**MERN BOOTCAMP**

**Mango DB** - <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

Run MangoDB - C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe

"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db"

**Express JS** - <https://expressjs.com/>

1) npm init => start server.

2) npm i express => install Express.

Before any project do this

// add all used packages as mentioned below.

const express = require("express");

const app = express();

const port = 3000

// create routes

app.get('/', (req, res) => {

  res.send('Hello World!')

})

app.listen(port, () => {

  console.log(`Example app listening on port ${port}`)

})

**Nodemon**

When we change code, we need to restart server, to prevent this we use nodemon (automatic reload)

npm install nodemon => install nodemon

update json file ->

"scripts": {

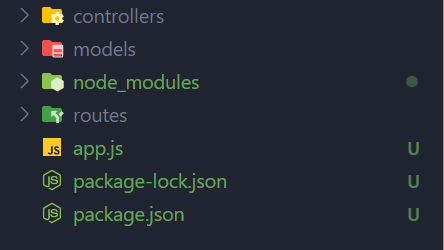
    "start": "nodemon index.js"

  },

To start server use -> npm start

**Backend Development**

Create projbackend folder and create folders as mentioned below



**Mongoose** – used to deal with mangodb and object modelling.

**Documentation** - <https://mongoosejs.com/docs/guide.html>

**Salt cryptography** - <https://en.wikipedia.org/wiki/Salt_(cryptography)>

It is used to store password in encrypted format.

Steps for backend –

1. Add required libraries.
2. Create Models (Mangoose Schema).
3. Export module as

module.exports = mongoose.model("User", userSchema);

[Virtuals](https://mongoosejs.com/docs/api.html#schema_Schema-virtual) are document properties that you can get and set but that do not get persisted to MongoDB. The getters are useful for formatting or combining fields, while setters are useful for de-composing a single value into multiple values for storage.

**UUID** – For the creation of [**RFC4122**](http://www.ietf.org/rfc/rfc4122.txt) UUIDs

<https://www.npmjs.com/package/uuid>

**Crypto**- used to encrypt password.

<https://www.npmjs.com/package/crypto-js>

**Connect App to mango dB using mangoose -** <https://mongoosejs.com/>

1. **Run mango DB**
2. **Start express server**

**Set Env variables for deployment of project**

dotenv - <https://www.npmjs.com/package/dotenv>

* create .env file in project folder and write variable name and assign value to it.
* To access variable use process.env.VARIABLE

**Middleware** - required for all mern apps.

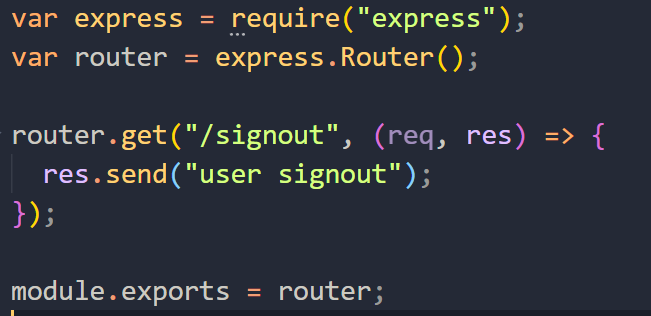
* Acts as intermediate, used for checking few things before routing to next page.
* <https://expressjs.com/en/guide/using-middleware.html>
* <https://expressjs.com/en/guide/writing-middleware.html>

**Most Common Middleware used –**

* **Body-parser** - <https://www.npmjs.com/package/body-parser> - Parse incoming request bodies in a middleware before your handlers, available under the req.body property.
  + **To use this install this, add to package.**json then use - app.use(bodyParser.json());
* **Cookie-parser** - <https://www.npmjs.com/package/cookie-parser> - Parse Cookie header and populate req.cookies with an object keyed by the cookie names. Optionally you may enable signed cookie support by passing a secret string, which assigns req.secret so it may be used by other middleware.
  + To use this install this, add to package.json then use - app.use(cookieParser());
* **cors** - <https://www.npmjs.com/package/cors> -  mechanism that allows restricted [resources](https://en.wikipedia.org/wiki/Web_resource) on a [web page](https://en.wikipedia.org/wiki/Web_page) to be requested from another [domain](https://en.wikipedia.org/wiki/Domain_name) outside the domain from which the first resource was served.
  + To use this install this, add to package.json then use - app.use(cors());

**Routing in express – Routing** refers to how an application’s endpoints (URIs) respond to client requests.

* <https://expressjs.com/en/guide/routing.html>
* User express.Router for easy routing as mentioned in documentation.
* Eg –



**Controller –** Controllers are typically call back functions that corresponds to the routers to handle requests.  All functions of routes are written in controller file. Keep file name of route and controller same.

* [**https://dev.to/ericchapman/nodejs-express-part-5-routes-and-controllers-55d3**](https://dev.to/ericchapman/nodejs-express-part-5-routes-and-controllers-55d3)
* [**https://lo-victoria.com/build-a-rest-api-with-nodejs-routes-and-controllers**](https://lo-victoria.com/build-a-rest-api-with-nodejs-routes-and-controllers)

**Robo3T –** Used to see what is happening in database in UI

* **Open app => create new connection => save**
* **Click connect.**

**Postman – used to send data or request data for website**

**Express validator -** <https://express-validator.github.io/docs/>

Used to validate data before saving data to database.

**­­­­**

**Authentication –**

1. **Database – check whether password matches in database**
2. **Token based approach – generate token and save in cookie, if token is present -> user is login.**

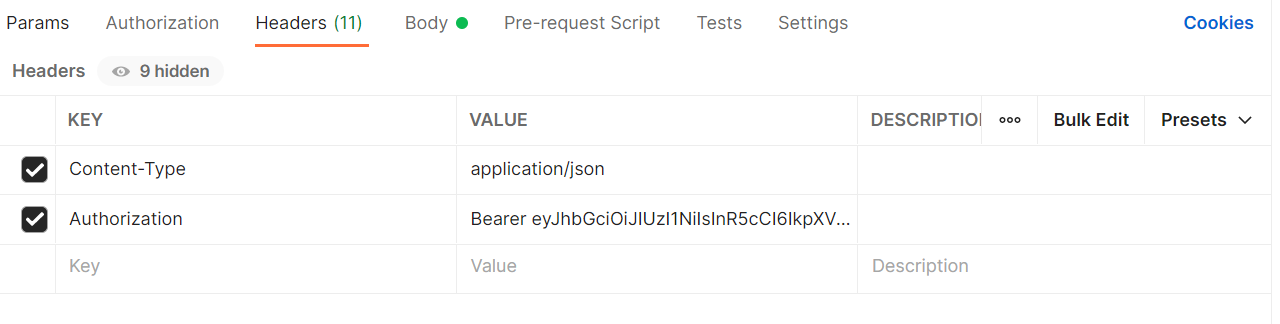
[https://www.npmjs.com/package/jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken%20)

<https://www.npmjs.com/package/express-jwt>

**Bearer token –** somebody who is carrying token

<https://jwt.io/introduction>

we cannot send token directly, so use

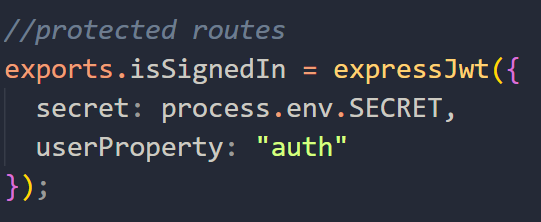


To **signout** user just clear cookies.



**How to protect routes?**

Use jwt to protect - <https://www.npmjs.com/package/express-jwt>



**Populate**

<https://mongoosejs.com/docs/populate.html>

Parameters - [https://expressjs.com/en/5x/api.html#router.param](https://expressjs.com/en/5x/api.html%23router.param)

Used to get parameters from url