Carepact Infotech pvt ltd

Satendra Sahu
Senior Software Developer

28/12/2021

—

Soket Programming + logs + more

—

Joseph Manavalan Sir

Topics

1. Socket
2. Carepact logs
3. Awslogs
4. Awslogs for local
5. Awslogs for server
6. Save image
7. Show logs on frontend
8. For admin user
9. For super admin
10. Api creation

THE PROCESS

Socket

Socket provide a functionality to create a chat application, Sockets have traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.

Idea is that when user write a message, then server can send message to one or more client

Requirement for socket

1. Download Node js
2. Download Express Js
3. Create a server.js file
4. Download Socket.io ➔ npm install socket.io

Socket.IO is composed of two parts:

A server that integrates with (or mounts on) the Node.JS HTTP Server socket.io

A client library that loads on the browser side socket.io-client

Create page like following page

```
import express from 'express'
const app = express()
const server = http.createServer(app);
import { Server } from "socket.io";
const io = new Server(server);
const port = process.env.PORT || 5000


app.get('/newsocket', (req, res) => {
    res.sendFile(__dirname + '/newSocket.html')
    console.log('one socket connected')

})


io.on('connection', (socket) => {

    socket.on('chat message', msg => {
        io.emit('chat message', msg);
        // console.log(outputData)

        if (typeof(msg) === 'object') {
            //console.log(typeof(msg), "hello this is object")
            tempCare(msg, SocketImgPath)
                // tempCust(msg, SocketImgPath)
            tempAWS(msg, SocketImgPath)
        } else {
            const obj = JSON.parse(msg);
            var outputData = obj
                //console.log("hello this is string ")
            tempCare(outputData, SocketImgPath)
                // tempCust(outputData, SocketImgPath)
            tempAWS(outputData, SocketImgPath)
        }


    });
});



server.listen(port, console.log(`server running at port ${port}`))
```

5. Create a htmlPage
   Like → newSocket.html

```
<!DOCTYPE html>

<html>

<head>
    <title>Socket.IO chat</title>
    <style>
        body {
            margin: 0;
            padding-bottom: 3rem;
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica,
Arial, sans-serif;
        }

        #form {
            background: rgba(0, 0, 0, 0.15);
            padding: 0.25rem;
            position: fixed;
            bottom: 0;
            left: 0;
            right: 0;
            display: flex;
            height: 3rem;
            box-sizing: border-box;
            backdrop-filter: blur(10px);
        }

        #input {
            border: none;
            padding: 0 1rem;
            flex-grow: 1;
            border-radius: 2rem;
            margin: 0.25rem;
        }

        #input:focus {
            outline: none;
        }

        #form>button {
            background: #333;
            border: none;
            padding: 0 1rem;
            margin: 0.25rem;
```

```
            border-radius: 3px;
            outline: none;
            color: #fff;
        }

        #messages {
            list-style-type: none;
            margin: 0;
            padding: 0;
        }

        #messages>li {
            padding: 0.5rem 1rem;
        }

        #messages>li:nth-child(odd) {
            background: #efefef;
        }
    </style>
</head>

<img> Welcome to new socket


<ul id="messages"></ul>
<form id="form" action="">
    <input id="input" autocomplete="off" /><button>Send</button>
</form>
<script src="/socket.io/socket.io.js"></script>

<script>
    var socket = io();

    var messages = document.getElementById('messages');
    var form = document.getElementById('form');
    var input = document.getElementById('input');

    form.addEventListener('submit', function(e) {
        e.preventDefault();
        if (input.value) {
            socket.emit('chat message', input.value);
            input.value = '';
        }
    });

    socket.on('chat message', function(msg) {
        var item = document.createElement('li');
```

```
        item.textContent = msg;
        messages.appendChild(item);
        window.scrollTo(0, document.body.scrollHeight);
    });
</script>
</body>

</html>
```

6. Need To focus here on some points
   1. For socket listen a request on server page with the help of node core module name as http Not by express
   Eg →

```
import http from 'http';

const server = http.createServer(app);

server.listen(port, console.log(`server running at port ${port}`))

// never user this for socket
// app.listen(port, console.log(`server running at port ${port}`)
```

   2. In html page event name should be same as defined in server page

```
io.on('connection', (socket) => {

    socket.on('chat message', msg => {
        io.emit('chat message', msg);
    });
});
```

Here event name is "'chat message'" it will be same on html page
Like this

```
<script>
    form.addEventListener('submit', function(e) {
        e.preventDefault();
        if (input.value) {
            socket.emit('chat message', input.value);
        }
    });
```

```
    socket.on('chat message', function(msg) {
    });
</script>
```

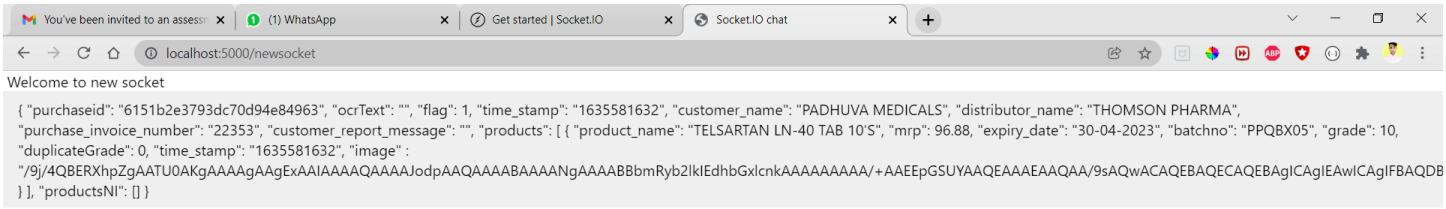
3. Import socket in server.js

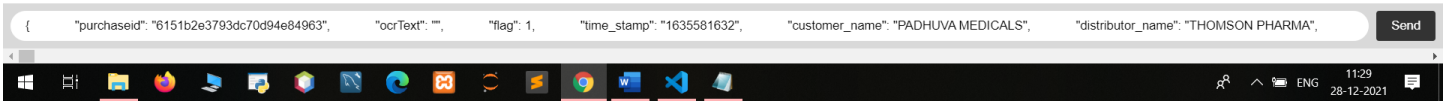
```
import { Server } from "socket.io";
```

4. Also add in html page

```
<script src="/socket.io/socket.io.js"></script>
```

7. If every thing is connected then
    1. Go on browser and fetch url like this
    2. localhost:5000/newsocket





8. Similarly all sockets will implement like this
    1. Ocrinput
    2. Inventory
    3. purchase

Note For More Knowledge Follow this document https://socket.io/get-started/chat

Logs

Carepact logs

Carepact logs is used for save data in local system which is coming through Socket, either from browser or from Android

Requirement for logs

1. on sever page
2. import 3 functions

```
import carePactLogsRoutes from './routes/carePactLogsRoutes.js'
import { tempCare } from './controllers/carepactLogsController.js'
```

```
io.on('connection', (socket) => {

    socket.on('chat message', msg => {
        io.emit('chat message', msg);
        // this is for android
```

```
        if (typeof(msg) === 'object') {
            //console.log(typeof(msg), "hello this is object")
            tempCare(msg, SocketImgPath)
                // tempCust(msg, SocketImgPath)
            tempAWS(msg, SocketImgPath)
        } else {
             //this is for browser
            const obj = JSON.parse(msg);
            var outputData = obj
                //console.log("hello this is string ")
            tempCare(outputData, SocketImgPath)
                // tempCust(outputData, SocketImgPath)
            tempAWS(outputData, SocketImgPath)
        }


    });
});
```

3. create router/ model/ controller page for carepactlog page

4 check on project for these pages

Description ->

Here data is coming from socket and handled by chat message event after that it is coming on msg variable.

If data is coming through android then typeof(msg) will be object which is handled in if condition. Else will handled in else case on above code.

After that it will call two fuctions

```
tempCare(msg, SocketImgPath)
tempAWS(msg, SocketImgPath)
```

here
msg ➔ socket input
SocketImgPath ➔ path of image which is saving in uploads/socketimg directory

1. on carepactController.js page

   request will handle here

```
const tempCare = async(data, SocketImgPath) => {
    var Data = [data]
    await saveCarePactLogs(Data, SocketImgPath)
}
```

Here incoming data is converting into array of objects

And calling this function

```
saveCarePactLogs = async(Data, SocketImgPath)
```

here all data is saving into related collection in mongo db

here three thing important

1. save image

```
            let base64Image = Data.image.split(';base64,').pop();
            let SocketImgName = `ProductImg${Data.time_stamp}.png`
            fs.writeFile(`${SocketImgPath}/${SocketImgName}`, base64Image, {
                encoding: 'base64'
            }, function(err) {
                console.log('Image File created in carepactlog for Products');
            });
```

2. either data will save Product array or in ProductNI Array

   in product array has one product one image
   like this →

```
{
        "purchaseid": "6151b2e3793dc70d94e84963",
        "ocrText": "",
        "flag": 1,
        "time_stamp": "1635581632",
        "customer_name": "PADHUVA MEDICALS",
        "distributor_name": "THOMSON PHARMA",
        "purchase_invoice_number": "22353",
        "customer_report_message": "",
        "products": [
            {
                "product_name": "TELSARTAN LN-40 TAB 10'S",
                "mrp": 96.88,
                "expiry_date": "30-04-2023",
                "batchno": "PPQBX05",
                "grade": 10,
                "duplicateGrade": 0,
                "time_stamp": "1635581632",
"image" : “”
}],
ProductsNI : []
}
```

In productNi Array has one image and multiple products
Like this


```
{
        "purchaseid": "6151b2e3793dc70d94e84963",
        "ocrText": "",
        "flag": 1,
        "time_stamp": "1635581632",
        "customer_name": "PADHUVA MEDICALS",
        "distributor_name": "THOMSON PHARMA",
        "purchase_invoice_number": "22353",
        "customer_report_message": "",
         "image" : “”

        "products": [
             {
                  "product_name": "TELSARTAN LN-40 TAB 10'S",
                  "mrp": 96.88,
                  "expiry_date": "30-04-2023",
                  "batchno": "PPQBX05",
                  "grade": 10,
                  "duplicateGrade": 0,
                  "time_stamp": "1635581632",
},
{
                  "product_name": "TELSARTAN LN-40 TAB 10'S",
                  "mrp": 96.88,
                  "expiry_date": "30-04-2023",
                  "batchno": "PPQBX05",
                  "grade": 10,
                  "duplicateGrade": 0,
                  "time_stamp": "1635581632",
},
{
                  "product_name": "TELSARTAN LN-40 TAB 10'S",
                  "mrp": 96.88,
                  "expiry_date": "30-04-2023",
                  "batchno": "PPQBX05",
                  "grade": 10,
                  "duplicateGrade": 0,
                  "time_stamp": "1635581632",
}

],
ProductsNI : []
}
```

So here implementation will change according to products Array and ProductsNi array Like this

This is for product Array

```
data.products.map((Data) => {
            let base64Image = Data.image.split(';base64,').pop();
            let SocketImgName = `ProductImg${Data.time_stamp}.png`
            fs.writeFile(`${SocketImgPath}/${SocketImgName}`, base64Image, {
                encoding: 'base64'
            }, function(err) {
                console.log('Image File created in carepactlog for Products');
            });
            if (Data.length !== 0) {
                Data.date = currentDate
                Data.date_time = currentDateTime
                Data.imageName = SocketImgName
                Data.imagePath = `${SocketImgPath}/${SocketImgName}`
            }
            Products = Data
        })
        data.productsNI.map((Data) => {
            if (Data.length !== 0) {
                Data.date = currentDate
                Data.date_time = currentDateTime
                Data.time_stamp = TimeStamp
            }
            ProductsNI = Data
        })
```

This is for ProductsNi Array

```
if (data.image) {
            base64Image = data.image.split(';base64,').pop();
            SocketImgName = `ProductImg${TimeStamp}.png`
            fs.writeFile(`${SocketImgPath}/${SocketImgName}`, base64Image, {
                encoding: 'base64'
            }, function(err) {
                console.log('Image File created in customerlog for ProductsNI');
            });
            ProductsNI = [...ProductsNI]
            ProductsNI.push({
                image: data.image,
```

```
                imageName: SocketImgName,
                imagePath: `${SocketImgPath}/${SocketImgName}`
            })
        }
```

After that call other functions according to requirements

```
saveCarePactLogs,
    tempCare,
    getCarePactLogs,
    saveCarePactLogsInAWSServer,
    awssave,
    getCarePactLogtosend,
    awsflagUpdation,
    findProductByInvoiceNo,
    findProductByInvoiceNo_ProductName,
    findProductByInvoiceNo_ProductName_BatchNo,
    findProductByInvoiceNo_ProductName_Date,
    findProductByInvoiceNo_ProductName_BatchNo_Date,
    deletelogsById
```

Same process will we for AwsLogsContrller.js

AwsLogsController.js

AwslogsController is used for save data in local system which is coming through Socket, either from browser or from Android.

What here main reason to implement this it is storing logs for limited time,

1. first logs will store in awslogs collection in database
2. first time awsflag will be false
3. function will call from server.js to save on server
4. then it will check
    1. internet is connected
    2. awsflag is false
    3. awsloggModel.length > 0
    4. then send data to server
    5. if data is sended to server then
    6. logs will be delete from awslog automatically
    7. storage will we free

Main thing to be remember

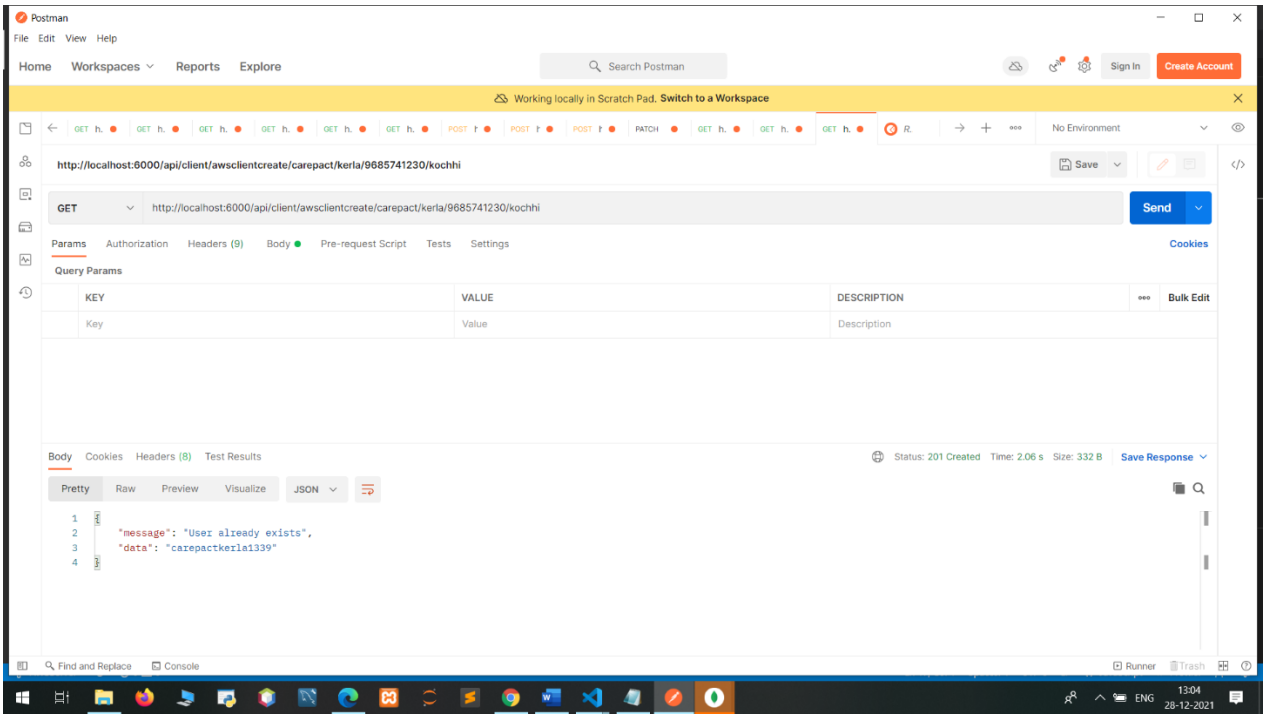
Run api on postman to get uniqid for server

```
const response = await
fetch(`http://3.7.50.74:5000/api/client/awsclientcreate/${name}/${address}/${phone}/${locatio
n}`);
```

Run api on postman to get uniqid for local aws

```
        const response = await
fetch(`http://localhost:6000/api/client/awsclientcreate/${name}/${address}/${phone}/${locatio
n}`);
```

and fetch unique id




```
    let uniqid = 'carepactkerla1339'
```

Note → don't try to create or /fetch uniqid directly on logs saving time always fetch through postman and save here manually

For sending logs local to server call this function

```
const saveCarePactLogsInAWSServer = async(req, res)
```

here function send the logs to server based on conditions

1. internet is connected
2. awsflag is false
3. awsloggModel.length > 0
4. then send data to server
5. if data is sended to server then

and for deleting logs call this function

1. logs will be delete from awslog automatically
2. storage will we free

```
const deleteImageFromLogs = async(req, res) =>
```

and for sending data to logs call this function from server.js each every min

```
const saveInAWS = async() => {
    try {
        const result = await awsLogsModel.find()
        result.length > 0 ? saveCarePactLogsInAWSServer() : ""
    } catch (err) {
        console.log(err.message)
    }
}

//for 1 min
setInterval(saveInAWS, 1000*60)
```

same process will be follow for AWS Server implementation

1. first request will come and check that customer is available or not
2. if available then a folder will create with the help of customer uniqid
3. else directly save image on existing folder
4. and will save logs same as carepactlogs on local

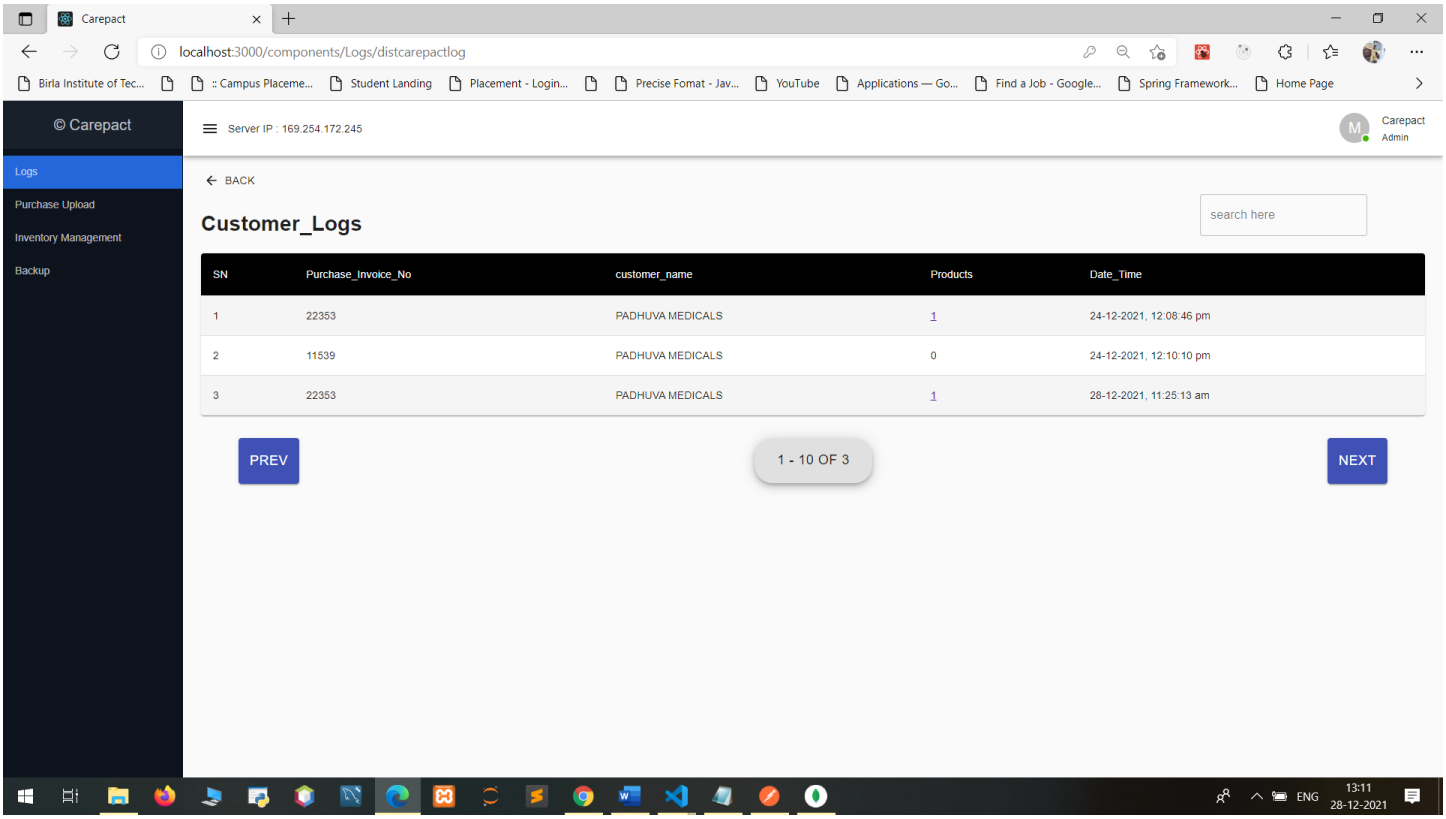
important function for client creation on server

```
const checkidexisit = asyncHandler(async(req, res, next)
const create = asyncHandler(async(req, res) =>
```

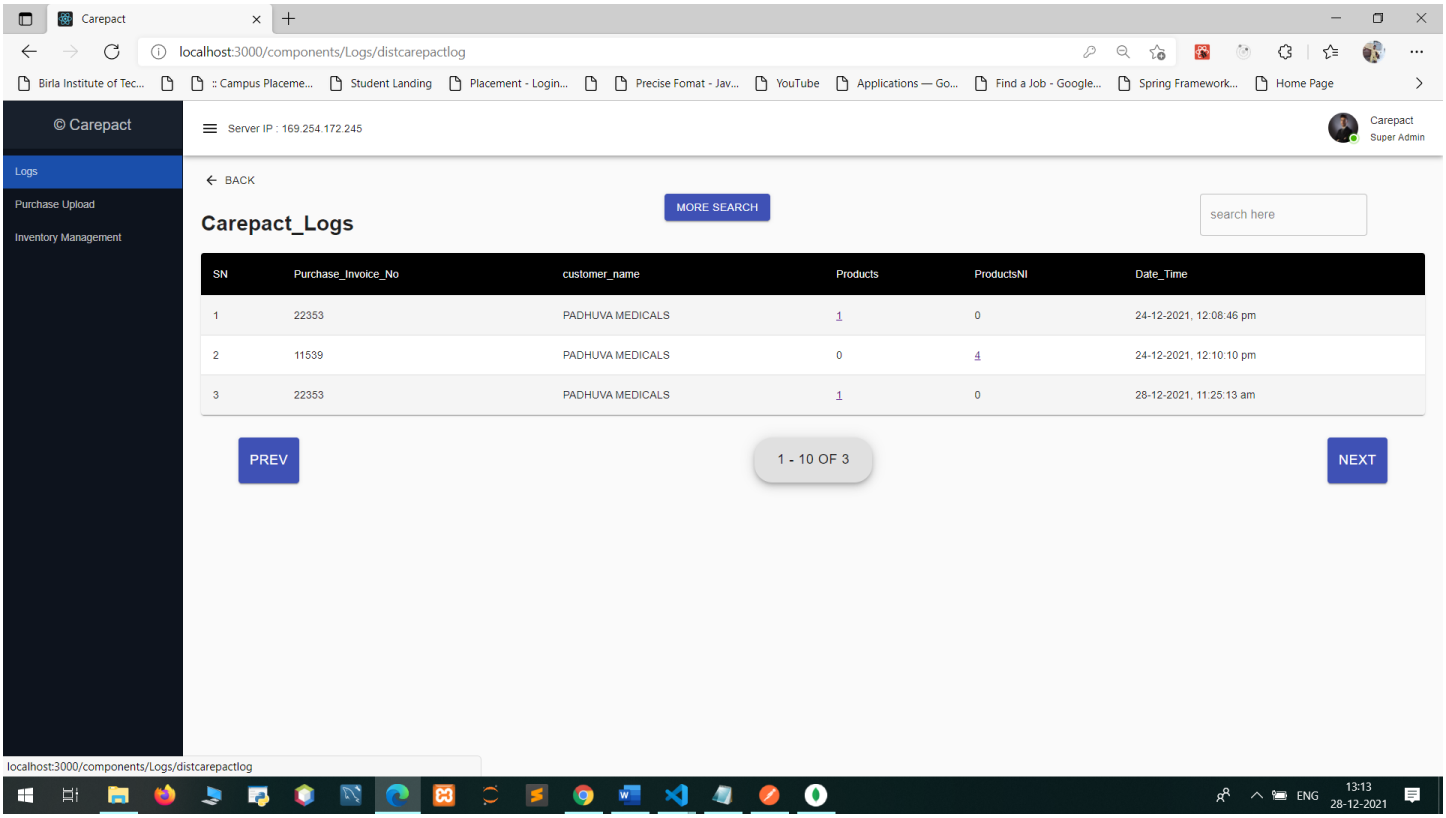
Frontend

Here data will show based on two situation

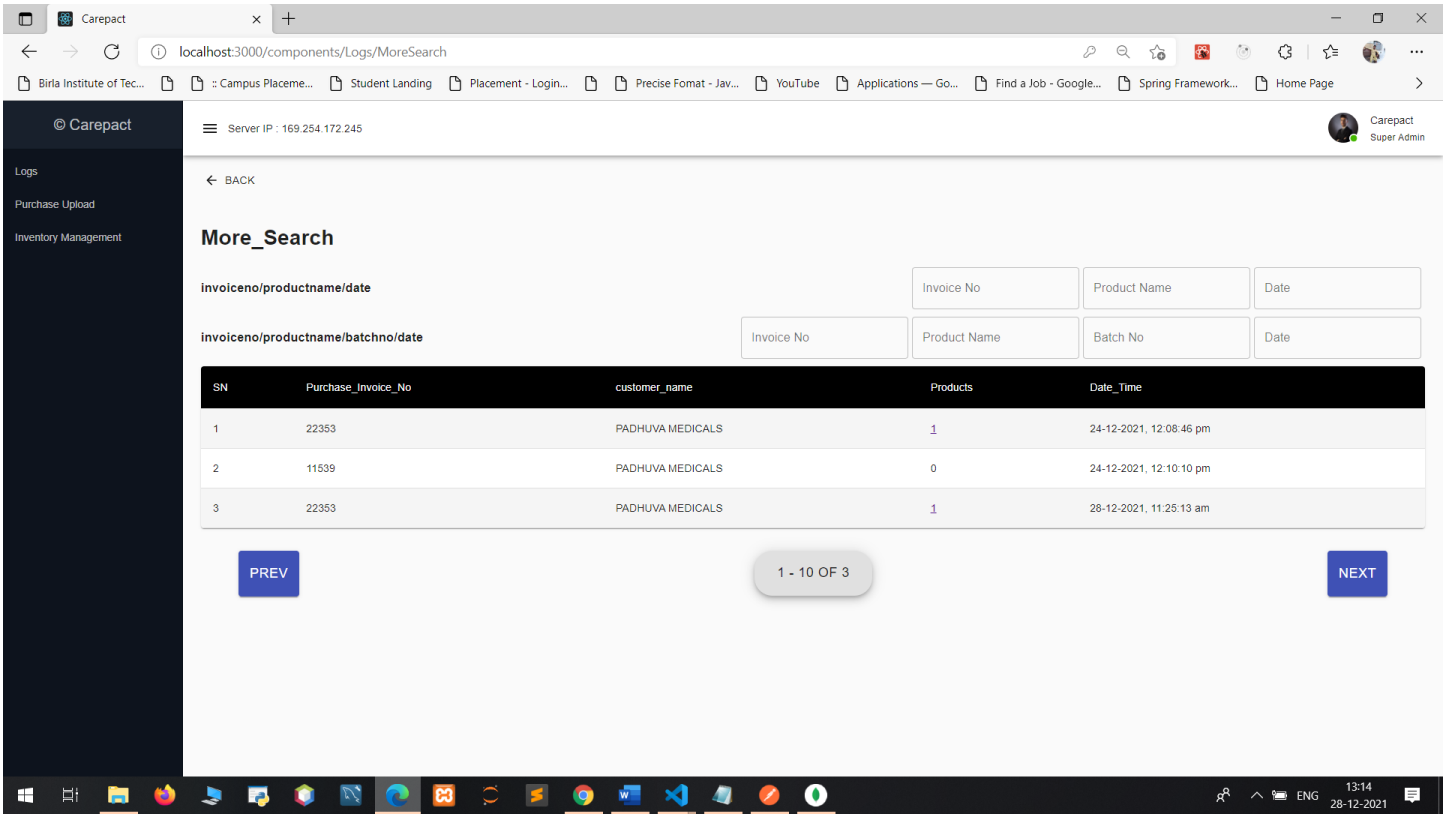
1. for client
   When login as a admin



2. for carepact
   When login as a super admin



Super admin can search based on different conditions



Different Apis

For CarepactlogsController

http://localhost:5000/api/carepactlogs/carelogs ==> getCarePactLogs
http://localhost:5000/api/carepactlogs/awssend ==> getCarePactLogtosend
http://localhost:5000/api/carepactlogs/awssaveflag ==> awssave
http://localhost:5000/api/carepactlogs/awsflag/:id ==> awsflagUpdation
http://localhost:5000/api/carepactlogs/carelogs/:invoiceno ==> findProductByInvoiceNo
http://localhost:5000/api/carepactlogs/carelogs/:invoiceno/:productname ==> findProductByInvoiceNo_ProductName
http://localhost:5000/api/carepactlogs/carelogs/:invoiceno/:productname/:batchno ==>
findProductByInvoiceNo_ProductName_BatchNo
http://localhost:5000/api/carepactlogs/carelogs/date/:invoiceno/:productname/:date ==> findProductByInvoiceNo_ProductName_Date
http://localhost:5000/api/carepactlogs/:invoiceno/:productname/:batchno/:date ==>
findProductByInvoiceNo_ProductName_BatchNo_Date
http://localhost:5000/api/carepactlogs/carelogs/:id ==> deletelogsById

For AwsLogsController

http://localhost:5000/api/awslogs/carelogs ==> getCarePactLogs
http://localhost:5000/api/awslogs/awssend ==> getCarePactLogtosend
http://localhost:5000/api/awslogs/awssaveflag ==> awssave
http://localhost:5000/api/awslogs/awsflag/:id ==> awsflagUpdation
http://localhost:5000/api/awslogs/carelogs/:invoiceno ==> findProductByInvoiceNo
http://localhost:5000/api/awslogs/carelogs/:invoiceno/:productname ==> findProductByInvoiceNo_ProductName
http://localhost:5000/api/awslogs/carelogs/:invoiceno/:productname/:batchno ==> findProductByInvoiceNo_ProductName_BatchNo
http://localhost:5000/api/awslogs/carelogs/date/:invoiceno/:productname/:date ==> findProductByInvoiceNo_ProductName_Date
http://localhost:5000/api/awslogs/:invoiceno/:productname/:batchno/:date ==> findProductByInvoiceNo_ProductName_BatchNo_Date
http://localhost:5000/api/awslogs/carelogs/:id ==> deletelogsById

For LocalAws Server

http://localhost:6000/api/logs/create ==> checkidexisit, folderexist, saveCarePactLogs
http://localhost:6000/api/logs/alllogs ==> getallLogs
http://localhost:6000/api/logs/getbyuniqid/:uniqid ==> getLogsbyuniqid
http://localhost:6000/api/logs/getbyinvoice/:invoice ==> getLogsbyinvoiceno
http://localhost:6000/api/logs/getbyinvoice/:invoice/:timestamp ==> findDataByinvoiceAndTimeStamp
http://localhost:6000/api/logs/carelogs/:invoiceno/:productname ==> findProductByInvoiceNo_ProductName
http://localhost:6000/api/logs/carelogs/:invoiceno/:productname/:batchno ==> findProductByInvoiceNo_ProductName_BatchNo
http://localhost:6000/api/logs/carelogs/date/:invoiceno/:productname/:date ==> findProductByInvoiceNo_ProductName_Date
http://localhost:6000/api/logs/:invoiceno/:productname/:batchno/:date ==> findProductByInvoiceNo_ProductName_BatchNo_Date
http://localhost:6000/api/logs/carelogs/:id ==> deletelogsById

http://localhost:6000/api/client/awsclientcreate/:name/:location/:phone/:address ===>create
http://localhost:6000/api/client/allawsclient ===>allclient

For Aws Server

http://3.7.50.74:5000/api/logs/create ==> checkidexisit, folderexist, saveCarePactLogs
http://3.7.50.74:5000/api/logs/alllogs ==> getallLogs
http://3.7.50.74:5000/api/logs/getbyuniqid/:uniqid ==> getLogsbyuniqid
http://3.7.50.74:5000/api/logs/getbyinvoice/:invoice ==> getLogsbyinvoiceno
http://3.7.50.74:5000/api/logs/getbyinvoice/:invoice/:timestamp ==> findDataByinvoiceAndTimeStamp
http://3.7.50.74:5000/api/logs/carelogs/:invoiceno/:productname ==> findProductByInvoiceNo_ProductName
http://3.7.50.74:5000/api/logs/carelogs/:invoiceno/:productname/:batchno ==> findProductByInvoiceNo_ProductName_BatchNo
http://3.7.50.74:5000/api/logs/carelogs/date/:invoiceno/:productname/:date ==> findProductByInvoiceNo_ProductName_Date
http://3.7.50.74:5000/api/logs/:invoiceno/:productname/:batchno/:date ==> findProductByInvoiceNo_ProductName_BatchNo_Date
http://3.7.50.74:5000/api/logs/carelogs/:id ==> deletelogsById

http://3.7.50.74:5000/api/client/awsclientcreate/:name/:location/:phone/:address ===>create
http://3.7.50.74:5000/api/client/allawsclient ===>allclient