**Cafe Management System:**

**Backend Part (NodeJS):**

**Start:**

**Backend Structure:**

1: Create Package.json file:

Run command **npm init** for create package.json file.

2: Install **Express, MYSQL, dotnev, cors .**

**Command:** npm install –save express mysql dotenv cors.

3: Install **Nodemon:**

**Command:** npm install –save dev nodemon.

4: Create New File at root folder **server.js**.

And add start in **scripts object** like:

**“start”: ”nodemon server.js”**

5: Create **.env** file at root folder.

Add port details in this file like:

**PORT = 8080**

And import this file in **server.js** file.

**Server.js File:**

1: Import **.env** file.

require(‘dotenv’).config();

2: import **http**:

const http = require(‘http’);

3: Create new file at root folder name **index.js.** Import this file in **server.js;**

const app = rquire(‘./index’);

4: Create Server for run:

const server = http.createServer(app);

//here app mean the index file we import first with name app;

5: Run server with **listen():**

server.listen(process.env.PORT);

//here **env.PORT** is PORT fetch from **.env file.**

**Index.js File:**

1: Import **express** framework:

const express = require(‘express’);

2: Import **cors:**

const cors = require(‘cors’);

3: Initialize **app**:

const app = EXPRESS();

app.use (cors());

app.use(express.urlencoded({extended: true}));

app.use(express.json());

module.exports = app;

**Connection.js file:**

Create new file connection.js file at root folder;

1: import mysql;

Const mysql = require(‘mysql’);

2: Import **.env** file;

Require(‘dotenv’).config();

3: Create variables in **.env file:**

DB\_PORT = 3306

DB\_HOST = localhost

DB\_USERNAME = root

DB\_PASSWORD =

DB\_NAME = cafenodejs

4: Create connection in **connection.js file.**

Var connection = mysql.createConnection({

port: process.env.DB\_PORT,

host: process.env.DB\_HOST,

user: process.env.DB\_USERNAME,

password: process.env.DB\_PASSWORD,

database: process.env.DB\_NAME

});

Connection.connect((err)=>{

If(!err){

Console.log(‘connected’);

}

Else{

Console.log(err);

}

});

5: Export Connection:

Module.exports = connection;

6: Import **connection file** in **index.js** file:

Const connection = require(‘./connection’);

Now structure is ready.

**Create Table:**

1: Create new file at root folder **table.js:**

Use it for create table in database.

2: Table Structure:

create table user(

id int primary key AUTO\_INCREMENT,

name varchar(250),

contactNumber varchar(250),

email varchar(50),

password varchar(250),

status varchar(20),

role varchar(20),

UNIQUE (email)

);

Basically this file is used for create tables in database and it is optional we can directly create table in phpMyAdmin.

**Routes:**

1: Create new folder name **Route** in root directory and create your route file in that folder our file name is **user.js;**

2: Import that file in **index.js** file and use it.

const userRouter = require(‘./router/user’);

app.user (‘/user’, userRoute);

In Router file **user.js**:

1: Import express:

Const express = require(‘express’);

2: Import **Connection** file for create query.

const connection = require(‘../connection’);

3: Import **Router**:

const router = express.Router();

4: Export router that we create in point 3.

module.exports = router;

**Create API in this(user.js) file:**

Create POST api:

**SignUp API:**

Router.post(‘/signup’ , (req, res) =>{

Let user = req.body;

Query = ‘SELECT email , password , role , status from user where email=?’

Connection.query(query, [user.email], (err , results)=>{

If(!err){

If(result.length <=0){

query =”INSERT into users(name , contactNumber , email , password, status , role)values(?, ? , ? , ? ‘false’ , ‘user’)”;

connection.query(query , [user.name , user.contactNumber , user.email , user.password], (err , results)=>{

if(!err){

resturn res.status(200).json({message:’successfully registered’});

}

Else{

Resturn res.status(500).json(err);

}

})

}

Else{

Return res.status(400).json({message:”Email Already Exist”});

}

}

Else{

Return res.status(500).json(err);

}

})

})

**JWT Token:**

1: Install **Jwt**:

npm install –save jsonwebtoken

**Use JWT:**

We use jwt in login api:

**How JWT token Work:**

1: Install jwt:

Npm install –save jsonwebtoken

2: Import **jwt** in the route file where you want to use here we use it.

const jwt = require(‘jsonwebtoken’);

3: create random string for **Jwt:**

**For create string:**

A: in cmd run node.

B: After node is run then run this command.

require('crypto').randomBytes(64).toString('hex')

here 64 is no.s of byte we can also use 128 or any other number.

4: Copy this created string and paste in **.env** file.

5: Import **.env** file in required route file here our file is **user.js.**

6: use **jwt.sign()** for create token.

 const response = {email:results[0].email , role:results[0].role};

 const accessToken =jwt.sign(response,process.env.ACCESS\_TOKEN,{expiresIn:'8h'});

 res.status(200).json({status:true, message:"User Login Successfully ",token:accessToken});

here email and role use to store in token and ACCESS\_TOKEN is string that we generate and paste in **.env** file.

**Complete Code for login and generate JWT is:**

router.post('/login' , (req , res)=>{

    const user = req.body;

    query = "select email , password, role, status from users where email =?";

    connection.query(query , [user.email] , (err , results)=>{

        if(!err){

            if(results.length <=0 || results[0].password != user.password){

                return res.status(401).json({status:false, message:'Incorrect Username or Password'});

            }

            else if(results[0].status === 'false'){

                return res.status(401).json({status:false,message:"Wait For Admin Approvel"});

            }

            else if(results[0].password == user.password){

                const response = {email:results[0].email , role:results[0].role};

                const accessToken = jwt.sign(response , process.env.ACCESS\_TOKEN, {expiresIn:'8h'});

                res.status(200).json({status:true, message:"User Login Successfully", token:accessToken});

            }

            else{

                return res.status(400).json({status:false, message:"Something Went Wrong"});

            }

        }

        else{

            return res.status(500).json(err);

        }

    })

})

**NodeMailer:(**use for send emails in node**)**

1: Install **Node Mailer:**

npm install --save nodemailer

**Forget Password API:**

1: Import **nodeMailer** in required route file here our file is **user.js.**

2: Create two more variables in **.env file.**

A: Email = [youremail@gmail.com](mailto:youremail@gmail.com)

B:Password = yourpassword

3: Create Transporter this is used for send mail:

Var transporter = nodemailer.createTransport({

service:’gmail’,

auth:{

user: process.env.EMAIL,

pass: process.env.PASSWORD

}

});

4: **Forgot Password** Api.

router.post('/forgotPassword', (req, res)=>{

    const user = req.body;

    query = "select email, password from users where email=?";

    connection.query(query , [user.email], (err , results)=>{

        if(!err){

            if(results.length <= 0){

                return res.status(200).json({status:true, message:'Password sent successfully to your email'});

            }

            else{

                var mailOptions = {

                    from :process.env.EMAIL,

                    to: results[0].email,

                    subject:'Password by Cafe Management',

                    html:'<p><b>Your loin Details is</b><br>Email:<b>'+results[0].email+'</b><br>Password:<b>'+results[0].password+'</b><br><a href="http://localhost:4200/">Login</a></p>'

                };

                transporter.sendMail(mailOptions, function(error, info){

                    if(error){

                        console.log(error);

                    }

                    else{

                        console.log('email sent '+ info.response);

                    }

                });

                return res.status(200).json({status:true, message:'Password sent successfully to your email'});

            }

        }

        else{

            return res.status(500).json(err);

        }

    })

});

Here we use **transporter.sendMail** function for send email.

**Get User API:**

**Authenticate User Mean Check Header Token:**

1: Create new folder name **services** at root directory.

2: Create new file with name **authentication** in services folder.

3: Create new file with name **checkRole** in services folder.

**Authentication.js:**

1: Import **.env** file for access ACCESS\_TOKEN in that file.

require(‘dotenv’).config();

2: Import **JWT** in this file.

Const jwt = require(‘jsonwebToken’);

3: Create function with name **AuthenticateToken();**

4:In function **AuthenticateToken();**

function authenticateToken(req , res , next){

    const authHeader = req.headers['authorization'];

    const token = authHeader && authHeader.split[' '][1];

    if(token == null)

    return res.sendStatus(401);

    jwt.verify(token , process.env.ACCESS\_TOKEN , (err , response)=>{

        if(err)

            return res.sendStatus(403);

            res.locals = response;

            next()

    })

}

5: Export AuthenticateToken function.

module.exports = {authenticateToken: authenticateToken};

**CheckRole.js File:**

1: import **.env** file.

2: Add **User = user** in **.env** file for use in checkRole.js file.

3: Write function **checkRole:**

function checkRole(req, res, next){

    if(res.locals.role == process.env.USER)

    res.sendStatus(401);

    else

    next()

}

4: Export checkRole function:

module.exports = {checkRole:checkRole}

**How to use Authentication and checkRole in api.**

1: Import these files in where we create our api’s here we use user.js file.

var auth = require('../services/authentication');

var checkRole = require('../services/checkRole');

2: now add call this files methods in where we create api’s like:

router.get('/get', auth.authenticateToken, checkRole.checkRole,  (req , res) =>{})

here **auth.authenticateToken** is used for check authentication token in header.

And **checkRole.checkRole** is return the user role mean admin.