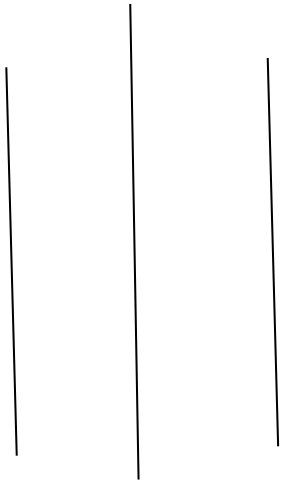
Project
On
Real-Time Chat App



Ashwin Shrestha
KIIT University(2019-2023)
(Personal Project)

1.0 Acknowledgements

I am sincerely thankful to Abhishek Lama Tamang, Rishi KK Shah and Piyush Agarwal of Affiliation who offered continual support and regular encouragement, whose meticulous remarks were a huge assistance to me.

1.1 Abstract:

I have developed a real-time chat app(Dollar L WebApp) where users can register and chat with registered users. It has the features of single-user chat and group chat where only the admin can add users and rename the group chat name. S/he can login as guest user and chat with registered Users. I have used bcrypt + salt for hashing the Password and JWT for authorization and authenticating the user.

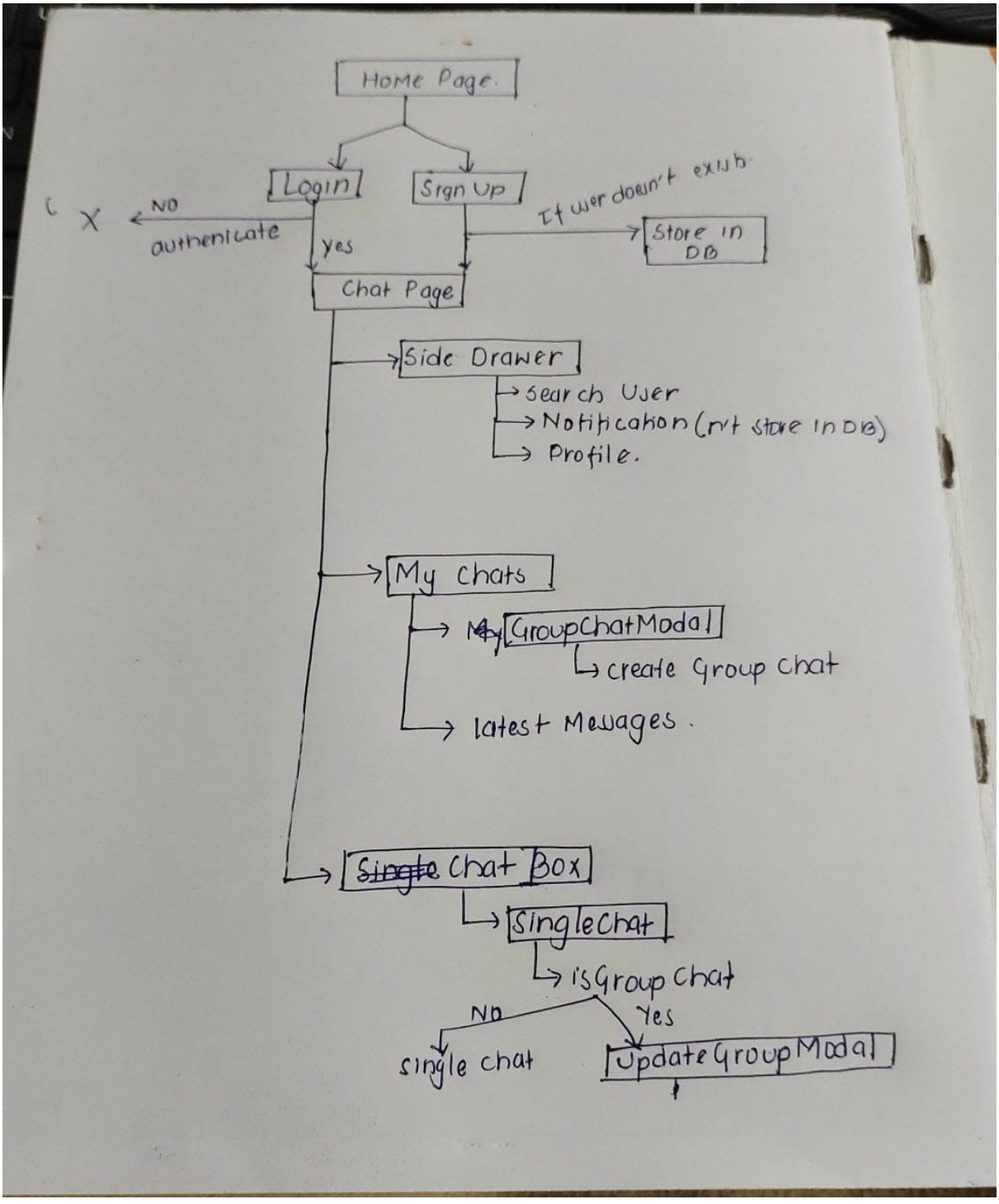
1.2 Technologies Used:

1. Frontend
    - React Js
    - Use Context **(State Management)**
    - Chakra UI
    - Socket.io **(for communication)**
2. Backend
    - Node Js
    - Express Js **(Frame Work of Node Js)**
    - JWT **(For Authorization)**
    - Bcrypt + Salt **(For Hashing the PassWord)**

3. DataBase
    - MongoDB
      Mongoose

4. Platform
    - VS Code
    - MongoDB Atlas(**Cloud Storoage**)

1.3 Basic App Structure

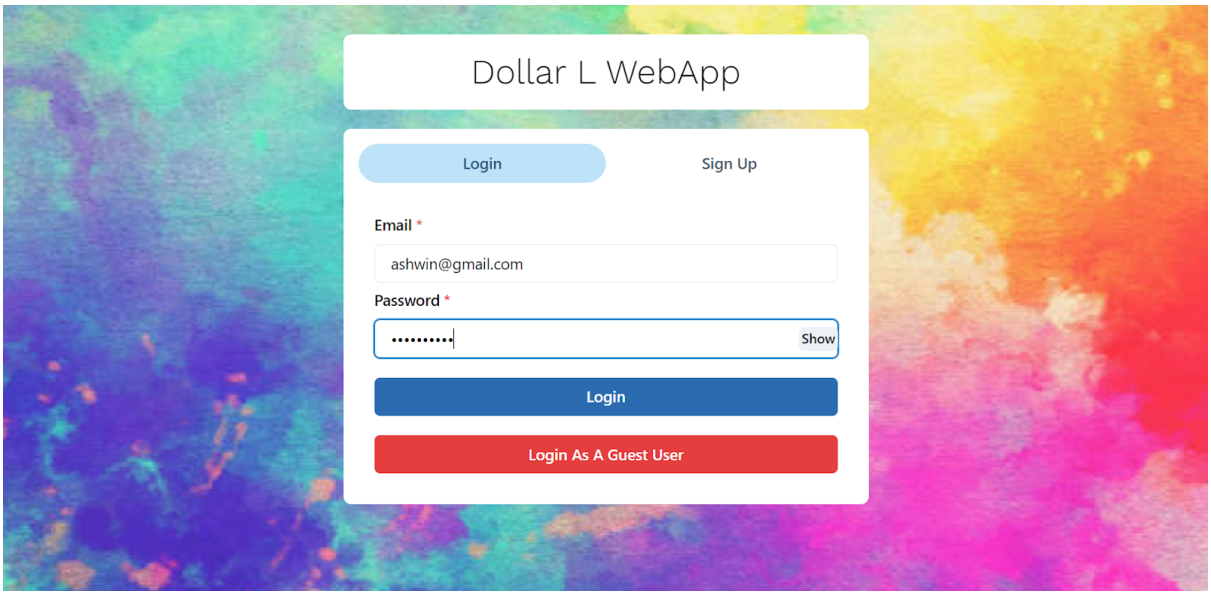


2. Overview of Technologies/Web App

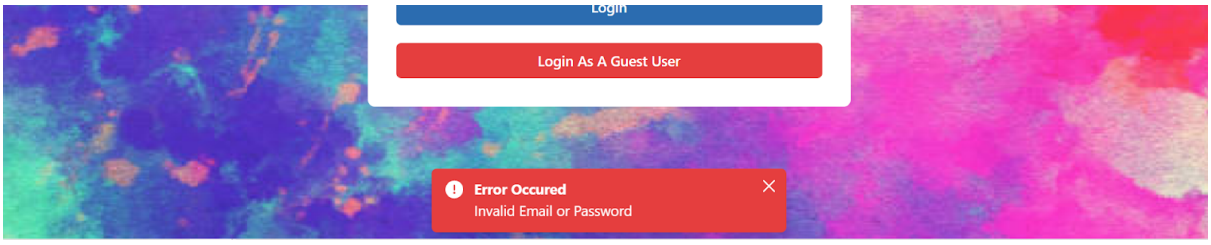
2.1 Frontend

2.1.1 HomePage

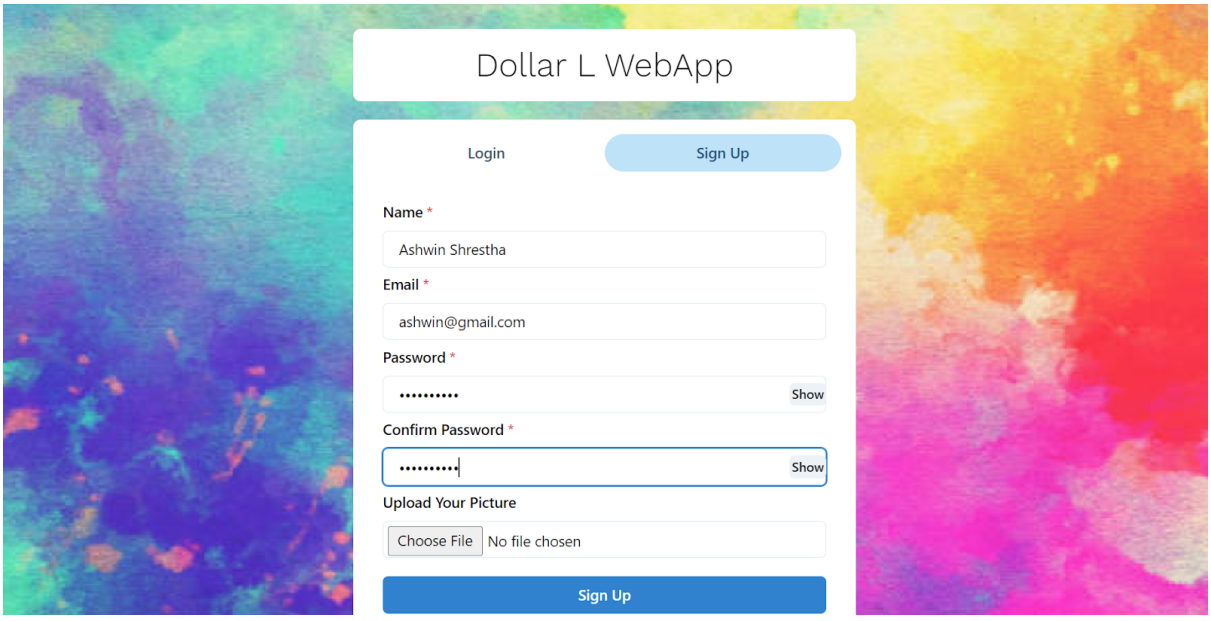
If the user has already registered, s/he can directly log in or s/he can log in as a guest user.



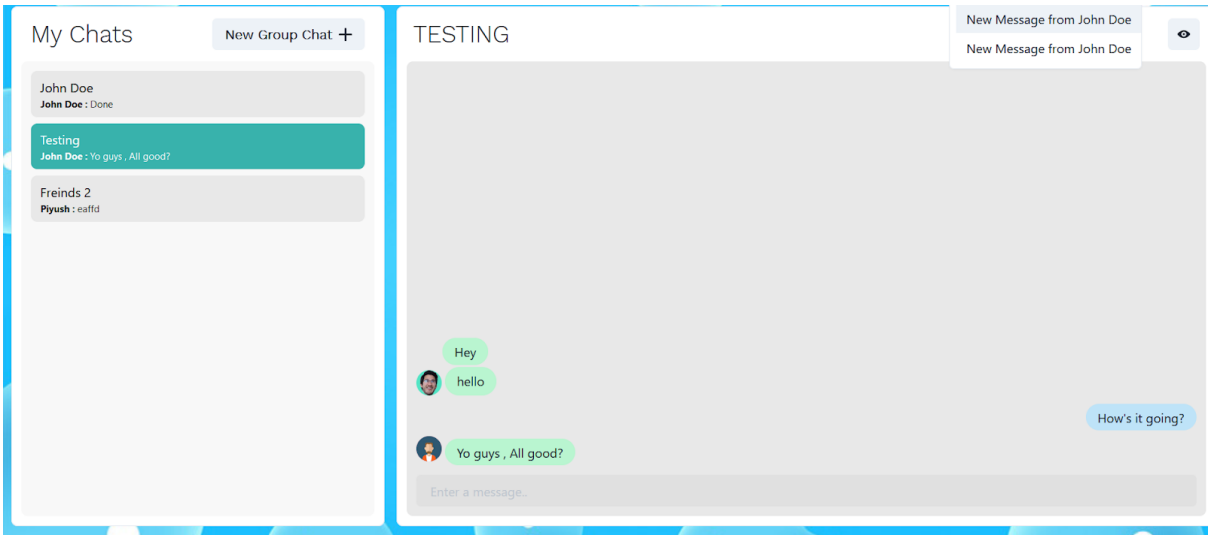
For Invalid Credentials

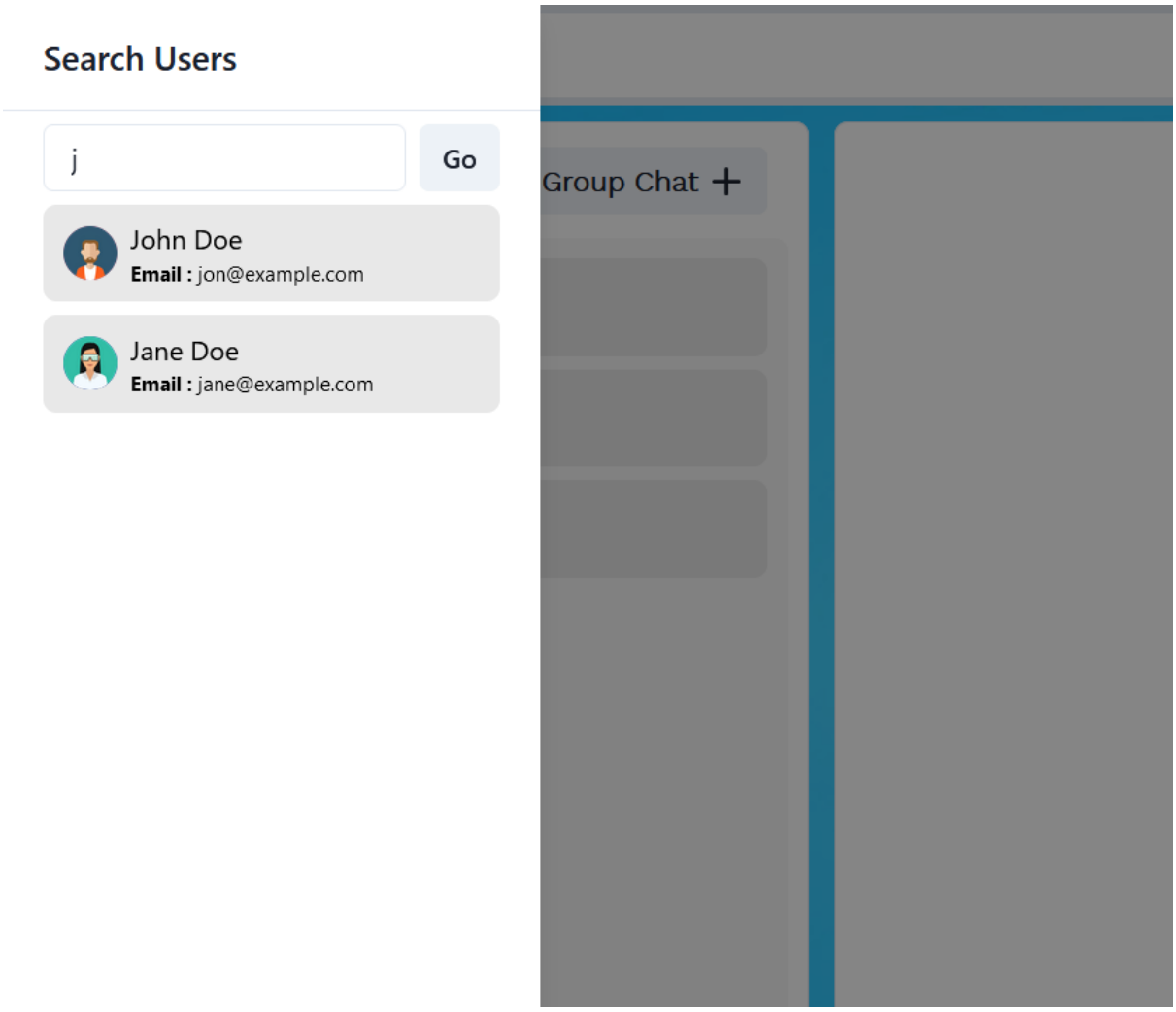


If s/he is not registered, s/he can sign up.



2.1.2 Chat



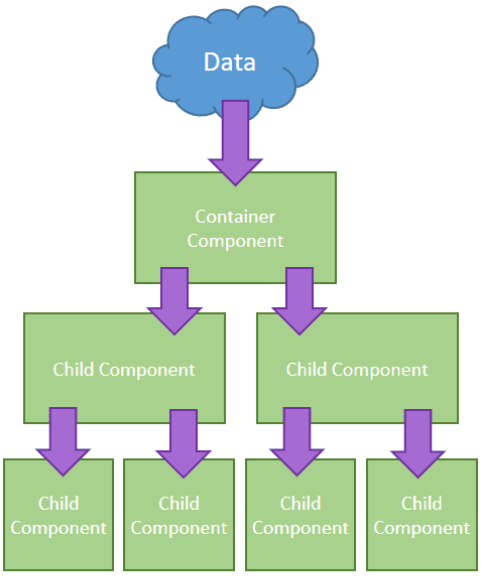


2.1.3 Chakra UI

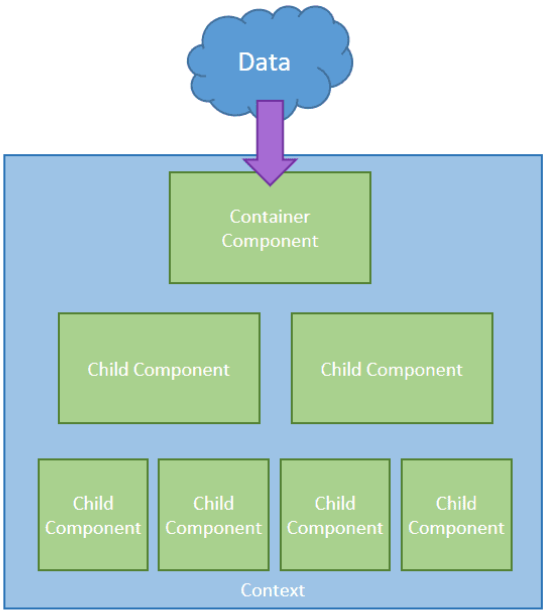
Chakra UI is a component-based library. It's made up of basic building blocks that can help to build the front-end of our web application. It is customizable and reusable, and most importantly it supports ReactJs, along with some other libraries too.

2.1.4 Context

Context is used for State Management

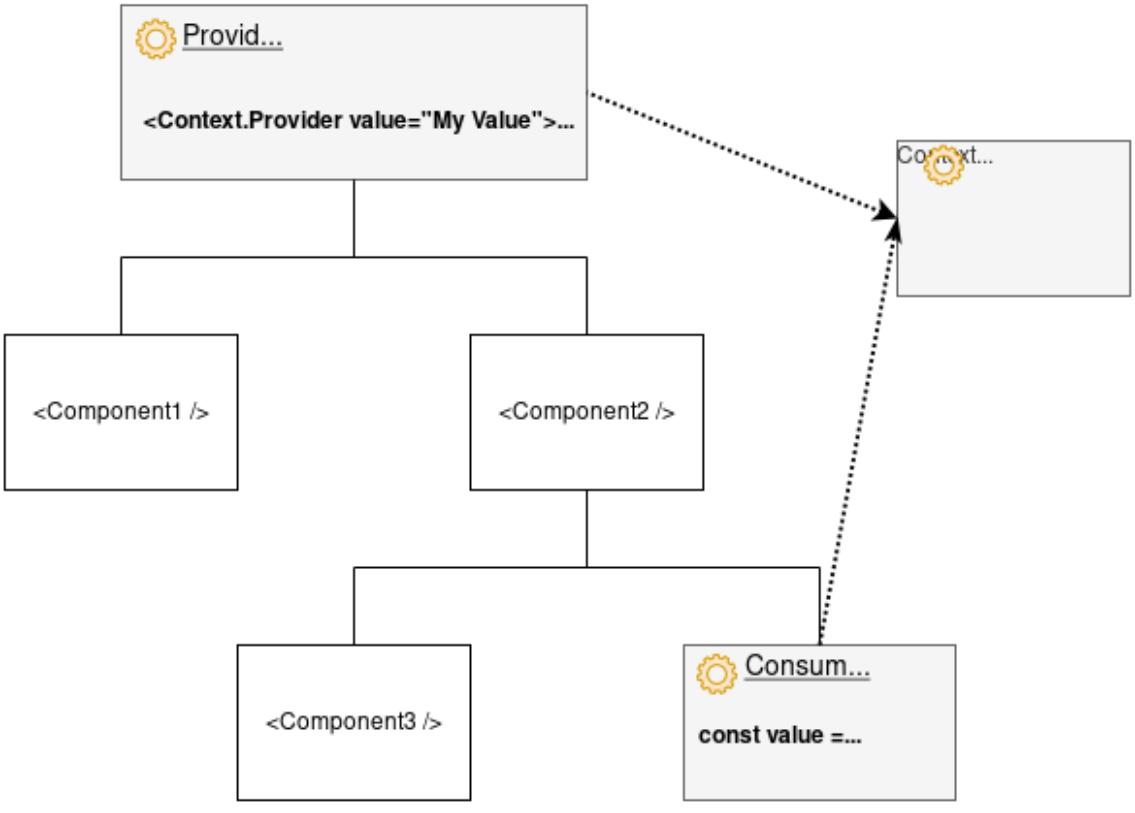


prop drilling



context API

React Context

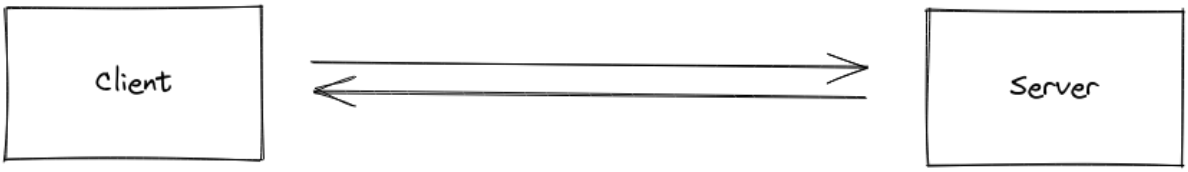


Viewer does not support full SVG 1.1

2.2 Backend

2.2.1 Socket.io

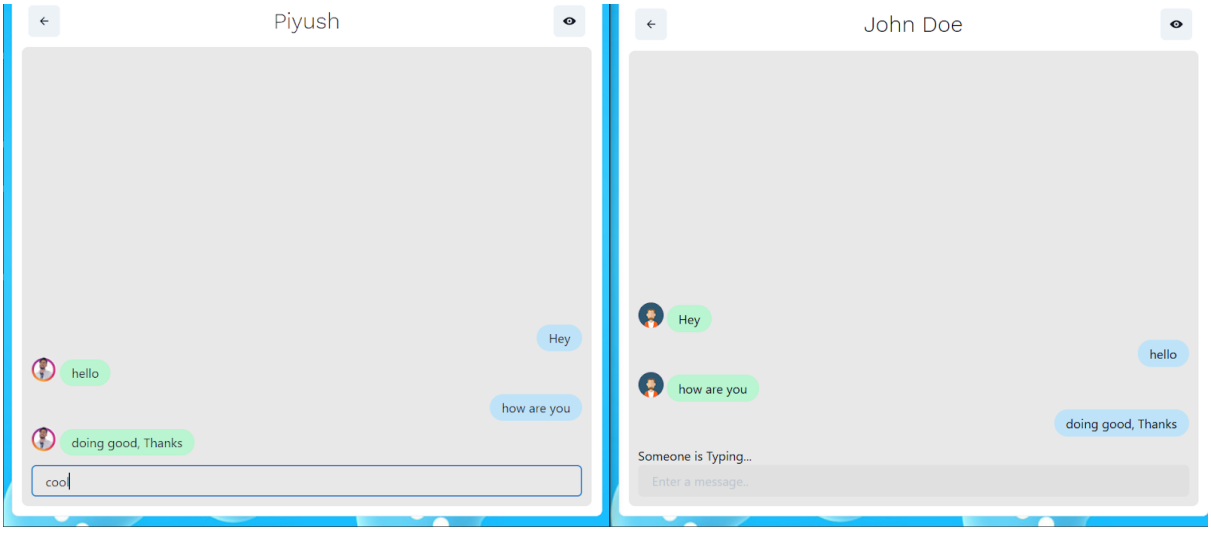
Socket.IO is a library that enables low-latency, bidirectional and event-based communication between a client and a server.
Whenever an event occurs, the idea is that the server will get it and push it to the concerned connected clients.




```
const io = require("socket.io")(server,.....);
io.on("connection", (socket) => {
 console.log("Connected to socket.io");
 socket.on("setup", (userData) => {
   socket.join(userData._id);
   socket.emit("connected");
 });
```

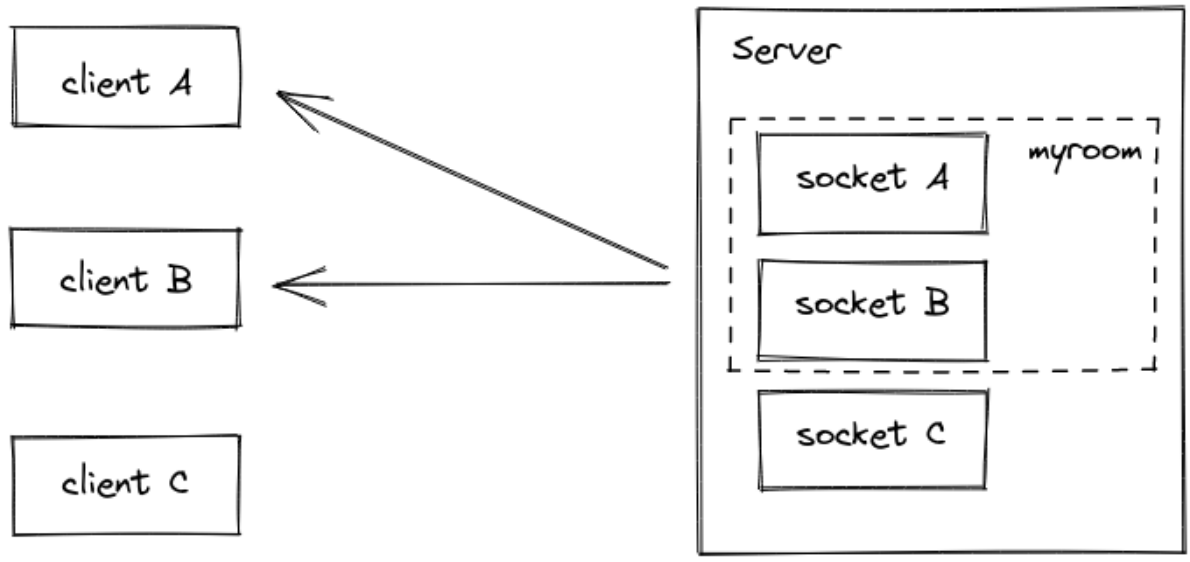
io is a Socket.IO server instance attached to an instance of http.Server listening for incoming events.
The **io.on** event handler handles connection, disconnection, etc., events in it, using the socket object.
socket argument of the connection event listener callback function is an object that represents an incoming socket connection from a client.
Both of them can listen for events with the **on** method.

socket.off—>Unbind the specified event handler (opposite of `.on()`).



We can create and fire custom events using the **socket.emit** function.
To handle this custom event on client we need a listener that listens for the event" **connected"**.

A *room* is an arbitrary channel that sockets can join and leave. It can be used to broadcast events to a subset of clients:



We can call join to subscribe the socket to a given channel:

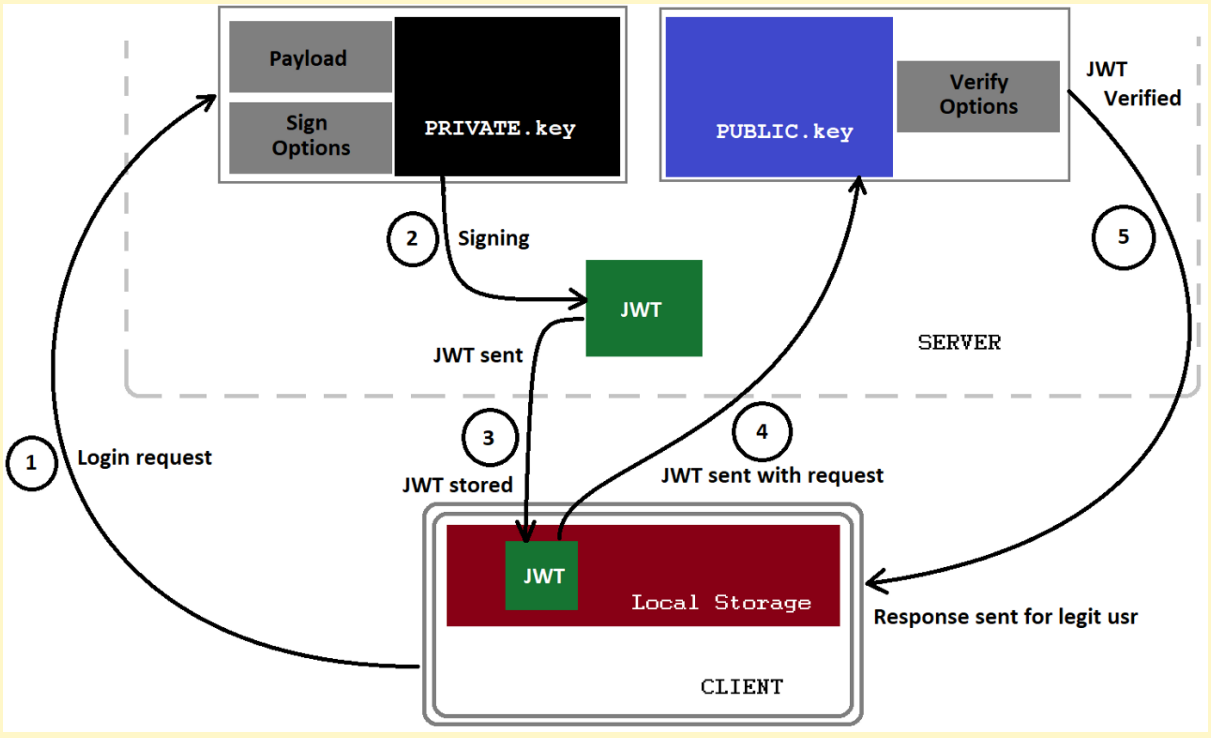
```
io.on("connection", socket => {
 socket.join("some room");
});
```

And then simply use to or in (they are the same) when broadcasting or emitting:

```
io.to("some room").emit("some event");
```

2.2.2 JWT

JWT, or JSON Web Token, is an open standard used to share security information between two parties — a client and a server.
JWTs are credentials, which can grant access to resources. (authorization)
“We do not record tokens, all validation and debugging is done on the client side.”



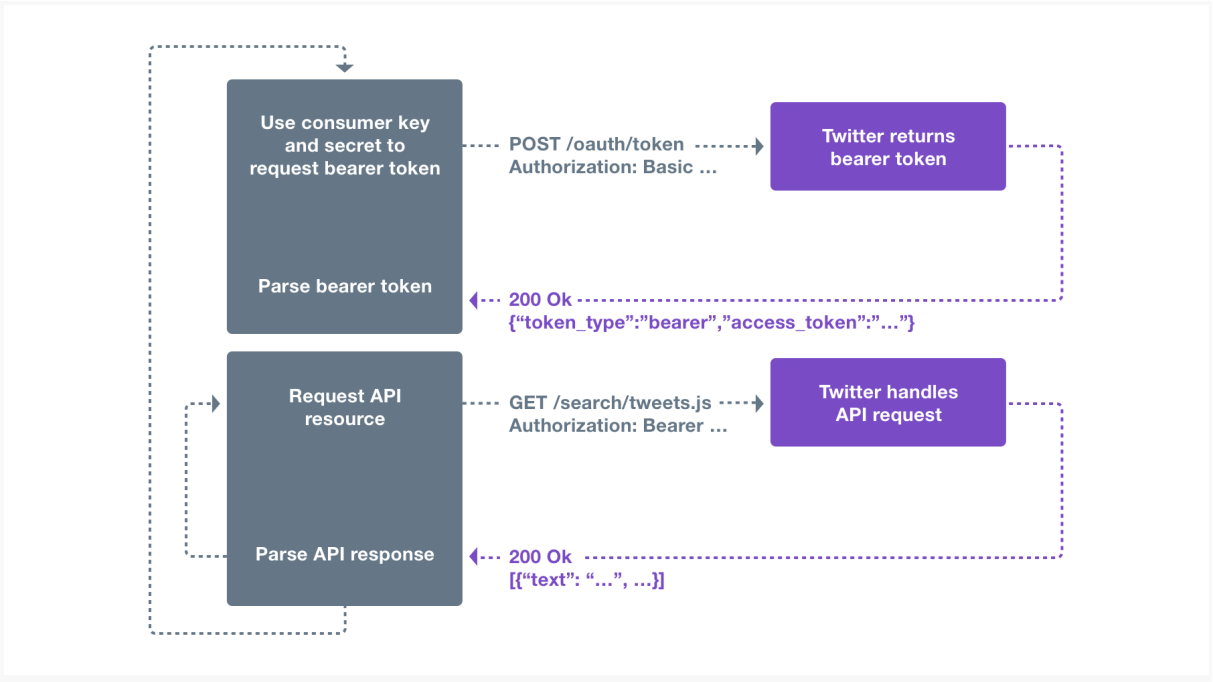
a. Generating Token

```
const generateToken = (id) => {
 return jwt.sign({ id }, "ashwin", {
   expiresIn: "30d",
 });
 //I goona sign new Token with that paticulary unique id(send from userController.js)
 //and also we have to have jwt secrect
 //and third is in how much time this Token Expires
};
```

b.Verifying Token

```
//token looks Like this---> Bearer ajhfjfjhdskhd  (from here we remove the Bearer and just take token)
    —> token = req.headers.authorization.split(" ")[1];

     //decodes token id
    //JWT verify method ---> is used for verify the token the take two arguments one is token string value,
    //and second one is secret key for matching the token is valid or not.
    //The validation method ---> returns a decode object that we stored the token in
   —>  const decoded = jwt.verify(token, "ashwin");
```



From Client:

```
const config = {
    headers: {
      Authorization: `Bearer ${user.token}`,
    },
  };
  const { data } = await axios.get(`/api/user?search=${search}`, config);
```

2.2.3 Express Js

Express is a free and open source web application framework for Node.js. It provides a simple routing for requests made by clients. It also provides a middleware that is responsible for making decisions to give the correct response for the clients' request.
eg.

```
router.route("/").post(protect, accessChat);
```

Protect , accessChat—>> authenticating the user, and authorizing the resources to the user.

2.2.4 Bcrypt + Salt(High Secure)

Bcrypt is a library of Node Js and used for password hashing function.

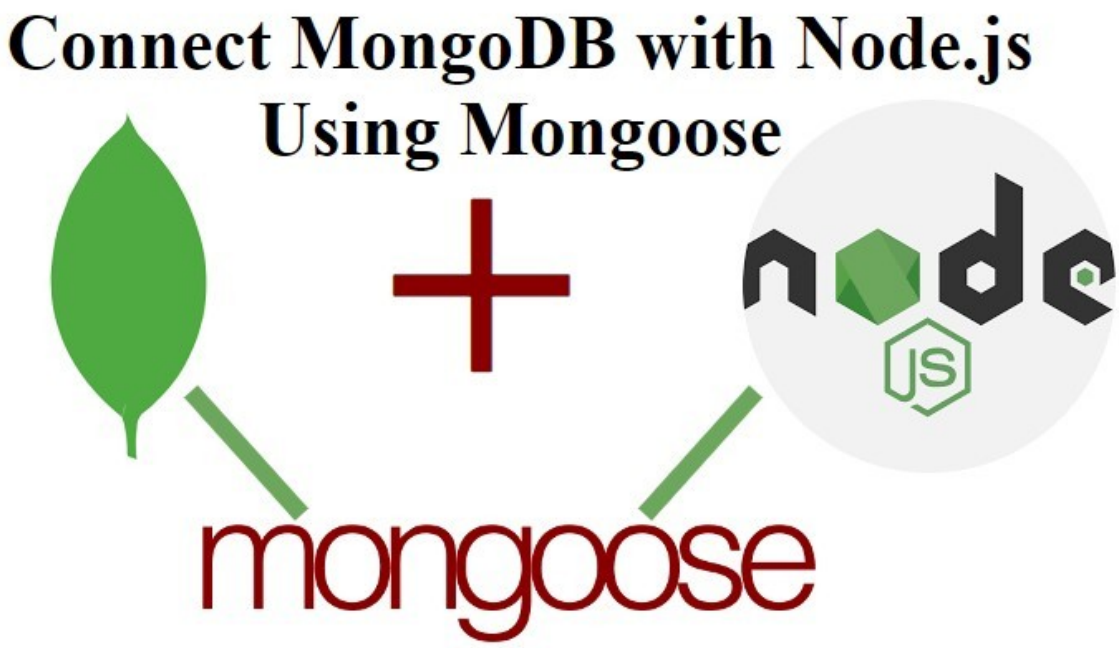
```
const salt = await bcrypt.genSalt(10); //higher the number , the strong salt will be generated
this.password = await bcrypt.hash(this.password, salt); //password + salt is hashed
```

Bcrypt is designed to be slow so that creating this dictionary(dictionary attack) will take very very long time(highly secure).

2.3 Database

2.3.1 MongoDB

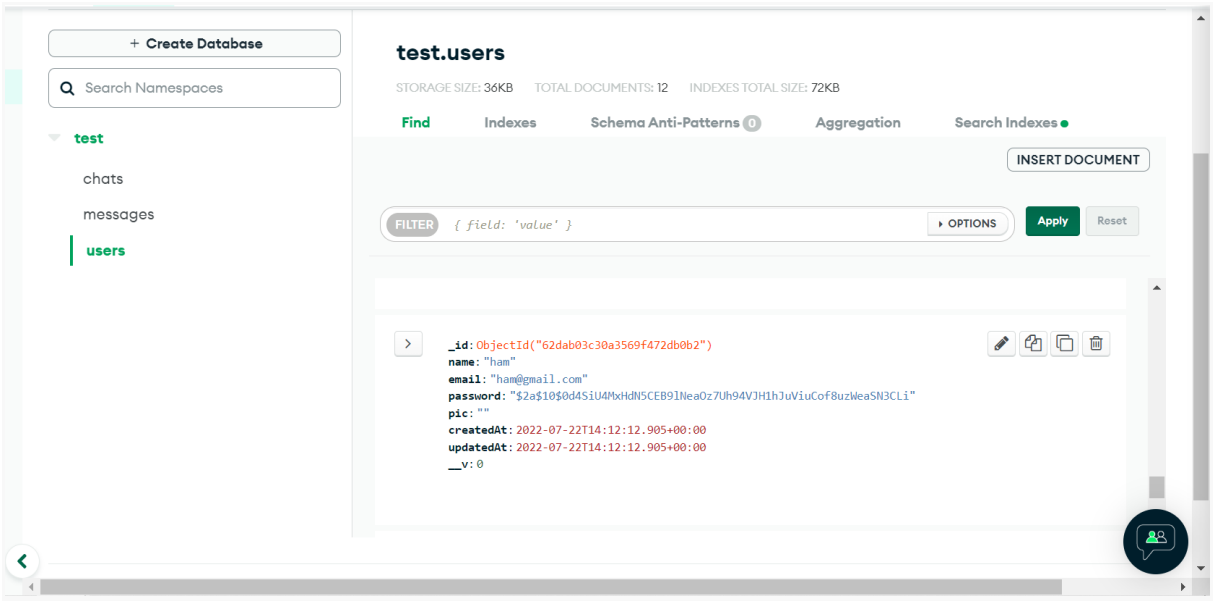
MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.



Mongoose is a MongoDB ODM i.e (Object database Modelling) that used to translate the code and its representation from MongoDB to the Node.js server.
Using mongoose, a user can define the schema for the documents in a particular collection. It provides a lot of convenience in the creation and management of data in MongoDB.

```
const conn = await mongoose.connect(
    "URL",
    {
      useNewUrlParser: true,//allow users to fall back to the old parser if they find a bug in the new parser
      useUnifiedTopology: true,//Set to true to opt in to using the MongoDB driver's new connection management engine.
    }
  );
```

DataBase



References:

1. Official Documentations, Wikipedia
    a. Node js(https://nodejs.org/en/docs/)
    b. Express js (https://expressjs.com/)
    c. React Js (https://beta.reactjs.org/)
    d. Chakra UI (https://chakra-ui.com/)
    e. Axios (https://axios-http.com/docs/intr)
    f. JWT (https://jwt.io/introduction)
    g. Bcrypt + Salt(https://en.wikipedia.org/wiki/Bcrypt)
    h. MongoDB (https://www.mongodb.com/docs/)
    i. Mongoose (https://mongoosejs.com/)
    j. Socket.io (https://socket.io/)

2. Youtube
    a. https://www.youtube.com/