Node + express + mongo rest api.

1. npm init - for generating package.json file. hit enter until it done

2. npm install express - install express

3. npm install nodemon - install nodemon
   every time refresh code refresh server

4. "start" : "nodemon app.js" - execute and run our server
   in a script remove test and add this

5. create app.js file

6. const express = require('express');
   add express to app

7. const app = express();
   execute express

(8) app.Listen(3000); - Listen to Port

(9) app.get('/', (req, res) => {
    res.send('we are on Posts');
});

↑ routs (get, Post, Jelete, Patch) update

(10) middlewares (Functions that execute with its route hit)

app.use('/', () => {
    console.log('---');
})

this run with call of '/' route

(11) npm install mongoose - install mongo

(12) npm install dotenv - to hide credential

(13) const mongoose = require('mongoose')

```
mongoose.connect('url', () =>
    : console.log ('connected to DB!')
);
```

Connect db

(14) use .env file to hide important data. anyone cannot see it

in .env

DB_Connection = URL

in app.js

```
require('dotenv/config');

mongoose.connect (Process.env.DB_
                  Connection,
    () =>
        console.log('connected to DB!')
```

(15) Add routes to different file

1. make routers folder
2. make Posts.js file

```
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  res.send('we are on Posts');

module.exports = router;
```

import

```
const PostRoute = require('./routes/Posts');
app.use('/Posts', PostRoute);
```

(16) make models.

1. make models folder
2. make Post.js file

~~Sch~~ const mongoose = require("mongoose");

```
const PostSchema = mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    description: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});
```

posts.js

```
router.Post('/', (req, res) => {
    Console.log(req.body);
})
```

call this using Postman
after calling we can see our data on consule

```
{
    "title": "----",
    "description": "----------"
}
```

(17) body Parser install - for req.body change to json object

npm install body-parser

app.js

```
Const bodyParser = require("body-Parser")

app.use(bodyParser.json());
```

every time when we call request this middleware called and chang data to json

(18) Add data to DB

in end point in Posts.js need to change like this

```
router.Post('/', (req, res) => {
    const Post = new Post({
        title : req.body.title,
        description : req.body.description
    });
    Post.save()
    .then(data => {
        res.json(data);
    })
    .catch(err => {
        res.json({message: err});
    });
})
```

reformat

```
router.Post('/', async (req, res) => {
    const Post = new Post({
        title : res.body.title,
        description: res.body.description
    });
    try {
        const savedPost = await Post.save();
        res.json(savedPost);
    } catch(err){
        res.json({ message : err});
    }
```

(19) Get data from DB

Posts.js

```
router.get('/', async(req, res) => {
    try{
        const Posts = await Post.find();
        res.json(Posts);
    }catch(err){
        res.json({message: err});
    }
});
```

get specific one

change

```
const Posts = await Post.findById(req.
                    params.PostId);
```

(20) delete data from DB

```
router.delete('/:postId', async(req, res) => {
  try{
    const removePost = await Post.remove({
    res.json(removePost);  _id: req.params.postId});
  } catch(err){
    res.json({message: err});
  }
});
```

(21) update data on DB

```
router.patch('/:postId', async(req, res) => {
try {
  const updatePost = await Post.updateOne({
              _id: req.params.postId},
              {$set: {title: req.body.title}});
  res.json(updatePost);
 } catch(err){
    res.json({message: err});
  }
});
```

(22) add cors to app

npm install cors

app.js

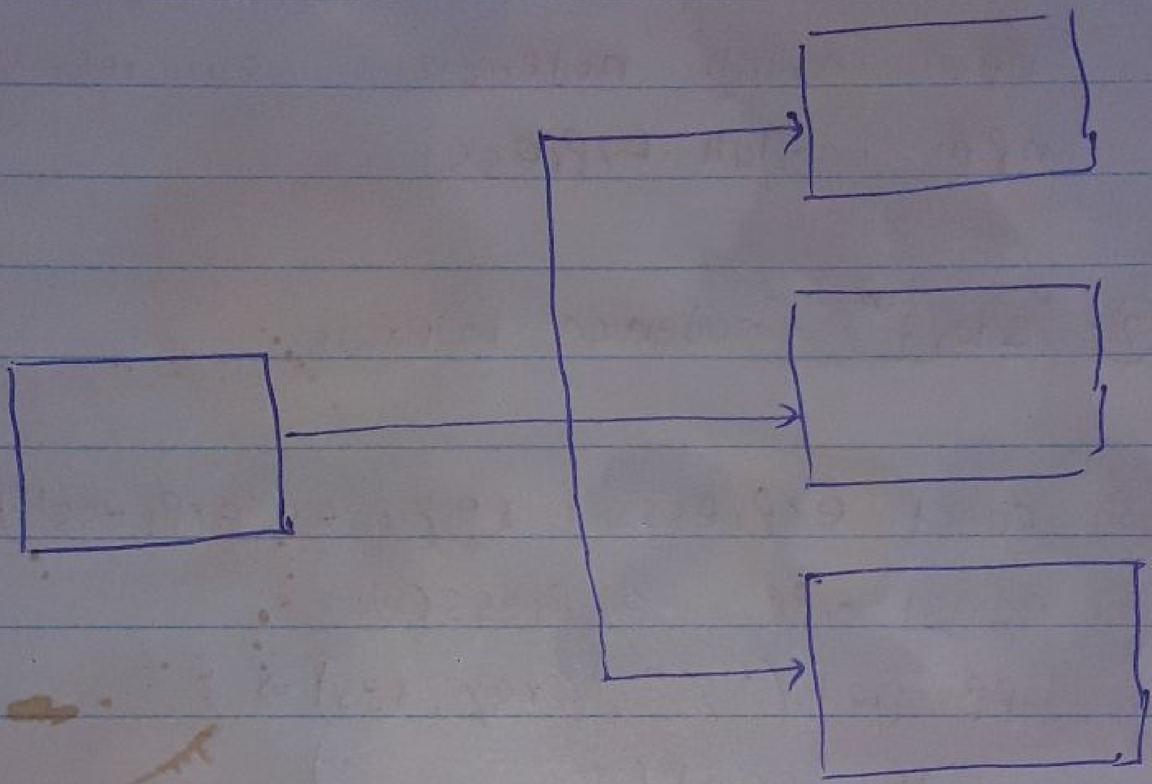
```
const cors = require('cors');

app.use(cors());
```

(23) Call from frontend

```
fetch('http://localhost:3000/Posts')
.then(result => {
    console.log(result);
})
```

JWT (json web token)

Authentication | Authorization



Backend | Clients

userගේ data හරියට check කරනවා මේ. ඒ user ද කියල

user එවන request checkකරනවා මේ ඒ userමද මේ request එවන්නෙ කියලා (using token)

once we send user in login Process

① npm init

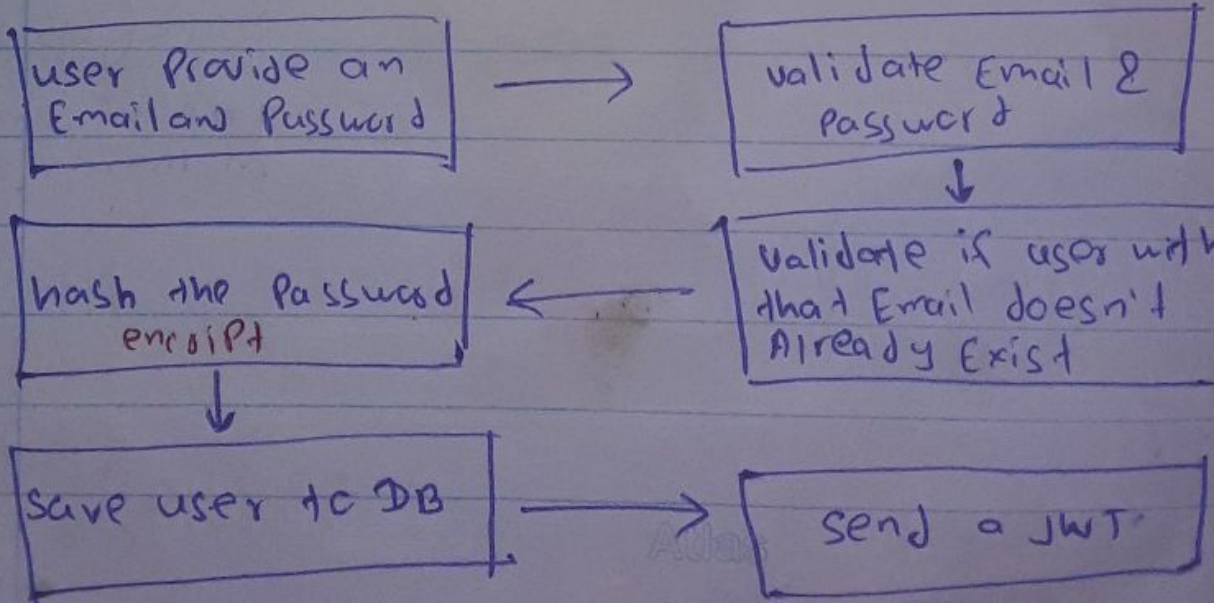
npm install nodemon auto reload nodemon

npm install express

② "start" - "nodemon index.js"

③
```
const express = require("express");
const app = express()
app.get("/", (req, res) => {
    res.send("- - -")
})
app.Listen(5000, () => {
    console.log("now this running on 5000")
})
```

Sign up route



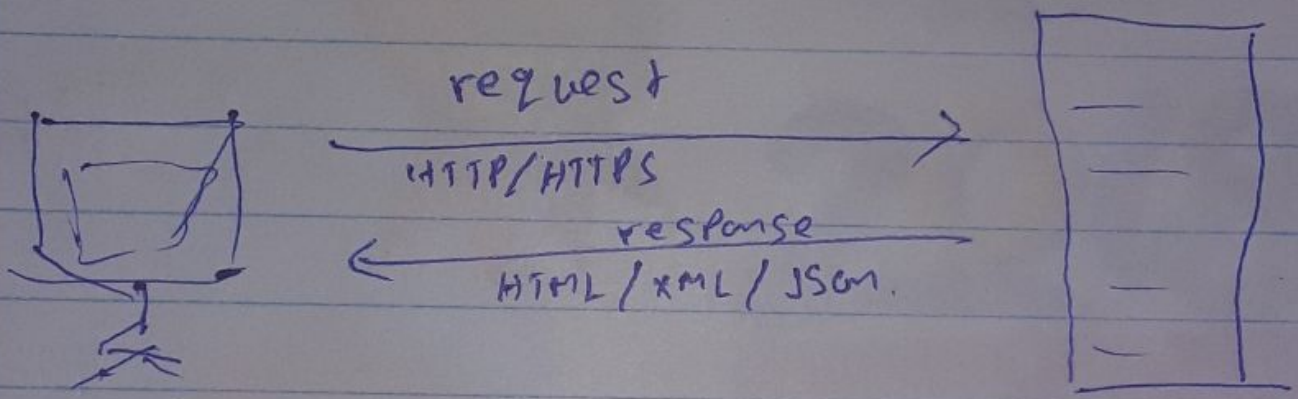
④ make routes folder
make auth.js file
add it to index.js file

```
const auth = require("./routes/auth")
const bodyParser = require("body-parser")
app.use(bodyParser.json());
app.use("/auth", auth)
```

⑤ express validater use for validation
email and password.
npm install express-validator

```
const {
```

client make request to server using HTTP/HTTPS ~~language~~ methods and server make respond to server using ~~a~~ language ~~server~~ ca client can understand



1st step for backend

Test request response cycle is working or not

find Simple ~~req~~ index for request