create folder

Git add .

Git commit -m “change comment”

Git push origin main/master

npm init

npm i express

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ creating nodeJS backend server\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create folder "backend" with file "server.js" :

const express=require("express");

const app=express();

app.listen(5000,console.log("server started on port 5000"));

to run-> node "location of file"

alternative-> add "start":"node backend/server.js" to scripts of package.json to just type (npm start) to run

app.get("/",(req,res)=>{

res.send("API is running")

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Outputing dummy data \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create "data.js" file in "data" folder inside backend folder

export data constant

//access all chats

app.get("/chat",(req,res)=>{

res.send(chats);

});

//access single chat ID put in the search parrameter (http://localhost:5000/chat/617a077e18c25468bc7c4dd4)

app.get("/chat/:id",(req,res)=>{

console.log(req.params.id);

});

//Access single chat details using ID

app.get("/chat/:id", (req, res) => {

const chatId = req.params.id;

const chat = chats.find((chat) => chat.\_id === chatId);

if (chat) {

res.send(chat);

} else {

res.status(404).send({ message: "Chat not found" });

}

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ using dotenv file to not make the port number public \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npm i dotenv

const dotenv = require("dotenv");

dotenv.config();

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log("Server started on port", PORT));

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ NODEMON TO NPM START EVERYTIME \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npm i nodemon

change node to nodemon in package.json scripts section

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CREATING FROTEND USING REACT AND STARTING UP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npx create-react-app frontend

remove unnecessary files like

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Setting up Chakra UI \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

see docs for setup and integration

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Connecting the frontedn to backend \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

frontend is on port 3000 while backend on 5000

adding a proxy to the package.json

"proxy":"http://127.0.0.1:5000"

now npm start both backend and frontend

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Creating multiple pages using react router dom \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npm i react-router-dom

this is how index.js looks like after wrapping chakra UI and router DOM

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

<React.StrictMode>

<BrowserRouter>

<ChakraProvider>

<App />

</ChakraProvider>

</BrowserRouter>

</React.StrictMode>

);

// add routes for different pages in App.js

<div className="App">

<Route path="/" component={Homepage} exact/>

<Route path="/chats" component={ChatPage} exact/>

</div>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ API call from out backend to frontend (in ChatPage.js)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npm i axios

//get the data of all users

return (

<div>

{chats.map((chat) => (

<div key={chat.\_id}>{chat.chatName}</div>

))}

</div>

);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Creating database schema in mongoose \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

npm i mongoose

create "chatModel.js" inside "models" folder inside backend folder

similarly create the userModel and messageModel

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Creating the frontend (Homepage.js)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

background image

box and tabs of chakra UI to create the loging and signup page

create Login.js and Signup.js inside "authentication" folder inside "components" folder inside the src folder of frontend

and export them to the Homepage (two tabs login and signup)

//vstack (chakra UI) for signup and login page

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Setting up MONGO DB atlas database \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

https://youtu.be/mnHjdgdCK6c?list=PLKhlp2qtUcSZsGkxAdgnPcHioRr-4guZf

create database in mongo db atlas

copy the connection string and put in the .env file

create "db.js" inside "config" folder to connect with database

optional-using npm i colors to beautify the error or log messages

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ user authentication \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create "userRoutes.js" inside "routes" folder in backend

import it in server.js

define route for registration and login with thier logic in controllers folder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Logic for registration \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create "userController.js" inside "controllers" folder in backend

npm i express-async-handler (to handle all errors automatically)

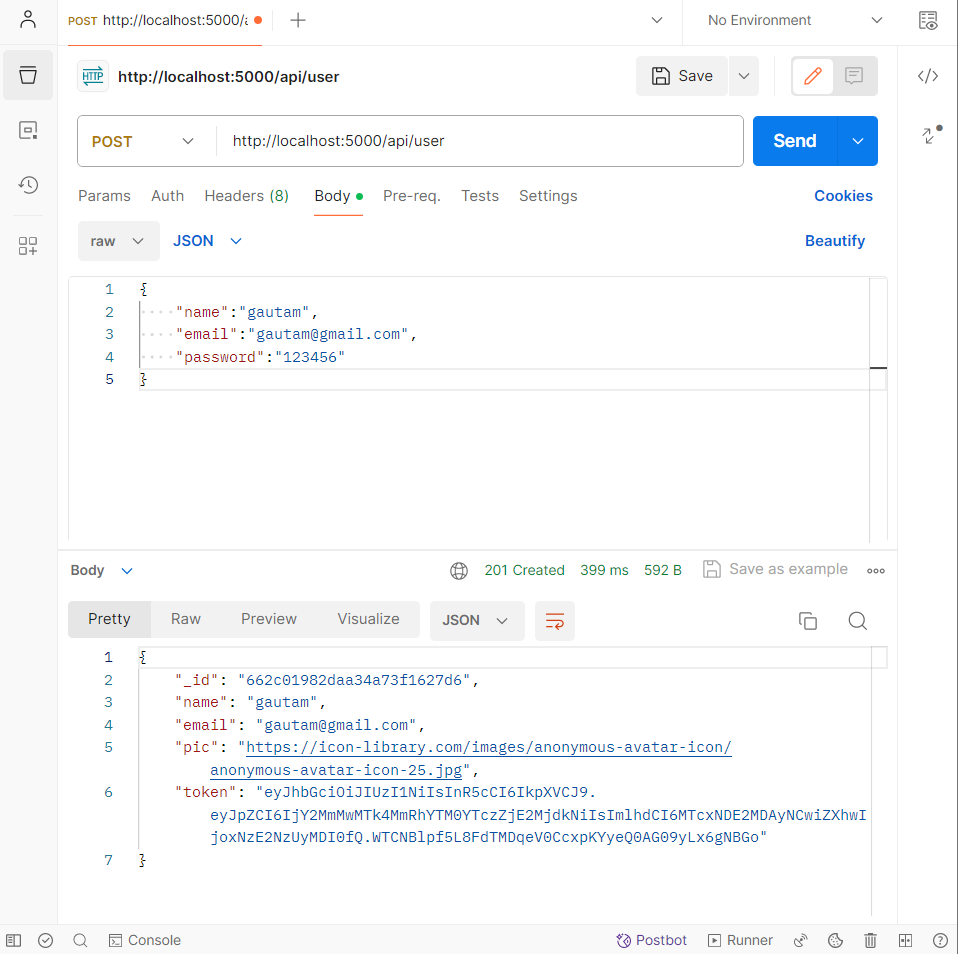
logic to check if user already exists else create a new user

also create and send a JWT token with the new user (create "generateToken.js" in config)

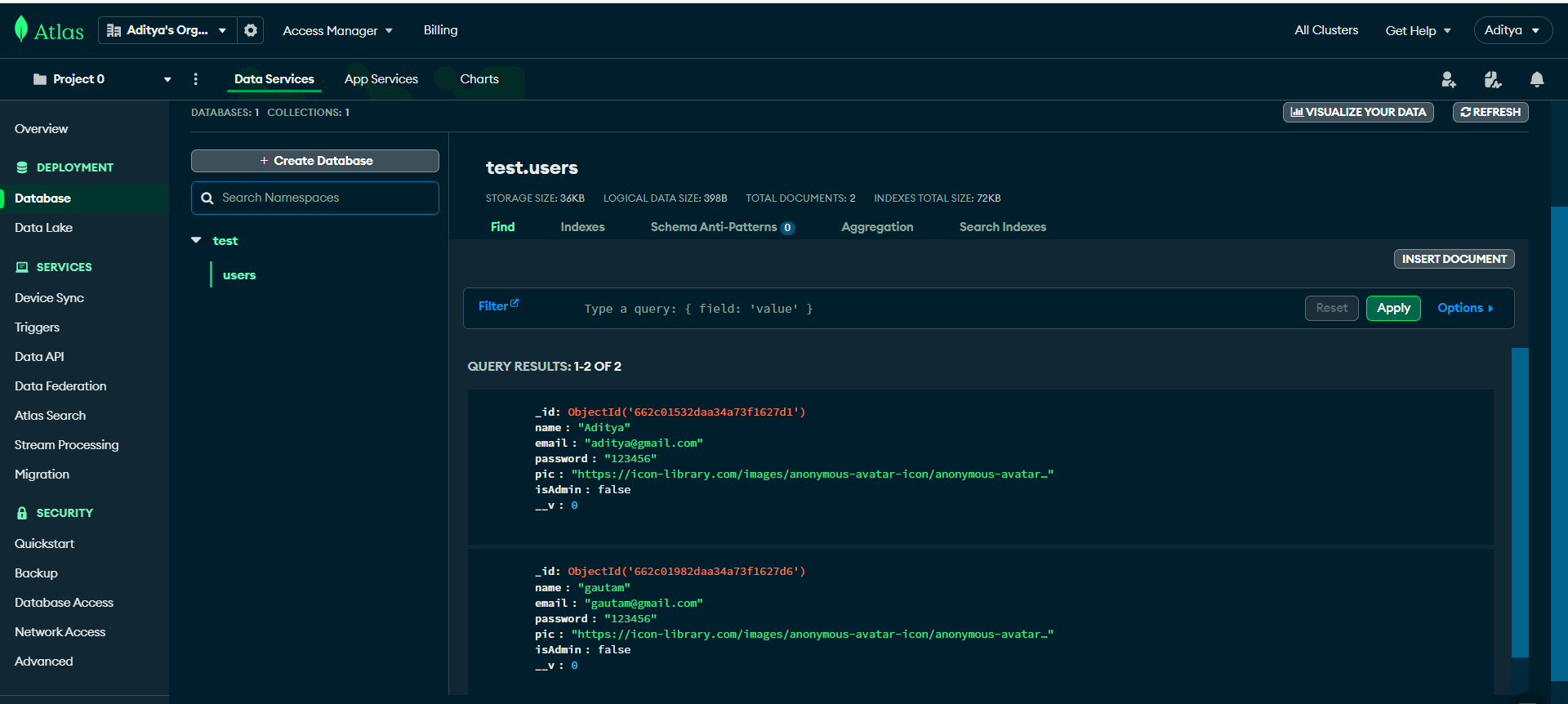
app.use(express.json()); -> to accept user input (in server.js)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Checking the API/JWT token using postman \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The jwt token is getting created with the new user



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Checking the API/JWT in Database\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ encrypting and storing the password (userModel.js) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Adding salt to the password

//before savoing the password,run this function (.pre())

userSchema.pre("save", async function (next) {

  //if alreadt modified/salt added dont do anything ie-next()

  if (!this.isModified) {

    next();

  }

  //else add salt to password

  const salt = await bcrypt.genSalt(10);

  this.password = await bcrypt.hash(this.password, salt);

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Login Functionality \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Router.post(‘/login’,authUseer) -> Add router in userRoutes.js

----------------(match password function in userModel.js)-------------

//method to match the password

userSchema.methods.matchPassword = async function (enteredPassword) {

  return await bcrypt.compare(enteredPassword, this.password);

};

authUser function in usercontroller.js (to match the entered and existing email password for login)

* When we create a new user through postman we can see in the MongoDB server that the password is being encrypted

A computer screen shot of a computer code

Description automatically generated

* A screenshot of a computer

  Description automatically generatedGiving email and password to login api,getting all details of user

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Error handling if user goes to any wrong route\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(in server.js)

app.use(notFound);

app.use(errorHandler);

* Create “middleware” folder in backend with files “errorMiddleware.js” (create the notFound and errorHandler functions in this )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SIGNUP FUNCTIONALITY ON FRONTEND\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Setup Cloudinary account to store image in it ,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ LOGIN FUNCTIONALITY ON FRONTEND\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROBLEM DIAGNOSIS

* check you are going to localhost:3000 and not localhost:300/chats when chats page is not created
* inspect -> applpication -> clear local storage

################ THE FOLLOWING IS THE BACKEND FOR THE CHATS PAGE ###########################

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ API and functionality for searching the other users \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ get all users functionality \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* add .get(allUsers) route to "/"
* now create this allUsers function inside userControllers.js
* the following function check if there is any query,then return all the users whose email or name contains “a”,”adi” or ”any parameter passed”

const allUsers = asyncHandler(async (req, res) => {

const keyword = req.query.search

? {

$or: [

{ name: { $regex: req.query.search, $options: "i" } },

{ email: { $regex: req.query.search, $options: "i" } },

],

}

: {};

const users = await User.find(keyword);

res.send(users);

});

* the following returns all the users except the current user who is logged in

const allUsers = asyncHandler(async (req, res) => {

..

..

..

const users = await User.find(keyword).find({ \_id: { $ne: req.user.\_id } });

res.send(users);

});

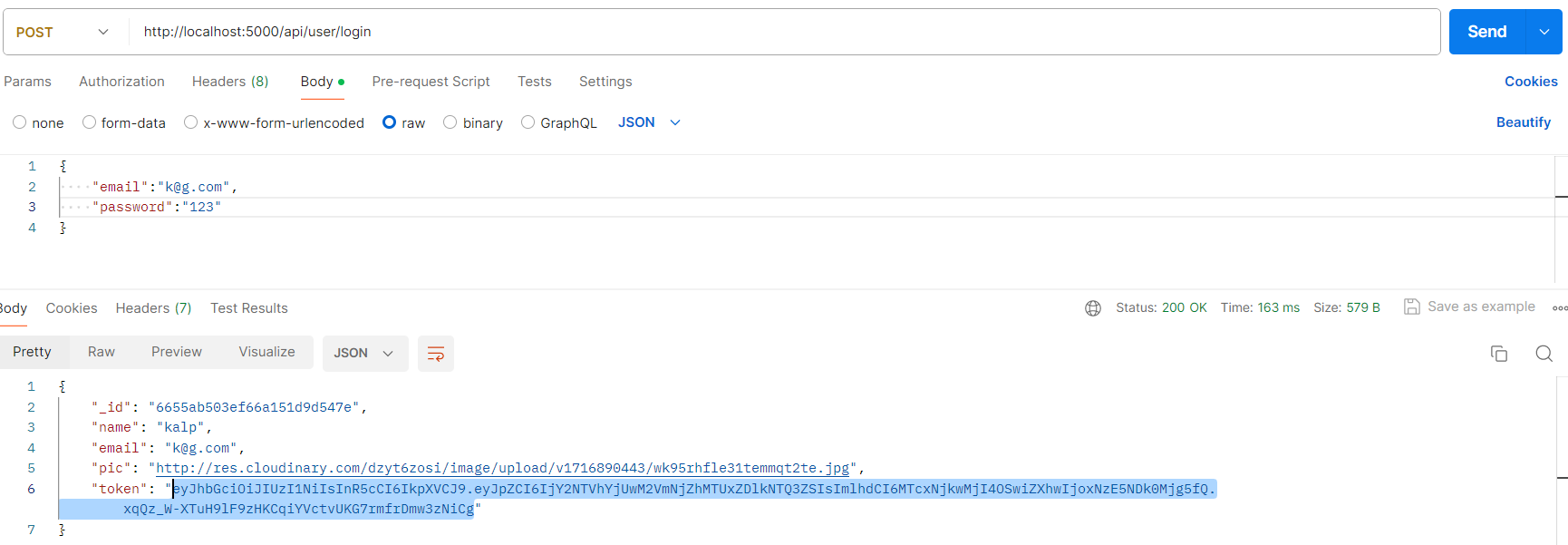
BUT TO USE THIS WE NEED TO CREATE AUTHORIZE THE UER FIRST AND CHECK IF HE IS LOGGED IN OR NOT

THEREFORE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CREATING AUTHORIZATION MIDDLEWARE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* create “authMiddleware.js” inside middleware folder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ GETTING USER DETAILS AFTER AUTHORIZING USING BEARER TOKEN \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* getting bearer token by logging in postman
* A screenshot of a computer

  Description automatically generatedGetting all users except itself when searching (select and add bearer token in authorization)
* THE PROTECT MIDDLEWARE DOESN’T ALLOW USERS TO ACCESS THE CHAT PAGE IF ITS NOT LOGGED IN

(DOESN’T LET THE USER ACCESS THE END POINT)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ NEW API END POINT FOR CHAT PAGE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* app.use("api/chat", chatRoutes);
* create “chatRoutes.js” folder in routes folder
* CREATING ALL THE API END POINTS HERE

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

router.route("/").get(protect, fetchChats);

router.route("/group").post(protect, createGroupChat);

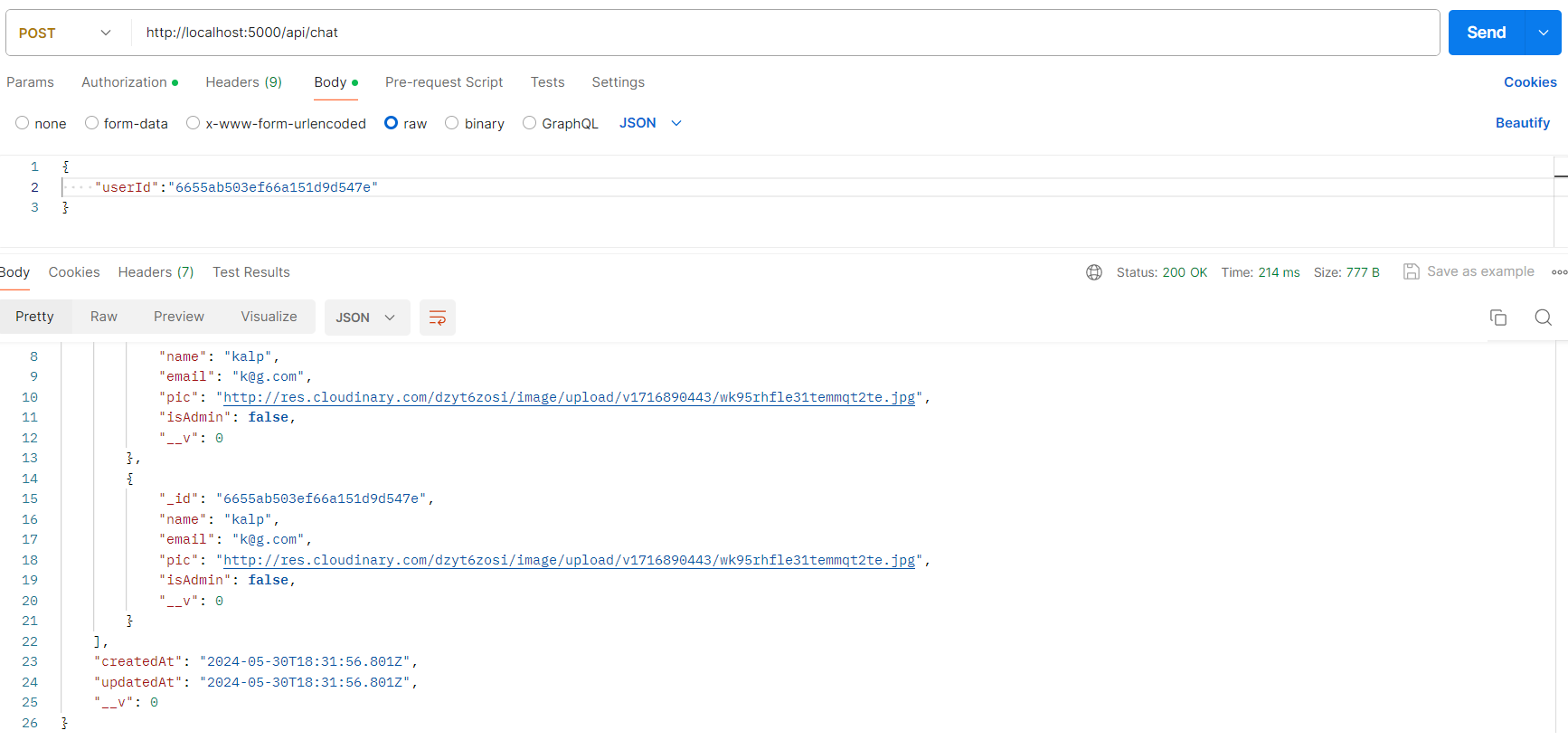
router.route("/rename").put(protect, renameGroup);

router.route("/groupremove").put(protect, removeFromGroup);

router.route("/groupadd").put(protect, addToGroup);

* CREATE “chatControllers.js” IN controllers FOLDER TO IMPLEMENT ALL THESE FUNCTIONS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ accessChat function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Function for creating or fetching one on one chat
* Check codes with comments for explanation.
* Testing the creation of a new chat (provide bearer token in authorization and user ID to create chat)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ fetchChat function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* A screenshot of a computer

  Description automatically generatedReturn all the chats that the user is a part of

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ create group function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A screenshot of a computer

Description automatically generated🡪add bearer token and the following raw in json (user and strigified form of users ID’s in the group)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ rename group function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A screenshot of a computer

Description automatically generatedsend bearer token with (chatId of the group and new chat namem to rename the group with)

################ THE FOLLOWING IS THE FRONTEND FOR THE CHATS PAGE ###########################

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ context api \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Helps to manage state of the project
* Context API provides a means to share values like state, functions, or any data across the component tree without passing props down manually at every level. This is particularly useful for global data that many components need to access.
* Create “chatProvider.js” inside new “Context” folder inside src
* WRAP AROUND THE WHOLE APP IN INDEX.JS USING   <ChatProvider> </ChatProvider>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CHAT PAGE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_