Whatsapp Clone

Powered by … NodeJS
ExpressJS Server
MongoDB Database
Mongoose to access the Messages
Pusher - To make the project real-time

1. Npx create-react-app whatsapp-mern

Create-react-app : tool provided by Facebook

- Deploying front-end on firebase. And Backend on Heroku

Firebase

Firebase.google.com → Web (</>)
1. → Register app
2. → Add Firebase SDK (Click Next)
3. → Install firebase CLI (**npm install -g firebase-tools**)
4. → Deploy to Firebase Hosting

FRONT-END

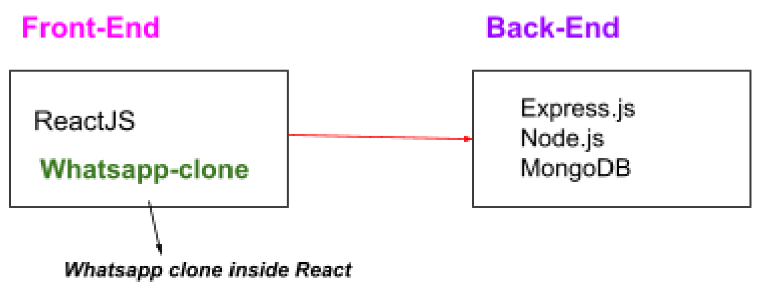
VS Code

In src: (Clean Up)
Delete :
1. App.test.js
2. Logo.svg
3. setupTests.js
4. Delete all of the header tag and its contents.

1. Delete all the pre-styling in App.css → center alignment is removed.
2. To delete the space above the statement → set all margin:0 to remove all the pre margins.
    ***{margin:0}**

Getting the config from firebase:
Firebase
Project Overview → Project settings → Your app → Select Config → Copy code.

Why Firebase if MERN?

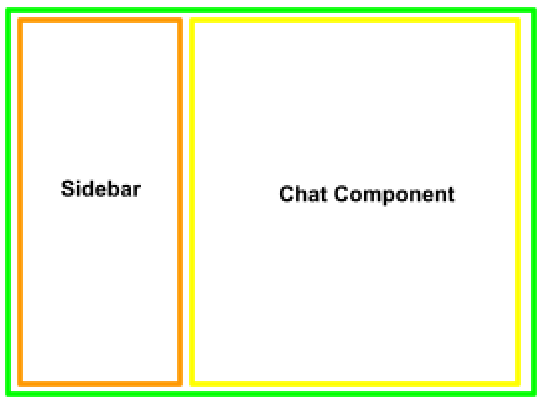


Firebase in front-end for:
Authentication
Hosting

Web Sockets: MongoDB mainstream and pusher → Once there is a change in MongoDB's selected collection that will fire off the change stream and that will fire off pusher which will fire off the fetch function on front-end which will refresh the whole conversation in real-time.

- **# DatabaseURL not in firebase config:**
  Works the same without it.
  But I tried creating a test project and although it didn't initially show up in the config after I went to the Realtime Database tab and clicked to create a database, the database URL property showed up in my web app config in the console.

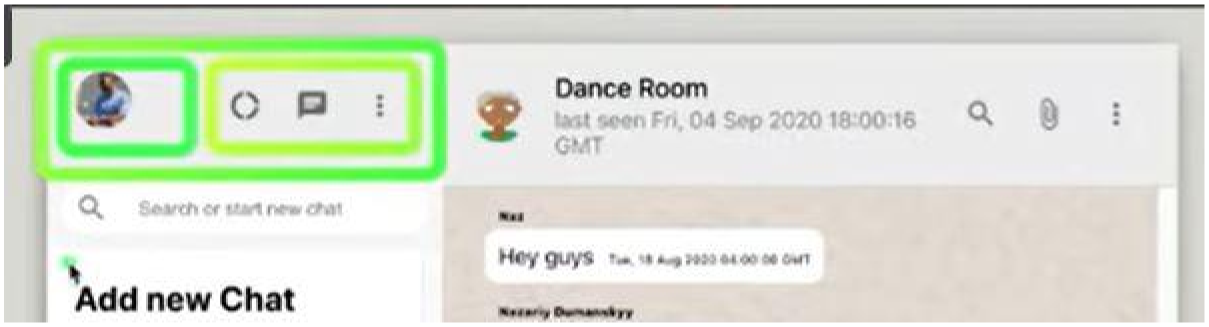
Design



1. Create **Sidebar.js**

Use **rfce.** With **ES7 Snippet. ES&+ React/Redux/React-Native (**dsznajder)

2. Create **Chat.js**
3. Create Sidebar.css and Chat.css



4. [MUI Core: Ready to use components, free forever]()

npm install @mui/material @emotion/react @emotion/styled
npm install @mui/icons-material

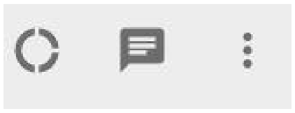
Why material-ui icon?
→ Provides bunch of google interface whirlpool effect and the icons are enhanced.

<DonutLargeIcon /> : import DonutLargeIcon from '@mui/icons-material/DonutLarge';
→ Donut icon is black at first. So need to wrap it in IconButton to make it grey.

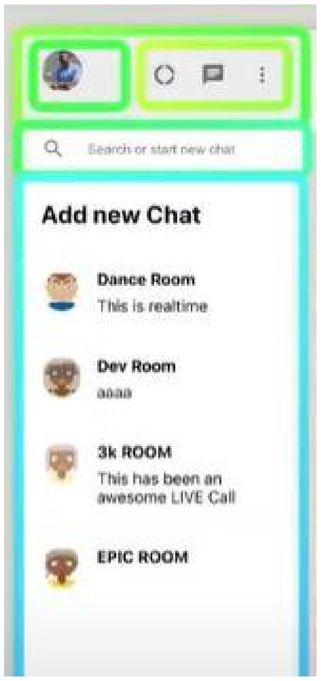
- **<IconButton>** doesn't work in @mui → So need to install **npm install @material-ui/core**

How do you decide views and components in React? Why not use typescript in your react app?
→ For the simplicity of the build, typescript is going to take a long time. We need to simplify the build as much as possible. But working with typescript to build this complexity is gonna take too long.

<ChatIcon />: import ChatIcon from '@mui/icons-material/Chat';
<MoreVertIcon />: import MoreVertIcon from '@mui/icons-material/MoreVert';



<SearchOutlinedIcon />: import SearchOutlinedIcon from '@mui/icons-material/SearchOutlined';


Search or start new chat

Add new Chat

Dance Room
This is realtime

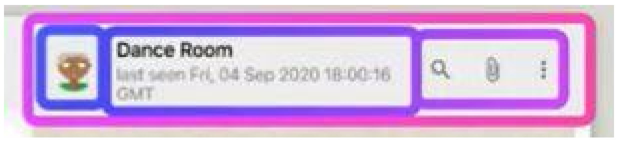
Dev Room
aaaa

3k ROOM
This has been an awesome LIVE Call

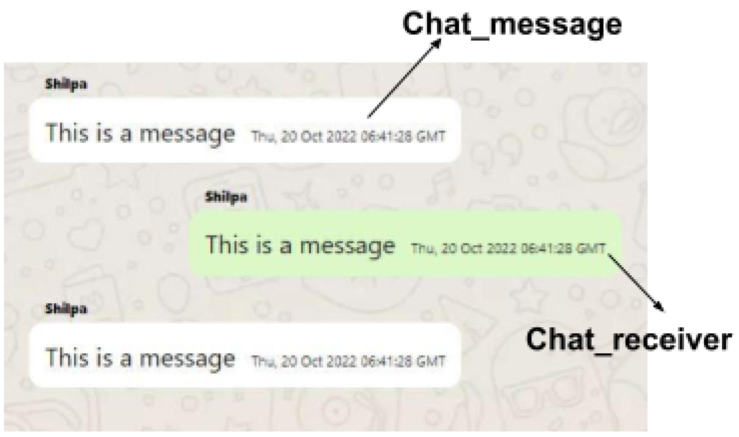
EPIC ROOM

```
<div className="sidebarChat">
      <Avatar />
      <div className="sidebarChat__info">
         <h2>Room Name</h2>
         <p>This is the last message</p>
      </div>
   </div>
```

Chat section:

1. 
   Dance Room
   last seen Fri, 04 Sep 2020 18:00:16 GMT

import SearchOutlinedIcon from '@mui/icons-material/SearchOutlined';
import MoreVertIcon from '@mui/icons-material/MoreVert';
import AttachFileIcon from '@mui/icons-material/AttachFile';



Shilpa
This is a message Thu, 20 Oct 2022 06:41:28 GMT

Shilpa
This is a message Thu, 20 Oct 2022 06:41:28 GMT

Shilpa
This is a message Thu, 20 Oct 2022 06:41:28 GMT

```
.chat__message {
    position: relative;
    font-size: 16px;
    padding: 10px;
    width: fit-content;
    border-radius: 10px;
    background-color: #ffffff;
    margin-bottom: 30px;

}
.chat__receiver{
    margin-left: auto;
    background-color: #dcf8c6;
}
```

- **Scroll bar CSS Solution:**
  Add it in the main css:

```
/* width */
::-webkit-scrollbar {
  width: 10px;
}

/* Track */
::-webkit-scrollbar-track {
  background: #f1f1f1;
}

/* Handle */
::-webkit-scrollbar-thumb {
  background: #888;
}

/* Handle on hover */
::-webkit-scrollbar-thumb:hover {
  background: #555;
}
```

import InsertEmoticonIcon from "@mui/icons-material/InsertEmoticon"
import MicIcon from "@mui/icons-material/Mic";

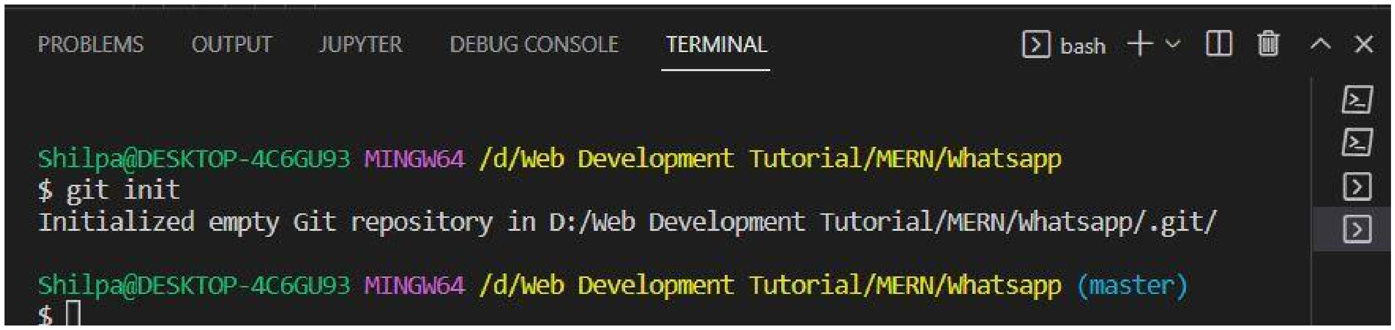
BACK-END

1. Create whatsapp-backend folder.

> cd whatsapp-backend

Git - Downloading Package (git-scm.com)
Make sure to download git and enter the path in environment variables.
C:/Program Files/Git
visual studio - 'git' is not recognized as the name of a cmdlet - Stack Overflow

Select **git bash** in terminal:



```
Shilpa@DESKTOP-4C6GU93 MINGW64 /d/Web Development Tutorial/MERN/Whatsapp
$ git init
Initialized empty Git repository in D:/Web Development Tutorial/MERN/Whatsapp/.git/

Shilpa@DESKTOP-4C6GU93 MINGW64 /d/Web Development Tutorial/MERN/Whatsapp (master)
$ 
```

> npm init
package name: (whatsapp-backend)

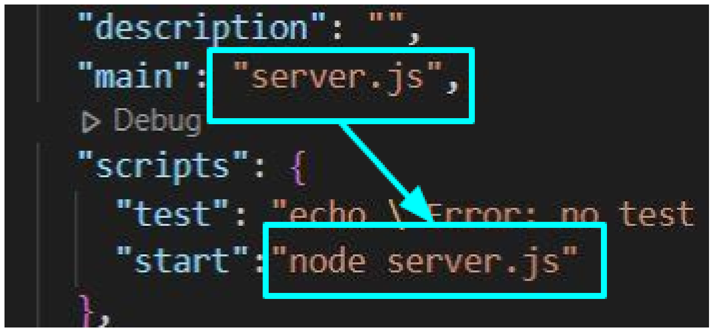
Rename entry point to **server.js**

{

```
  "name": "whatsapp-backend",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Shilpa Vinayaka",
  "license": "ISC"
}
```

→ **Was added to package.json**

Add new script to package.json in "**scripts**" variable.
"start":"node server.js" → This is required to deploy in Heroku later.
Note: The main variable (`server.js`) name should match the new script (`node server.js`):



```
"description": "",
"main": "server.js",
▷ Debug
"scripts": {
  "test": "echo \"Error: no test
  "start":"node server.js"
},
```

- **Express:**
- **Mongoose:** connectivity to mongoDB

> npm i express mongoose

Add a .gitignore file

[**Database Deployments | Cloud: MongoDB Cloud**]

(Creating an API)
const app = express(); //application instance
const port = process.env.PORT || 9000; //Port where we will be running our application

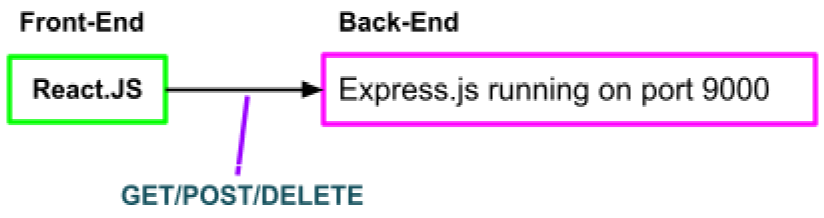
Status Code: Everything within 200 is to say that the request is saying okay
200 → Server is saying Okay
201 → Created (When we send a message and it gets stored successfully in DB)
404 → Page not found
500 → Internal server error

Express.js is running on port 9000

GET/POST/DELETE → Different Requesting sent from frontend to backend server

GET: Fetch info
POST: Push Info

DELETE: Delete Info

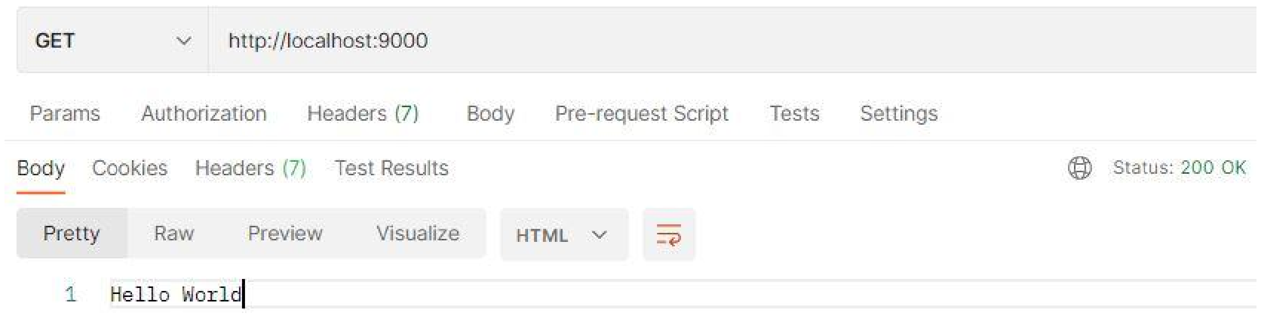


nodemon server.js

POSTMAN

Can use **nodemon - - inspect server.js** and opens up browser debug console for the node server which is really a handy.

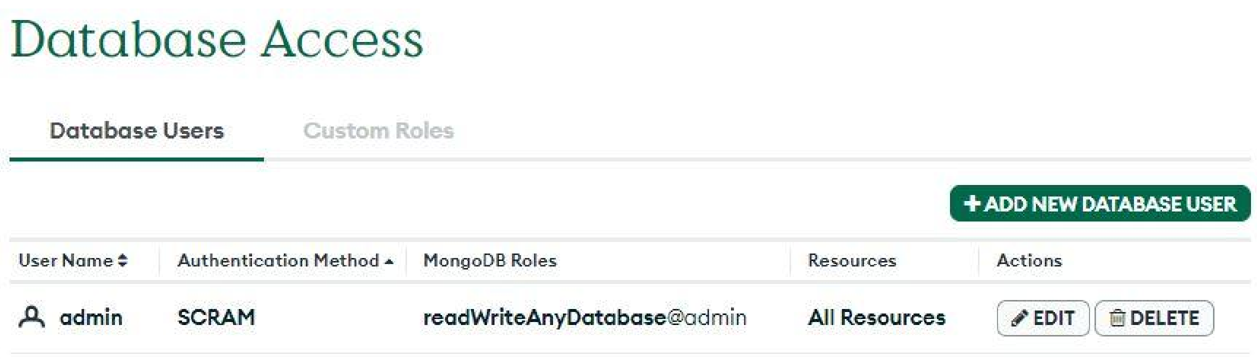
Contacting our port. Check if it's working



```
Hello World
```

Connecting to Database

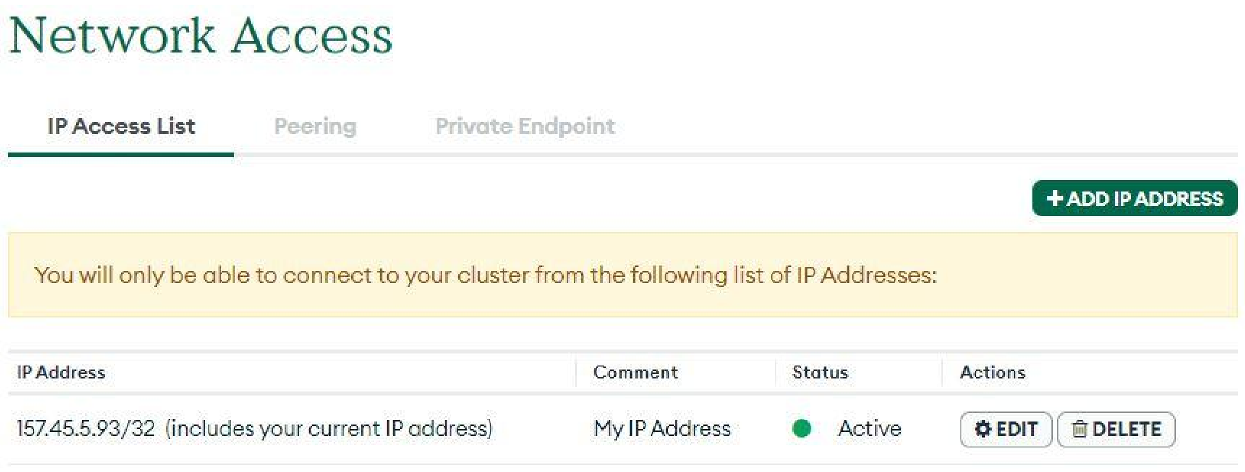
→ Create a database user in MongoDB cloud.
Enter Username and autogenerate password.
Database: Username, Password
admin
20FGhS5XtWib9Z7b



Database Access

User Name	Authentication Method	MongoDB Roles	Resources	Actions
admin	SCRAM	readWriteAnyDatabase@admin	All Resources	EDIT DELETE

Add IP address: Add current IP address (Keep safe from others)



Network Access

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
157.45.5.93/32 (includes your current IP address)	My IP Address	● Active	EDIT DELETE

Connect to Cluster → Connect your application

- Copy the connection String

mongodb+srv://admin:<password>@cluster1.dpdq4x3.mongodb.net/?retryWrites=true&w=majority

```
// DB config
const connection_url
="mongodb+srv://admin:20FGhS5XtWib9Z7b@cluster1.dpdq4x3.mongodb.net/?retryWrites=true&w=majority"
mongoose.connect(connection_url, {
   useCreateIndex: true,
   useNewUrlParser: true,
   useUnifiedTopology: true
})
```

```
(node:24244) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error originated either
by throwing inside of an async function without a catch block, or by rejecting
a promise which was not handled with .catch(). To terminate the node process on unhandled promise rejection,
use the CLI flag `--unhandled-rejections=strict` (see
https://nodejs.org/api/cli.html#cli_unhandled_rejections_mode). (rejection id: 2)
(node:24244) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future,
promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
```

javascript - Server Discovery And Monitoring engine is deprecated - Stack Overflow

Database Schema

dbMessages.js

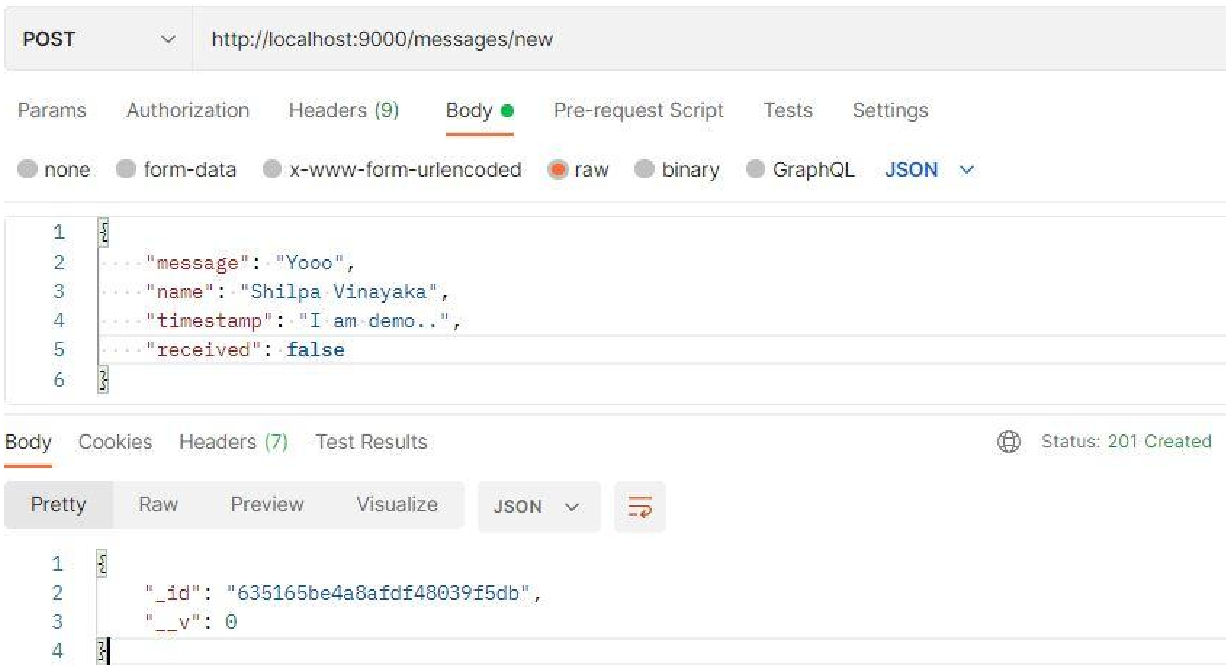
```
import mongoose from "mongoose";
const whatsappSchema = mongoose.Schema({
   message: String,
   name: String,
   timestamp: String,
   received: Boolean

});
export default mongoose.model('messageContent', whatsappSchema);
```

→ **Name of the table(DataSchema):** messageContent

Posting: Storing our data to the DB.
→ **req**.body contains the data from the body.
→ **err**: will store the error message when **if(err)** becomes true. Here status code **500**
→ **data** will store the body data which will be sent to the DB to be stored(**post**)

```
//api to post messages
app.post('/api/v1/messages/new', (req,res)=> {
   const dbMessage = req.body;
   Messages.create(dbMessage, (err, data) => {                 // MongoDB create function
      if(err){
         res.status(500).send(err);
      } else {
         res.status(201).send(data);                           // Creating data: Status code-201
      }
   })
})
```

But this is not the format we want. Not in JSON format.


POST http://localhost:9000/messages/new

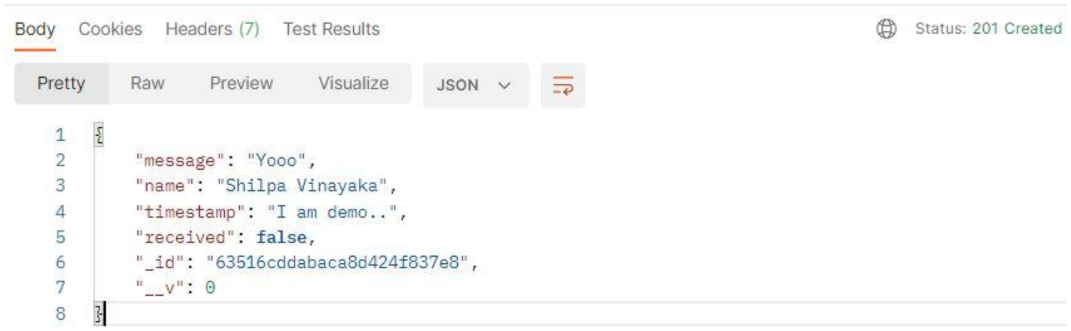
```
{
    "message": "Yooo",
    "name": "Shilpa Vinayaka",
    "timestamp": "I am demo..",
    "received": false
}
```

Status: 201 Created

```
{
    "_id": "635165be4a8afdf48039f5db",
    "__v": 0
}
```

Solution: Middlewares

After entering the **middlewares**:
→ **app.use(express.json())**



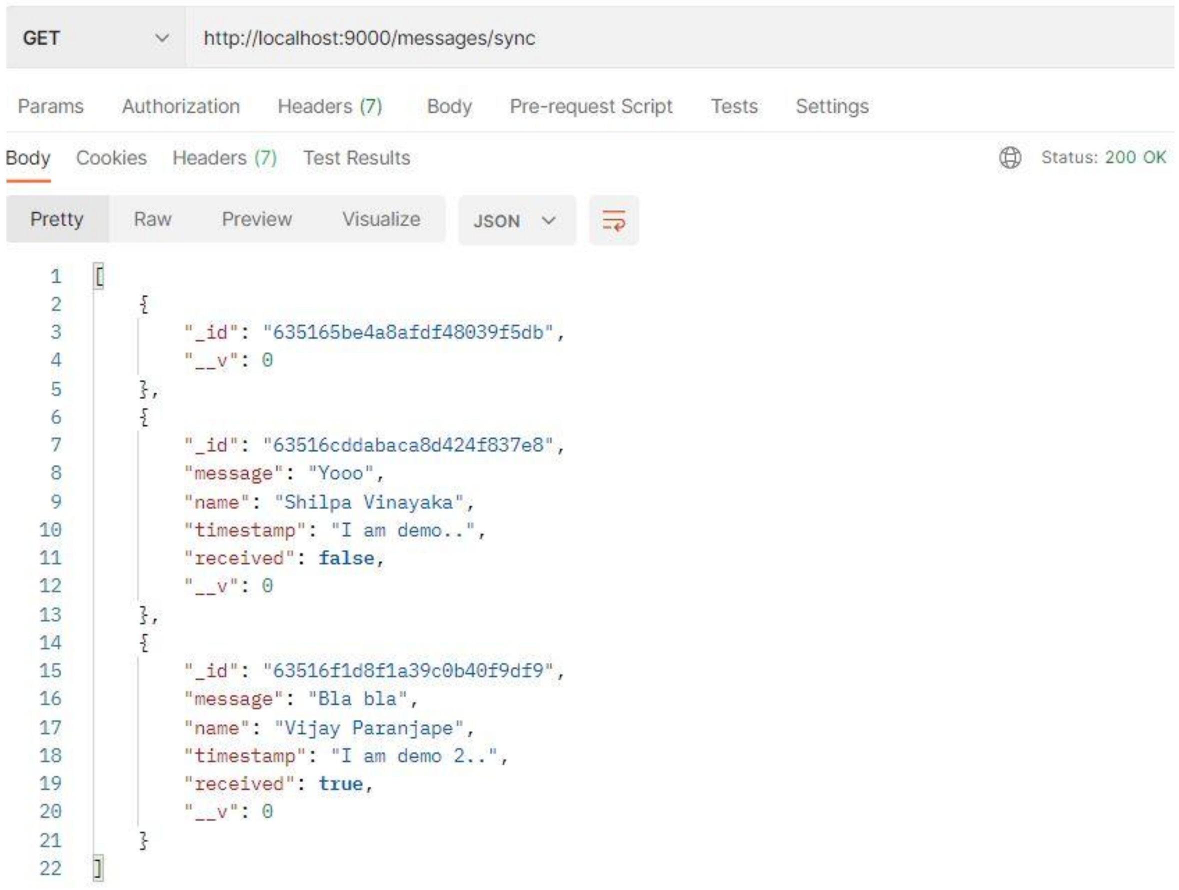
Status: 201 Created

```
{
    "message": "Yooo",
    "name": "Shilpa Vinayaka",
    "timestamp": "I am demo..",
    "received": false,
    "_id": "63516cddabaca8d424f837e8",
    "__v": 0
}
```

Getting the data from the database

```
app.get('/messages/sync', (req,res)=> {
   Messages.find((err, data) => {
      if(err){
         res.status(500).send(err);
      } else {
         res.status(200).send(data);
      }
   })
});
```

→ All the data in DB is obtained:



GET http://localhost:9000/messages/sync

Status: 200 OK

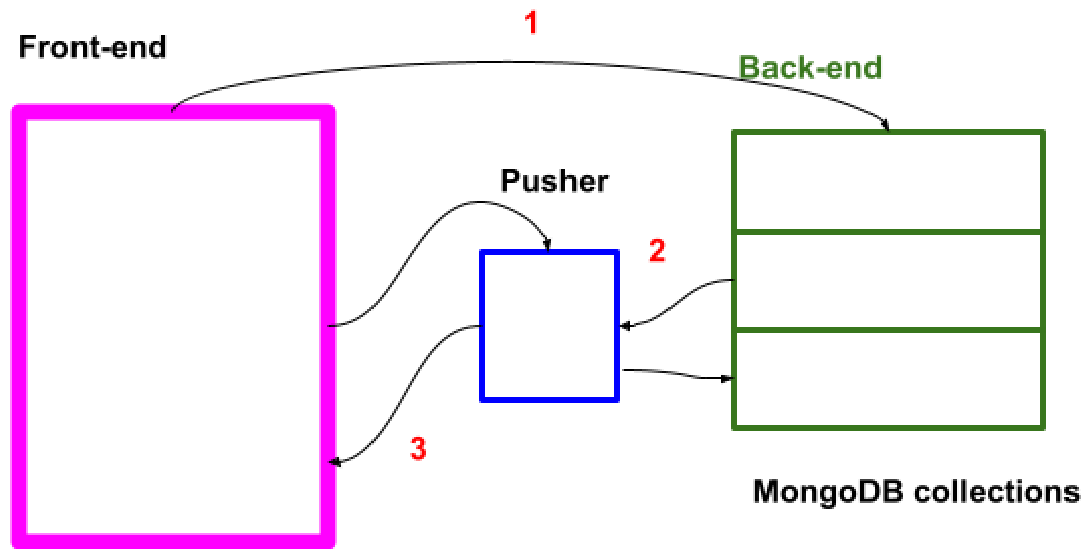
```
[
    {
        "_id": "635165be4a8afdf48039f5db",
        "__v": 0
    },
    {
        "_id": "63516cddabaca8d424f837e8",
        "message": "Yooo",
        "name": "Shilpa Vinayaka",
        "timestamp": "I am demo..",
        "received": false,
        "__v": 0
    },
    {
        "_id": "63516f1d8f1a39c0b40f9df9",
        "message": "Bla bla",
        "name": "Vijay Paranjape",
        "timestamp": "I am demo 2..",
        "received": true,
        "__v": 0
    }
]
```

PUSHER

With the firebase we were using a real-time DB → when something is added or deleted the application is triggered and the exact same time the change on the application is done.
However with MongoDB, this is not the case.

In MongoDB, we have to click refresh buttons or add a functionality that refresh every 5 seconds → to cal in the api. This makes the machine slow.
So as a solution to make the MongoDB real-time, Pusher was a best fitting method to overcome this.

Pusher has a new attribute, that is a MongoDB change stream. *It gives the ability to MongoDB the firebase.* Whenever there is a change in collection (*When new message gets added)* → Change stream triggers the pusher → Uploads the message to Pusher → Connect that to the frontend → **Pusher Server will trigger the front-end and force push down the data (Hence called Pusher)**



Summary:
We need to keep hitting refresh to get new messages. Or a functionality to fetch the messages every 5 seconds. That is wasteful and the best way of doing things.

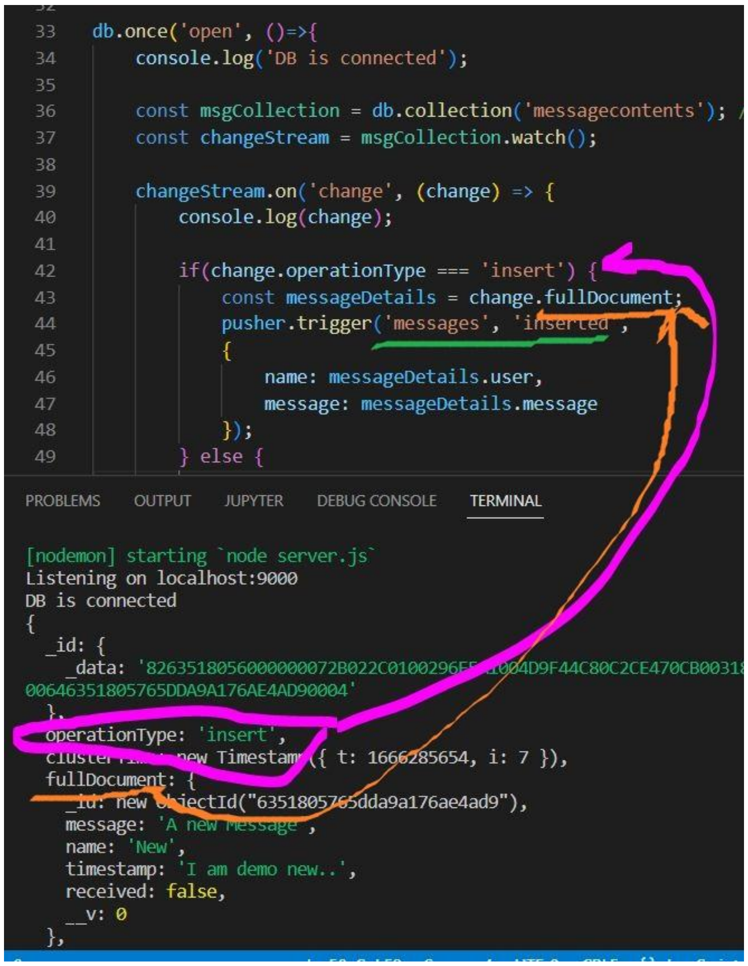
1. Front-end triggeres the back-end. (message is typed. Which is sent to DB and is stored in DB)
2. Backend triggers the Pusher. (DB then sends it to pusher)
3. Pusher then triggeres the frontend. (Pusher has the messages which wishes to show on the front-end)
4. Front-end then reload the data.

Install pusher to VS Code:

→ **npm i pusher**

Add new App config:

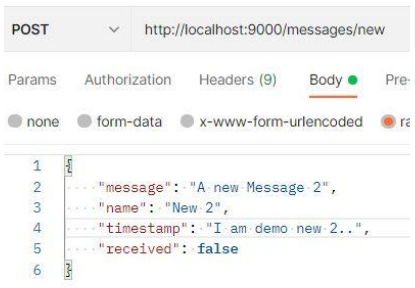
```
const pusher = new Pusher({
  appId: "1494928",
  key: "c6ea993ce75819e1df85",
  secret: "5afb82bd2ca88d6e4613",
  cluster: "ap2",
  useTLS: true
});
```



```
db.once('open', ()=>{
    console.log('DB is connected');

    const msgCollection = db.collection('messagecontents'); /
    const changeStream = msgCollection.watch();

    changeStream.on('change', (change) => {
        console.log(change);

        if(change.operationType === 'insert') {
            const messageDetails = change.fullDocument;
            pusher.trigger('messages', 'inserted',
            {
                name: messageDetails.user,
                message: messageDetails.message
            });
        } else {
```

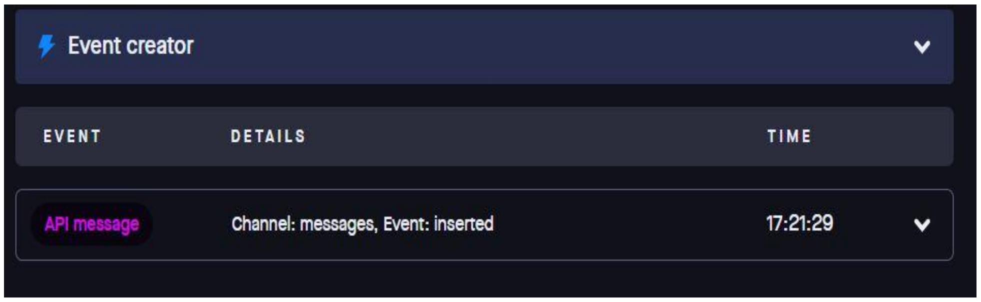
```
[nodemon] starting `node server.js`
Listening on localhost:9000
DB is connected
{
  _id: {
    _data: '82635180560000000072B022C0100296E5F1004D9F44C80C2CE470CB00318
00646351805765DDA9A176AE4AD90004'
  },
  operationType: 'insert',
  clusterTime: new Timestamp({ t: 1666285654, i: 7 }),
  fullDocument: {
    _id: new ObjectId("6351805765dda9a176ae4ad9"),
    message: 'A new Message',
    name: 'New',
    timestamp: 'I am demo new..',
    received: false,
    __v: 0
  },
```

On sending(POST) the messages (sent to DB), it can be seen on pusher (as it gets triggered by the DB)



POST http://localhost:9000/messages/new

```
{
    "message": "A new Message 2",
    "name": "New 2",
    "timestamp": "I am demo new 2..",
    "received": false
}
```



EVENT	DETAILS	TIME
API message	Channel: messages, Event: inserted	17:21:29

How to manage the chat privacy in the user contact?

1. End-to-End encryption *(Most secure nowadays)*
2. Security on Database (lower 1 security)

SECURITY

```
app.use((req, res, next) => {
   res.setHeader("Access-Control-Allow-Origin", "*");
   res.setHeader("Access-Control-Allow-Headers", "*");
   next();
})
```

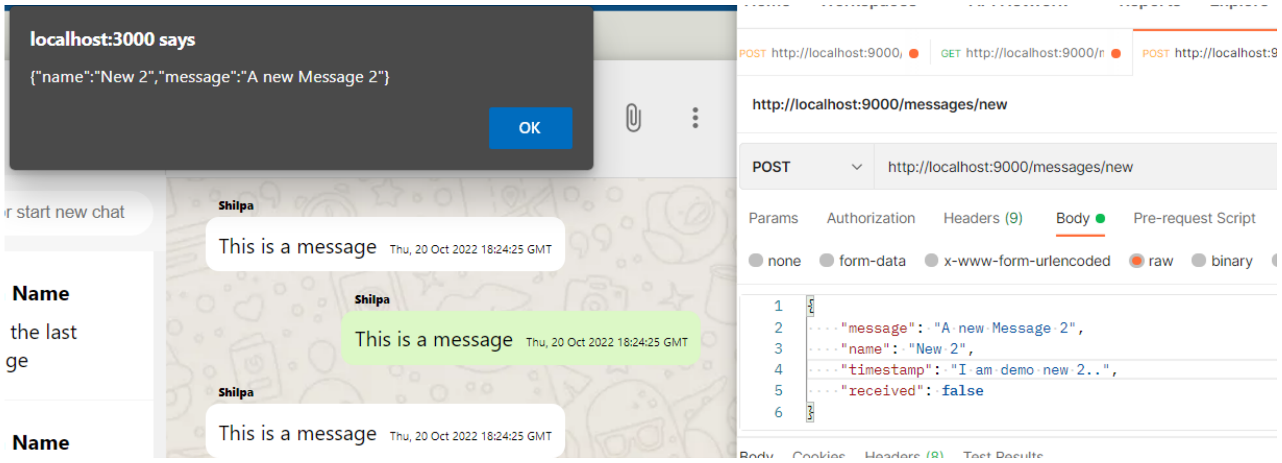
(Alternative)
> npm i cors

app.use(cors()); **→ Use this instead of the above code (Alternative)**

Install npm i pusher-js → in front-end

```
 useEffect(() => {
   Pusher.logToConsole = true;

   const pusher = new Pusher('c6ea993ce75819e1df85', {
     cluster: 'ap2'
   });

   const channel = pusher.subscribe('messages');
   channel.bind('inserted', (data) => {
     alert(JSON.stringify(data));
   });
 }, []);
```

Add the above code to App.js *(got from pusher website)*



localhost:3000 says

```
{"name":"New 2","message":"A new Message 2"}
```

OK
r start new chat
Name
the last
ge
Name
Shilpa
This is a message Thu, 20 Oct 2022 18:24:25 GMT
Shilpa
This is a message Thu, 20 Oct 2022 18:24:25 GMT
Shilpa
This is a message Thu, 20 Oct 2022 18:24:25 GMT
POST http://localhost:9000/
GET http://localhost:9000/r
POST http://localhost:9
http://localhost:9000/messages/new
Params
Authorization
Headers (9)
Body
Pre-request Script
none
form-data
x-www-form-urlencoded
raw
binary

POST http://localhost:9000/messages/new

```
{
    "message": "A new Message 2",
    "name": "New 2",
    "timestamp": "I am demo new 2..",
    "received": false
}
```

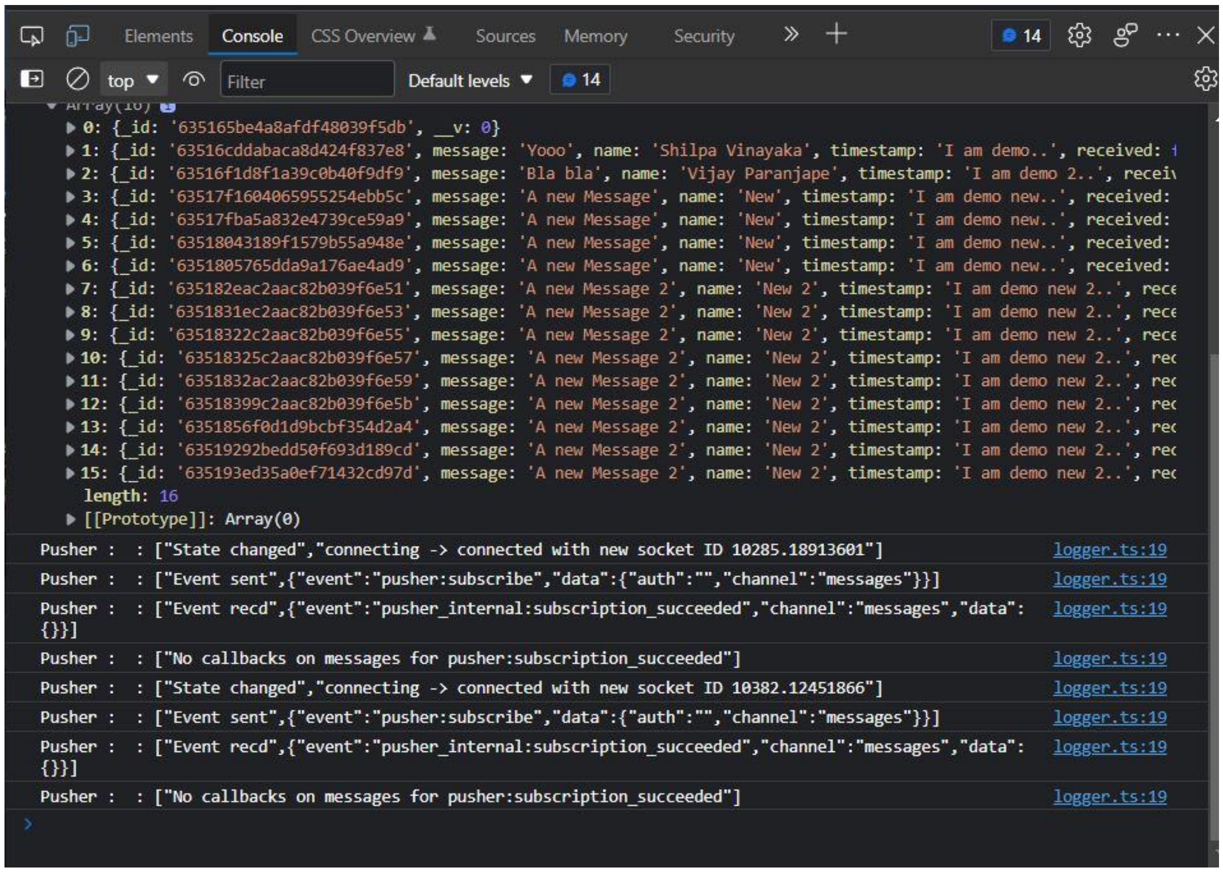
In layman terms: The above prompt means → real-time MongoDB is done.

Install axios in front-end:

→ npm i axios

Axios is responsible for fetching any initial information.

```
 const [messages, setMessages] = useState([]);
 useEffect(() => {

   axios.get('/messages/sync').then((response) =>{
     setMessages(response.data);
   })
 }, []);
```



```
▼ Array(16)
  ▶ 0: {_id: '635165be4a8afdf48039f5db', __v: 0}
  ▶ 1: {_id: '63516cddabaca8d424f837e8', message: 'Yooo', name: 'Shilpa Vinayaka', timestamp: 'I am demo..', received:
  ▶ 2: {_id: '63516f1d8f1a39c0b40f9df9', message: 'Bla bla', name: 'Vijay Paranjape', timestamp: 'I am demo 2..', recei
  ▶ 3: {_id: '63517f1604065955254ebb5c', message: 'A new Message', name: 'New', timestamp: 'I am demo new..', received:
  ▶ 4: {_id: '63517fba5a832e4739ce59a9', message: 'A new Message', name: 'New', timestamp: 'I am demo new..', received:
  ▶ 5: {_id: '63518043189f1579b55a948e', message: 'A new Message', name: 'New', timestamp: 'I am demo new..', received:
  ▶ 6: {_id: '6351805765dda9a176ae4ad9', message: 'A new Message', name: 'New', timestamp: 'I am demo new..', received:
  ▶ 7: {_id: '635182eac2aac82b039f6e51', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rece
  ▶ 8: {_id: '6351831ec2aac82b039f6e53', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rece
  ▶ 9: {_id: '63518322c2aac82b039f6e55', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rece
  ▶ 10: {_id: '63518325c2aac82b039f6e57', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
  ▶ 11: {_id: '6351832ac2aac82b039f6e59', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
  ▶ 12: {_id: '63518399c2aac82b039f6e5b', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
  ▶ 13: {_id: '6351856f0d1d9bcbf354d2a4', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
  ▶ 14: {_id: '63519292bedd50f693d189cd', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
  ▶ 15: {_id: '635193ed35a0ef71432cd97d', message: 'A new Message 2', name: 'New 2', timestamp: 'I am demo new 2..', rec
    length: 16
  ▶ [[Prototype]]: Array(0)
Pusher :  : ["State changed","connecting -> connected with new socket ID 10285.18913601"]                logger.ts:19
Pusher :  : ["Event sent",{"event":"pusher:subscribe","data":{"auth":"","channel":"messages"}}]          logger.ts:19
Pusher :  : ["Event recd",{"event":"pusher_internal:subscription_succeeded","channel":"messages","data":  logger.ts:19
{}}]
Pusher :  : ["No callbacks on messages for pusher:subscription_succeeded"]                                logger.ts:19
Pusher :  : ["State changed","connecting -> connected with new socket ID 10382.12451866"]                logger.ts:19
Pusher :  : ["Event sent",{"event":"pusher:subscribe","data":{"auth":"","channel":"messages"}}]          logger.ts:19
Pusher :  : ["Event recd",{"event":"pusher_internal:subscription_succeeded","channel":"messages","data":  logger.ts:19
{}}]
Pusher :  : ["No callbacks on messages for pusher:subscription_succeeded"]                                logger.ts:19
>
```

```
channel.bind('inserted', (newMessage) => {
    alert(JSON.stringify(newMessage));
    setMessages([...messages, newMessage])
  });
}, [messages]);
```

…messages → keep all the old messages
newMessage → add new message

[messages]); → last line is necessary to refresh for new messages

- **Array.prototype.map() expects a return value from arrow function array-callback-return**

[javascript - How fix this warrning warning Array.prototype.map() expects a return value from arrow function array-callback-return? - Stack Overflow]
When you use {} in an arrow function it creates a code block that expects an explicit return statement
Just change the {} to () and an implicit return occurs.

My current IP address has changed from yesterday.

- **UnhandledPromiseRejectionWarning: MongooseServerSelectionError: Could not connect to any servers in your MongoDB Atlas cluster. One common reason is that you're trying to access the database from an IP that isn't whitelisted. Make sure your current IP address is on your Atlas cluster's IP whitelist:**

Solution: Edit the current IP address and RUN

When I add message to whatsapp, the Number of total documents in MONGODB CHANGESSSS!!!!!!!!!

YAYYYYYY
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!