------

src/

components or layouts

Header.js

Footer.js

MenuList.js

pages/

HomePage/

HomePage.js

PrimeVideoPage/

PrimeVideoPage.js

components/

LatestVideo.js

TrendingVideo.js

AboutUsPage/

components/ or views/

History.js

Team.js

Career.js

AboutUsPage.js

routes/

AppRoutes.js

Naming Conventions

------------------

Refer: https://www.freecodecamp.org/news/clean-coding-for-beginners/

Casing Types

--------

1. PascalCase

\* Recommended for Object Oriented Program's classes

\* also Recommended for Interface in TypeScript

\* Recommended for React Components and the folder of Components

examples

---

class CarDetails {

}

interface Employee {

}

const MenuList() => {

return (<p>some jsx</p>)

}

2. camelCase

\* Recommended for variables, let, const, functions, methods in JS & TS

\* Recommended for id's of html elements

\* Recommended for variables, const, fn, methods, event handlers of React Component

examples

-----

var myCar = 'BMW';

var isLoggedIn = true;

function getCarInfo() {

//....

}

function add() {

}

3. kebab-case

\* Recommended for URLs

\* Recommended for file names, folder names, assets in front end projects

\* Recommended for css classes unless you are using css modules in react

\* okay for id's of html elements if your architect says ok

examples

-----

btn-primary

slideshow-container

youtube-logo.png

4. snake\_case

\* never ever use this.

\* okay for const in JS & TS if architect accepts

examples

---

const MAX\_STUDENTS\_PER\_CLASS = 30;

----------------

Core Concepts of ReactJS

=======

Props [DONE]

States [DONE]

Events [DONE]

Conditional Rendering [DONE]

Lists & Keys [DONE]

Styling in React Components

1. External Styles [DONE]

2. Internal Styles [DONE]

3. Inline Styles [DONE]

Advanced Topics

---------

Routing [DONE]

Controlled Comp [DONE]

uncontrolled components [DONE]

Forms [DONE]

REST API Calls [DONE]

HOC [DONE]

Error boundary [DONE]

Fragment [DONE]

StrictMode [DONE]

Context API [DONE]

Unit Testing [DONE]

Build and Deployment [DONE]

Lazy Loading and Suspense [DONE]

Redux - Toolkit (2 hours)

**Day 7, 8:**

Single Page Apps (SPA)

----------------------

7 Characteristics

------------------

1. Pages should NOT reload

2. URL should change [DONE]

3. History should be maintained w/o Page refresh

4. Header and Footer should NOT be loaded again and again

5. In b/w header and footer -- contextual updates should show up [DONE]

6. Active Menu should be set

7. Page title should change

Limitations of ReactJS

-----

1. Doesn't have tools to help make SPA. Use react-router-dom

2. Doesn't have any tool to help connect to REST API. use axios or else fetch (native js)

3. Doesn't offer any form and form validation related tools. use either react-hook-form (recommended) or formik

4. For Most of the other things, you need to look for open source tools

Routing in React JS

-------------------

Step 0

------

Identify the Menus and also the URLs. Set them up in MenuList comp

Home http://localhost:3000/

Prime Video http://localhost:3000/prime-video

User Management http://localhost:3000/users

Products http://localhost:3000/products

Todos App http://localhost:3000/todos

Unit Testing http://localhost:3000/unit-testing-demo

About Us http://localhost:3000/about-us

Contact Us http://localhost:3000/contact-us

Step 1:

----

Check the app. URL is changed. Pages are still getting refreshed.

Let's work on creating necessary components for the above URLs

Create the following components for the Menus

Home HomePage Component

Prime Video PrimeVideoPage Component

User Management UserManagementPage Component

Products ProductsPage Component

Todos App TodosPage Component

Unit Testing UnitTestingDemoPage Component

About Us AboutUsPage Component

Contact Us ContactUsPage Component

Step 2:

-----

We need a third-party package to make it a SPA.

npm i react-router-dom

Let's seup react-router-dom in App.js

Move the whole app under <BrowserRouter></BrowserRouter>

Step 3:

-----

Let's Config the routes in between header and footer in App.js

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/prime-video" element={<PrimeVideoPage />} />

<Route path="/users" element={<UsersPage />} />

<Route path="/products" element={<ProductsPage />} />

<Route path="/todos" element={<TodosPage />} />

<Route path="/unit-testing-demo" element={<UnitTestingDemoPage />} />

<Route path="/about-us" element={<AboutUsPage />} />

<Route path="/contact-us" element={<ContactUsPage />} />

</Routes>

Step 4:

----

Check the app. Components will be loaded properly.

But, the pages will still refresh. Let's fix it.

Replace all <a href=''> tags with <Link to=''>

Step 5:

------

Now, check the app.

Pages Will NOT reload.

History is maintained w/o Page refresh

Header and Footer NOT loaded again and again

You have work on Setting the active menu and page title.

**Day 9, 10:**

Data Binding

------------

\* keeping the data in variable in JS and displaying it in JSX

Cross Component Communication

-------

1. Parent to Child Component Communication [DONE]

using props

2. Child to Parent Component Communication [TODO]

using props, passing callback fn as prop to the child comp

3. Any to Any Component Communication

3.1 using Context API of React JS [DONE]

3.2 using Redux-Toolkit [TO LEARN] / Redux / Zustand / MobX / Jotai / Recoil / Flux

======

Hooks

======

\* In react Hooks are functions

\* Hooks will have 'use' prefix

\* Rules of Hooks

\* Only Call Hooks at the Top Level

\* Only Call Hooks from React Functional components

Context API to share data across the app

------

Step 1: create the context for the feature by using createContext()

Step 2: export the context so that it can be used as a layer to share the data across many components in the app

// try this is App.js

Step 3: supply the data into the context through the provider

Step 4: get the data using useContext() hook in the necessary component

Step 5: if any comp wants to share the data through the context provider

==================

**Day 11:**

Errors in React JS

---

1. Compilation error

2. Runtime error

Higher Order Component (HOC)

----

\* it is a function that takes in SourceComponent and returns a New Component

\* it is a pattern

=====

SDLC

----

Analyse Requirements => Design => Dev => QA => Deployment => Maintain

Dev => QA => (bugs) => Dev => QA => (bugs) => Dev => QA => (bugs) => Dev => QA => (QA PASSED) => Deployment

Dev

----

Source Document: FRD, LLD, HLD, UX Designs

Test Cases:

1. add should work with only number -- it can receive params in the form of numbers

2. add must return number

3. add button should be disabled if other than numbers are entered

function add(a, b){

if(typeof a === 'number' && typeof b === 'number'){

return a + b;

}

}

add(10, 20); // 30

add(10, "30"); //

add("10", "30"); //

**Day 12: Cyclone (No KT)**

**Day 13:**

Modern Approach

====

Dev should write test cases => Dev should write the functionality code => (TOOLS) => (0 bugs) => Deployment

Automated Testing

-----

0. Static Type Checking (prop-types or typescript)

1. Unit Testing

\* testing the functional code by writing some code

\* testing tiny, independent unit of the component

\* testing the component in isolation (w/o connecting to rest api)

\* tools:

1. JEST (Unit testing framework and also a TEST RUNNER),

2. React Testing Library (RTL -- testing utility)

2. Integration Testing

3. End to End Testing

Testing Pattern:

----------

AAA (Arrange, Act, Assert)

Unit Testing Topics

------

Quick File walkthrough [DONE]

Testing JSX [DONE]

Testing Props [DONE]

Testing States and Events [DONE]

Testing Reducer and Context [TODO]

**Day 14:**

Build and Deployment

-----

Build Checklist

----

1. check console for any errors.

2. check console for any warnings.

3. remove all console.logs

4. look for run time errors. if anything occurs, fix the issue.

5. look for 'Compiled with warnings' errors in command prompt. if found, fix them too

How to take a build?

---

npm run build

**Day 15:**

Any to Any Comp Commn (2015-26)

State Management in React Apps

1. ReactJS + Flux [Not Maintained anymore]

\* too much complexity

2. ReactJS + Redux (2016 till today) ==> a.k.a classic redux [LEARN SEPARATELY]

\* complex

Middleware Integrations (choose anyone)

1. redux-thunk (38-40 steps)

2. redux-saga (46 steps)

Tools to be integrated

\* react-redux

\* @redux-devtools/extension

\* immer.js

3. ReactJS + ContextAPI (Hooks) (2019 till today) (Demo'd in prime-video-app)

\* Easier

4. ReactJS + Redux-Toolkit (Redux, React-Redux, Redux-thunk) => a.k.a modern redux

\* Easier

Middleware Integrations

1. redux-thunk (default one)

2. redux-saga

Tools, by default integrated

\* react-redux

\* @redux-devtools/extension

\* immer.js

======

How to create JS based react-toolkit project?

npx create-react-app my-app --template redux

How to create TS based react-toolkit project?

npx degit reduxjs/redux-templates/packages/vite-template-redux my-app

3 Important Concepts of Redux

----

Store

the whole state of the app.

Action

what happens in the app

Reducer

updates the store based on the action happened

3 Principles of Redux

-----

1. There should be single source of truth for the whole app. i.e Store

(it should be an object)

2. The only way to update the state is by dispatching an action

that tells what happened in the app

3. Reducer should update the store based on action occurred

and also based on previous and next state

createSlice({})

\* needs name

\* initial state object

\* also the reducers fns

Steps for Redux Toolkit Implementations

=====

Step 1:

---

Create Users comp in src/features/users/Users.js

import React from 'react'

const Users = () => {

return (

<div>Users</div>

)

}

export default Users;

------

Step 2:

---

Create slice for the feature

import React, { useEffect } from "react";

import { useSelector, useDispatch } from "react-redux";

import { addUserAsync, fetchUsersAsync } from "./usersSlice";

const Users = () => {

const usersInfo = useSelector((state) => { // getting state item from store

return state.users;

});

const dispatch = useDispatch(); // getting dispatch from store

useEffect(() => {

dispatch(fetchUsersAsync());

}, []);

console.log(usersInfo);

const handleSubmit = (event) => {

event.preventDefault();

dispatch(addUserAsync({

name: 'a',

email: 'a@b.com',

phone: 12345678

}));

}

if (usersInfo.isLoading) {

return <div className="spinner-border text-primary"></div>;

}

if (usersInfo.isError) {

return <div>Some error occurred! Try again Later</div>;

}

return (

<div className="row">

<h1>Users</h1>

<div className="col-md-4">

<h2>Add User</h2>

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Enter Name"

className="form-control"

/>

<br />

<input

type="text"

placeholder="Enter Phone"

className="form-control"

/>

<br />

<input

type="text"

placeholder="Enter E-Mail"

className="form-control"

/>

<br />

<button type="submit" className="btn btn-primary">Create User</button>

</form>

</div>

<div className="col-md-8">

<div className="row">

{usersInfo.userList?.map((user) => {

return (

<div className="col-md-4" key={user.id}>

<div className="card">

<div className="card-body">

<h5 className="card-title">{user.name}</h5>

<h6 className="card-subtitle mb-2 text-muted">

E-Mail: {user.email}

</h6>

<p className="card-text">Phone: {user.phone}</p>

</div>

</div>

</div>

);

})}

</div>

</div>

</div>

);

};

export default Users;

Step 3:

-------

in src/app/store.js

import { configureStore } from '@reduxjs/toolkit';

import counterReducer from '../features/counter/counterSlice';

import usersReducer from "../features/users/usersSlice";

export const store = configureStore({

reducer: {

counter: counterReducer,

users: usersReducer,

},

});

Now, Check the app!

---

**Day 16:**

NodeJS

-----

\* Runtime

\* JS => (NodeJs Runtime) => Machine Language

Back End App

------

~10 Examples

App #1: SSR App using NodeJS (40 mins)

App #2: REST API using ExpressJS

\* SSR App using ExpressJS

\* CRUD App (User Management)

\* Add User (POST)

\* List Users (GET)

\* User Details (GET)

\* Edit User (PUT/PATCH)

\* Delete User (DELETE)

\* DB Connectivity (mongo[JS] and Mysql [SQL])

\* Authentication

\* Encrypt the password

\* Never store the encrypted password

\* Storing only salt and hash

JS

---

\* released in 1995

\* scripting language

\* JS Engine would execute (JS engine was part of the browser)

Browser Core

-----

HTML Renderer + CSS Parser + JS Engine

NodeJS History

----

\* Netscape Navigator by Netscape in 1994 (NOT Open Sourced) -- Stopped around ~2007-08

\* Opera by Opera in 1995 (NOT Open Sourced)

\* Internet Explore by MS in 1995 (NOT Open Sourced)

\* Safari by Apple in 2003 (Webkit Core - Open Sourced)

\* Firefox Browser by Mozilla in 2004 (Mozilla Core - Open Sourced)

\* Chrome Browser by Google (Open Sourced Chromium Core = Webkit + Mozilla + V8 JS Engine) in 2008

\* V8 JS Engine was released as OSS

\* V8 JS Engine would convert JS into machine language and execute the code

\* NodeJs was developed using V8 Engine and released as installable runtime in many platforms in 2009

\* Opera chose Chromium Core

\* Edge chose Chromium Core

\* Brave chose Chromium Core

\* Vivaldi chose Chromium Core

Types of Apps

------

Front End Apps, Back End Apps, Mobile / Tablet Apps, Desktop Apps, TV Apps, Smartwatch Apps

Front End Apps

-----

HTML, CSS, JavaScript

Back End Apps

-----

Java, DotNet, PHP, Perl, Python, Ruby, JavaScript (NodeJS / ExpressJS)

Mobile / Tablet Apps

------

Java (Kotlin), Swift, DotNet, JavaScript (Ionic or React Native)

Desktop Apps

-----

C, C++, Java, DotNet, Python, JavaScript (Electron JS)

TV Apps

----

Java (Kotlin), Swift, DotNet, JavaScript (React or React Native)

Database

-----

SQL, JavaScript (MongoDB, and many RDBMS)

If NodeJs has to become Server Side language, it should deal with the following

-----

1. It should have Access to File System

2. Receive Http Req and Process Http Responses

3. Connect to Databases

4. Be faster

**Day 17:**

Digital Communication

-----------

Simplex

Transmitter => Receiver

Example: AM Radio, FM Radio

Half-Duplex

Transreceiver ==> Transreceiver

Transreceiver <== Transreceiver

Example: Pager, Walkie Talkie, Http

Duplex / Full-Duplex

Transmitter <==> Receiver

Http Protocol

----

Request ==> Server (10:58:00 AM)

Response <== Server (10:58:01 AM)

Request ==> Server (10:58:03 AM)

Response <== Server (10:58:05 AM)

\* Stateless Protocol

App #1

-----

REST API

App #2

----

Video Chat app

Frameworks in NodeJS

-------

\* ExpressJS

**Day 18:**

User Management App - CRUD App

---

1. Create User

http://localhost:3001/api/users/

POST Method

2. Listing Users

http://localhost:3001/api/users/

GET Method

3. User Details

http://localhost:3001/api/users/2

GET Method

4. Update User

http://localhost:3001/api/users/2

PUT/PATCH Method

5. Delete User

http://localhost:3001/api/users/2

DELETE Method

MVC

---

Model (Data)

View (Template / anything that is supplied to end users as output)

Controller

Database

----

\* place to store data

\* many types of databases

1. Relational Database Management System (RDBMS)

Examples: Oracle, MySQL, MariaDB, MS SQL, PostgreSQL

\* data would be stored in tables

\* tables will have fixed columns and rows

\* suitable for structured data

\* schema based

\* interact with db using Structured Query Language (SQL)

2. NoSQL Databases (Not Only SQL)

Examples: MongoDB, etc

\* data would be stored in collections

\* collections will have documents made up of fields/keys and values

\* suitable for unstructured data

\* schema-less

\* interact with MongoDB using JavaScript

\* databases will have 2 sides

1. Admin side (creating db, backing up, restoring) [NOT TO LEARN]

2. Database Client side (creating data, managing data) [TO LEARN]

\* databases will have server and client

\* database server should be running in some port number

\* databases will be protected with db credentials (username and password)

\* to access the db, you need DB clients

1. Shell / Command Line Client [TO LEARN]

2. GUI client

(MS SQL Server, MySQL Workbench, PHPMyAdmin, SQLYog, HeidiSQL, Compass [TO LEARN], Robo 3T [TO LEARN])

3. Programming Client [TO LEARN]

For MongoDB: MongoDB driver (official), Mongoose (We will use this)

If you want to have partially structured data (i.e with schema)

use Mongoose

----

**Day 19**

How to start the MongoDB Server?

-----

in windows command prompt go into this path

cd C:/Program Files/MongoDB/Server/7.x/bin

Start the server

----

mongod

Start the shell client

-----

mongo (may not work in mongodb v6+)

========

MongoDB commands

------

to list all dbs (only the dbs with collections will appear)

show dbs

to know the currently selected db

db

to create a new db or to use existing db (it will switch to the db as well)

use usersDb

check the currently selected db

db

to create a collection

db.createCollection('users')

to see all collections of the selected db

show collections

CRUD Operations

-----

CREATE: Let's insert some data into users collection

db.users.insert({name: "John"})

db.users.insert({phone: 536475675, city: 'London'})

db.users.insert({age: 20})

db.users.insert({name: 'Steve', favCities: ["Toronto", "Amsterdam", "Kuala Lumpur"]})

READ: Let's read the data from db collections

To find all documents

db.users.find()

to fetch one document

db.users.find({age: 20}) // there may be many docs with age 20

db.users.find({name: 'John'}) // there may be many docs with name John

db.users.findOne({"\_id" : ObjectId("657aa97344f78d81bda5bd77")})

UPDATE: Let's update the data

db.users.updateOne( { name: "John" },

{

$set: {

name: 'Steve',

story: "A teenage girl risks everything–including her life–when she falls in love with a vampire."

}

})

DELETE: You explore it here

Refer: https://www.mongodb.com/docs/mongodb-shell/crud/delete/

====

**Day 20:**

Some Example API Endpoints

----

localhost:3001/api/users -- GET, POST

localhost:3001/api/users/1 -- GET, UPDATE, DELETE

localhost:3001/api/users?active=true&limit=10&page=1 -- GET (1-10)

localhost:3001/api/users?active=true&limit=10&page=2 -- GET (11-20)

localhost:3001/api/employees -- GET

localhost:3001/api/employees/1 -- GET, UPDATE, DELETE

localhost:3001/api/auth/signup - POST

localhost:3001/api/auth/login - POST

localhost:3001/api/auth/reset-pw - POST [NOT TO LEARN]

localhost:3001/api/auth/verify-email?key=3245678o34w3erwrwrt239756345734056345634659347568345634 - GET [NOT TO LEARN]

localhost:3001/api/products/ - LIST all products

localhost:3001/api/products/furnitures - LIST all furnitures

localhost:3001/api/products/furnitures/sofas - LIST all sofas

localhost:3001/api/products/books - LIST all books

localhost:3001/api/products/books/fiction - LIST all fictions

localhost:3001/api/products?category=books&subcategory=fiction&sort=price-l-to-h - LIST all fictions

------

Refer for Auth:

https://github.com/arunprabu/nodejs-expjs-sep2022/blob/master/user-management-rest-api/utils/auth.util.js

https://github.com/arunprabu/nodejs-expjs-sep2022/blob/master/user-management-rest-api/routes/api/users.js

**Day 21:**

Build and Deployment

-------------

If it is a JAva Project -- you need to take a build

Java => javac => (bytecode) => JRE => (Machine Language)

But, in NodeJS -- we don't need to take a build -- simply deploy

JavaScript => NodeJsRE => (Machine Language)

Deployment Steps

-----

1. copy the entire project project repo without node\_modules folder

2. paste it in a machine in which you want to deploy this app

3. Install NodeJS (appropriate version) in the deployable machine

4. start the db server

5. go to the command prompt and locate the project

6. npm i

7. npm start

------------

**Day 22:**

NPM Packages

--

\* default packages

\* third-party packages

Package Management with NPM

-----

1. Search for packages

2. Installing with command

2. Updating with command

3. Uninstalling with command

3 Players

---------

1. Package Developers

2. npmjs Marketplace for Package

3. Web Developers

Any to Any Comp Commn (2015-26)

State Management in React Apps

1. ReactJS + Flux [Not Maintained anymore]

\* too much complexity

2. ReactJS + Redux (2016 till today) ==> a.k.a classic redux [LEARN SEPARATELY]

\* complex

Middleware Integrations (choose anyone)

1. redux-thunk (38-40 steps)

2. redux-saga (46 steps)

Tools to be integrated

\* react-redux

\* @redux-devtools/extension

\* immer.js

3. ReactJS + ContextAPI (Hooks) (2019 till today) (Demo'd in prime-video-app)

\* Easier

4. ReactJS + Redux-Toolkit (Redux, React-Redux, Redux-thunk) => a.k.a modern redux

\* Easier

Middleware Integrations

1. redux-thunk (default one)

2. redux-saga

Tools, by default integrated

\* react-redux

\* @redux-devtools/extension

\* immer.js

======

How to create JS based react-toolkit project?

npx create-react-app my-app --template redux

How to create TS based react-toolkit project?

npx degit reduxjs/redux-templates/packages/vite-template-redux my-app

3 Important Concepts of Redux

----

Store

the whole state of the app.

Action

what happens in the app

Reducer

updates the store based on the action happened

3 Principles of Redux

-----

1. There should be single source of truth for the whole app. i.e Store

(it should be an object)

2. The only way to update the state is by dispatching an action

that tells what happened in the app

3. Reducer should update the store based on action occurred

and also based on previous and next state

createSlice({})

\* needs name

\* initial state object

\* also the reducers fns

Steps for Redux Toolkit Implementations

=====

Step 1:

---

Create Users comp in src/features/users/Users.js

import React from 'react'

const Users = () => {

return (

<div>Users</div>

)

}

export default Users;

------

Step 2:

---

Create slice for the feature

import React, { useEffect } from "react";

import { useSelector, useDispatch } from "react-redux";

import { addUserAsync, fetchUsersAsync } from "./usersSlice";

const Users = () => {

const usersInfo = useSelector((state) => { // getting state item from store

return state.users;

});

const dispatch = useDispatch(); // getting dispatch from store

useEffect(() => {

dispatch(fetchUsersAsync());

}, []);

console.log(usersInfo);

const handleSubmit = (event) => {

event.preventDefault();

dispatch(addUserAsync({

name: 'a',

email: 'a@b.com',

phone: 12345678

}));

}

if (usersInfo.isLoading) {

return <div className="spinner-border text-primary"></div>;

}

if (usersInfo.isError) {

return <div>Some error occurred! Try again Later</div>;

}

return (

<div className="row">

<h1>Users</h1>

<div className="col-md-4">

<h2>Add User</h2>

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Enter Name"

className="form-control"

/>

<br />

<input

type="text"

placeholder="Enter Phone"

className="form-control"

/>

<br />

<input

type="text"

placeholder="Enter E-Mail"

className="form-control"

/>

<br />

<button type="submit" className="btn btn-primary">Create User</button>

</form>

</div>

<div className="col-md-8">

<div className="row">

{usersInfo.userList?.map((user) => {

return (

<div className="col-md-4" key={user.id}>

<div className="card">

<div className="card-body">

<h5 className="card-title">{user.name}</h5>

<h6 className="card-subtitle mb-2 text-muted">

E-Mail: {user.email}

</h6>

<p className="card-text">Phone: {user.phone}</p>

</div>

</div>

</div>

);

})}

</div>

</div>

</div>

);

};

export default Users;

Step 3:

-------

in src/app/store.js

import { configureStore } from '@reduxjs/toolkit';

import counterReducer from '../features/counter/counterSlice';

import usersReducer from "../features/users/usersSlice";

export const store = configureStore({

reducer: {

counter: counterReducer,

users: usersReducer,

},

});

Now, Check the app!

---

-------

NodeJS Ecosystem

---------

Dependencies

------------

1. Project Dependencies

\* some packages are needed for the functionality of the app to work

\* Examples: jquery, bootstrap, react, axios, dropzone, datatables.net

\* How to Install?

npm i jquery (or)

npm install jquery (or)

npm i jquery --save (or)

npm i jquery -S

Can be seen in package.json

"dependencies": {

"dropzone": "^6.0.0-beta.2",

"jquery": "^3.7.1"

},

2. Development Dependencies

\* some packages are needed for the development of the app

\* Examples: typescript, sass, less, webpack, mocha, chai

\* How to install?

npm i webpack --save-dev (or)

npm install webpack --save-dev (or)

npm i webpack -D (or)

Can be seen in package.json

"devDependencies": {

"webpack": "^5.89.0"

}

--------------------------------

Proxy related commands to try

---

npm config rm proxy

npm config rm https-proxy

==================

**Day 23:**