GET/POST/PUT/DELETE DB

Create folder mongo\_apis

Terminal : Browse expressjs.com > getting started tab above > Express Generator <http://expressjs.com/en/starter/generator.html>, we don’t want views folder to be created

Cmd : npx express-generator --no-view

In app.js file :

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

this is used to serve static files in express > browse express static <https://expressjs.com/en/starter/static-files.html>

To serve static files such as images, CSS files, and JavaScript files, use the express.static built-in middleware function in Express.

Nodejs server not only sends JSON response but also HTML, CSS pages too

Graphical user interface, text, application, email

Description automatically generated

Html, css files are present in public folder

Graphical user interface, text

Description automatically generated

Mongodb Atlas

Network access : 0.0.0.0/0 : allows all IP addresses

Database access : dinga

Atlas : Cluster : connect : connect your application : cluster string link

Terminal : npm install - -save mongoose, npm install - -save nodemon, change start script in pkg.json to start from nodemon

Mongoose is used to connect mongodb & express

In app.js, replace var with const, import mongoose

We are adding mongoose.connect syntax in app.js & not in the routes bcoz the connection to mongodb should be made only once, if in routes multiple times connection happens

Browse collection > create new collection

npm start cmd at last

Body parser : taken from the client as a JSON object

Cookie parser : when cookie is sent/stored to server > it’s sent in string format or irregular format, cookieParser makes sure that the data is sent in proper format. Like JSON.parse

Write Students route for post req, send data in postman body, Body > raw, text to JSON

If datatype is a collection of string, studentDetails : Object

"name" : String,

Text

Description automatically generated

{

    "name" : "Asha",

    "branch" : "ECE",

    "rollNo" : 10

}

Syntax for CRUD : <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/>

POST : Make changes in postman for new entry to db : <http://localhost:8080/students/createData>

Graphical user interface, text, application, email

Description automatically generated

Get : router : <http://localhost:8080/students/getDetails>

Browse Remove id from mongoose results

URL is http, not https : if we develop in local, it’s http. If it’s https then special certificate called SSL certificate is needed

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Steps :

1. Setup an ExpressJS Server
2. Install mongoose, nodemon
3. Change pkg.json start script to nodemon
4. Mongoose.connect : mongodb atlas > set network access & database access, Cluster – database – Atlas : connect > connect your application > string in app.js (Change password & db name) (json entries from mongoose npm)
5. Define Routes (Schema instance is created & saved)
6. Create database in mongodb atlas, then post req from postman, data comes in req.body in code
7. Create new Schema : blueprint of json, set collection & export & save the schema in routes (save())
8. Once the data is saved successfully, refresh collection in atlas
9. Get route (find()), particular data also can be found
10. Conclusion : we created POST req, made an instance of Schema, saved data to database & from GET route we got data

PUT Request : <http://localhost:8080/students/updateDetails> : for rollNo (get from atlas), set branch

Body > raw > JSON

hen Graphical user interface, text, application, email

Description automatically generated

Make changes in the above record to update other records, then verify in GET req

Update records from postman or frontend client (React) or db

DELETE Route : <http://localhost:8095/students/deleteStudentDetail>

Graphical user interface, text, application, email

Description automatically generated

to delete rollNo (id) : don’t do, bcoz Schema gets disturbed

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Mongoose.connect String, password, db name, cluter url are confidential & should not be leaked: never upload to GitHub

So in order to make secure & use in code : environment variables : we ignore few things (like .gitignore)

environment variables are the variables defined in the Server environment & Local files : app.js should be uploaded but mongoose.connnect (URL) should not be uploaded

dotenv npm module in nodejs : browse dotenv npm : <https://www.npmjs.com/package/dotenv>

cmd : npm install dotenv –save //--save to save dependencies in pkg.json

dotenv dependency is reflected in pkg.json, add require('dotenv').config()in app.js 2nd line

create .env file & store constant : cut the string & paste in .env file

process.env.DB\_URL\_STRING

Graphical user interface, text, application, email

Description automatically generated

now .gitignore file to store .env file so that .env file won’t go to GitHub : gitignore.io website : <https://www.toptal.com/developers/gitignore> create gitignore file & add in vs code

npm modules should be ignored so node below

Graphical user interface, application, website

Description automatically generated

CORS : install cors bcoz whenever we make request from localhost, it doesn’t directly go to Heroku server. Browse cors npm : <https://www.npmjs.com/package/cors>

npm install cors --save

add in app.js Install & import/require

Graphical user interface, text, application, email

Description automatically generated

Heroku Deployment Steps

1. Browse deploy nodejs app to Heroku <https://devcenter.heroku.com/articles/deploying-nodejs> for deployment steps
2. Run cmd to know node version : node –version, add engines dependency to pkg.json & edit the node version, change the script to node from nodemon bcoz we NO longer need to restart the server again & again once it’s deployed to Heroku
3. Cmds (**Npm is a tool that use to install packages.** **Npx is a tool that use to execute packages**. )

npx heroku local web

1. Check the .gitignore file from doc & code whether are they present
2. Cmd :
   1. git init
   2. git add . : add manually + symbol
   3. add files manually : click on source control (3rd icon) & click + & add

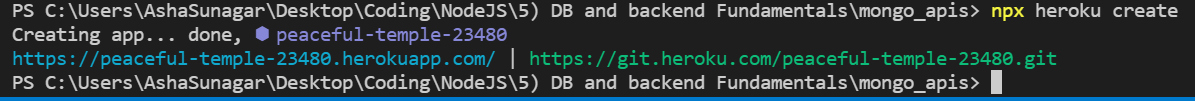
Graphical user interface, text, application, chat or text message

Description automatically generated



3 horizontal dots > commit > commit staged

* 1. Git commit is done
  2. Login to Heroku : npx heroku login : press any key & browser opens automatically, click on login in browser which is reflected in VS code terminal
  3. Create Heroku app : npx heroku create, now see Heroku browser dashboard



* 1. Cmd to know origin is created : git remote -v
  2. Cmd git branch : it’s master
  3. Cmd git push heroku master > DONE
  4. In Browser Heroku, click open app : some error : cmd npx heroku logs –tail

Error bcoz of d string in app.js but .env file is not pushed (DB\_URL\_String is undefined) so set that variable : browse setting environment variable Heroku : store .env file in server <https://devcenter.heroku.com/articles/config-vars>

Cmd npx heroku config (no use)

So set envt var manually in dashboard settings > Click on Reveal Config vars

This variable should not be on Github, but should be on Server Side

Paste the string without “ “

Graphical user interface, application, Teams

Description automatically generated

Click on open app : base url <https://peaceful-temple-23480.herokuapp.com/> edit postman local urls & check <https://peaceful-temple-23480.herokuapp.com/students/getDetails>

React App Deployment : Netlify

Express/Node/Backend APIs : Heroku

Now create React App : <https://reactjs.org/docs/create-a-new-react-app.html>

npx create-react-app my-app

cd my-app

npm start

in react app, remove default code from App.js

install axios module : npm install axios –save //--save to reflect in pkg.json, take GET API URL (Heroku URL)

To deploy react app in netlify : cmd npm run build : build file gets created, from file manager drag & drop that build folder to netlify drop, change the name using domain settings

Images in mongoDB atlas

Graphical user interface, application

Description automatically generated