

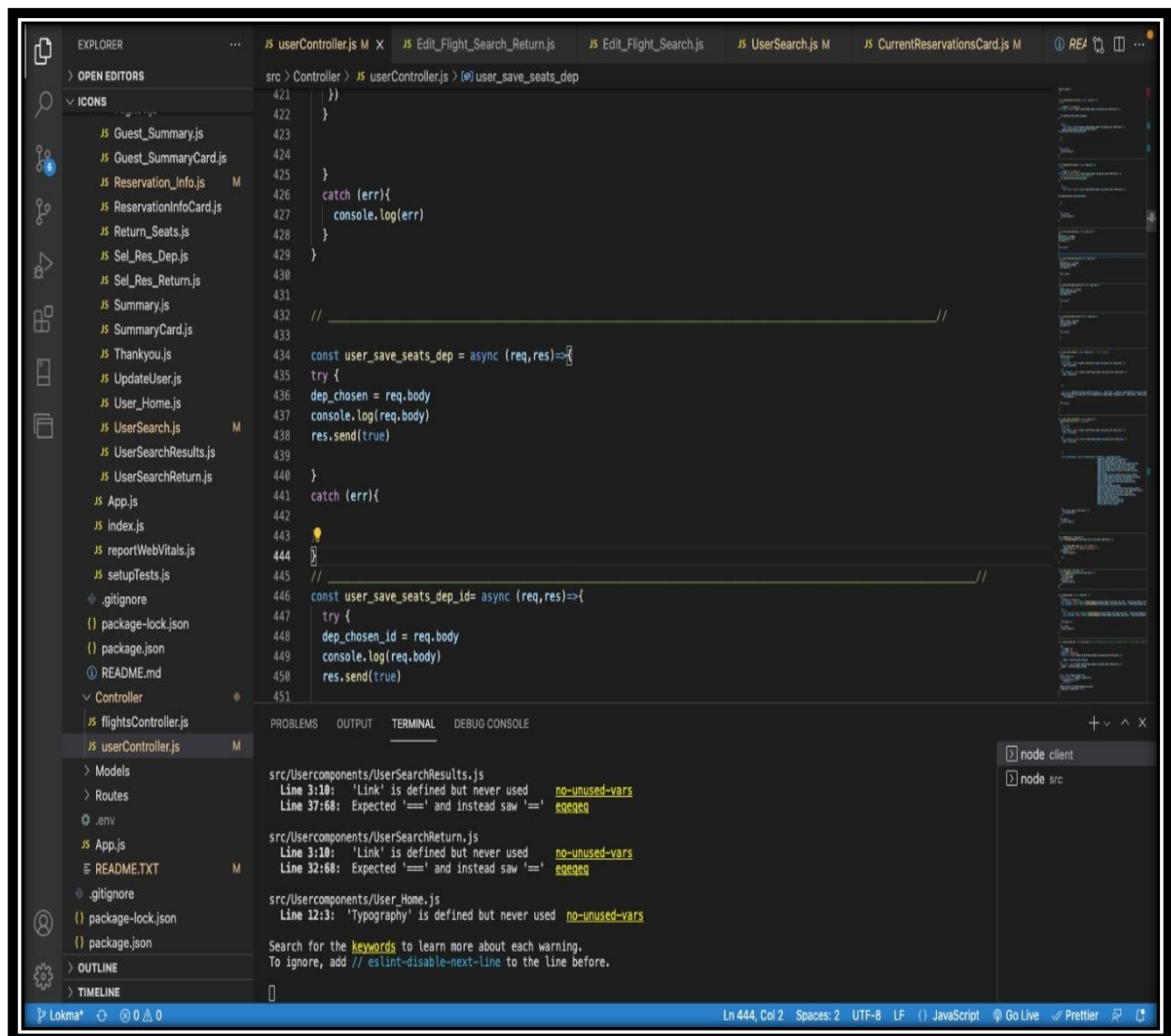
Sky Reserve

Motivation:

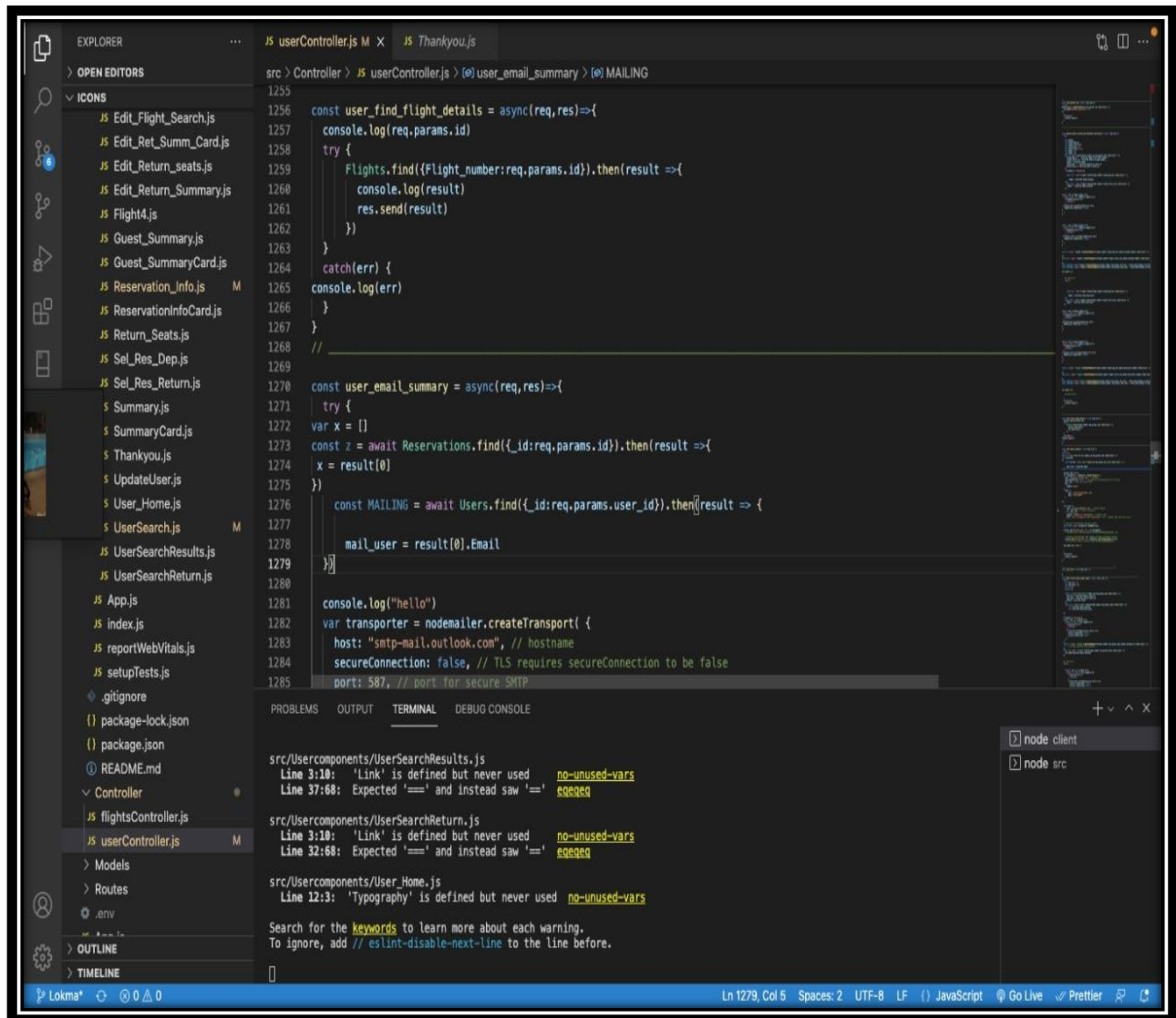
Airline reservation system is an online web-site that facilitates finding flight data from all airlines. It is an exceptionally advantageous system, especially when it comes to the restrictions of the pandemic. Travelling down to airports for reservations is currently challenging, whilst reserving tickets from your current seat at home doesn't squander your time and energy. One of the most substantial merits of the web-site, is its capability to find all the available seats to give you the extravagance of choosing what is most suitable for you. Finding convenient flight fares is indispensable, therefore the web-site is providing various different prices to allow you to choose what suits you the most.

Code style

- Regarding the use of **local** and **global** variables, global variables are not frequently present in the project, as they serve a specific role. For instance, this **global variable** which is used in saving the **departure seats** of the user, however if the user decided to cancel before completing reservation the seats are not saved in the database. They would **only be saved** in the **global variable**.



- Regarding the naming convention of the local variables which are visible here in this example, this variable is called **mail_user** which indicates that it is the specified mail of the user we want to send to.



The screenshot shows the Visual Studio Code editor with a project structure on the left. The Explorer view shows a file tree with folders like 'Controller', 'Models', and 'Routes', and various JavaScript files. The main editor displays the code in 'userController.js', specifically the 'MAILING' section. The code defines two asynchronous functions: 'user_find_flight_details' and 'user_email_summary'. The 'user_email_summary' function uses 'await Users.find' to retrieve user information and assigns the email to a variable named 'mail_user'. The 'PROBLEMS' panel at the bottom shows several ESLint warnings, including 'no-unused-vars' for variables like 'Link', 'Typography', and 'Link'.

```
src > Controller > JS userController.js > [0] user_email_summary > [0] MAILING
1255
1256 const user_find_flight_details = async(req,res)=>{
1257   console.log(req.params.id)
1258   try {
1259     Flights.find({Flight_number:req.params.id}).then(result =>{
1260       console.log(result)
1261       res.send(result)
1262     })
1263   }
1264   catch(err) {
1265     console.log(err)
1266   }
1267 }
1268 //
1269
1270 const user_email_summary = async(req,res)=>{
1271   try {
1272     var x = []
1273     const z = await Reservations.find({_id:req.params.id}).then(result =>{
1274       x = result[0]
1275     })
1276     const MAILING = await Users.find({_id:req.params.user_id}).then(result => {
1277       mail_user = result[0].Email
1278     })
1279   }
1280 }
1281 console.log("hello")
1282 var transporter = nodemailer.createTransport({
1283   host: "smtp-mail.outlook.com", // hostname
1284   secureConnection: false, // TLS requires secureConnection to be false
1285   port: 587, // port for secure SMTP
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
src/Usercomponents/UserSearchResults.js
Line 3:10: 'Link' is defined but never used no-unused-vars
Line 37:68: Expected '=' and instead saw '=' eqeqeq

src/Usercomponents/UserSearchReturn.js
Line 3:10: 'Link' is defined but never used no-unused-vars
Line 32:68: Expected '=' and instead saw '=' eqeqeq

src/Usercomponents/User_Home.js
Line 12:3: 'Typography' is defined but never used no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

Ln 1279, Col 5 Spaces: 2 UTF-8 LF JavaScript Go Live Prettier

- The function's name is also specific indicating that this function is made for sending a summary to the user as **mail**.
- Regarding the **error return values**, some of the functions send **true** similar to **1** if they perform their aim correctly.

The screenshot shows a VS Code editor with two files open: `userController.js` and `Thankyou.js`. The `userController.js` file contains two asynchronous functions: `user_find_flight_details` and `user_email_summary`. The `user_email_summary` function uses `await` to fetch user data and then calls `MAILING` to send an email. The `Thankyou.js` file is currently empty. The terminal window at the bottom displays linting errors from ESLint, including 'no-unused-vars' and 'Expected '=' and instead saw ''=' errors in `UserSearchResults.js`, `UserSearchReturn.js`, and `User_Home.js`.

```
src > Controller > JS userController.js > [O] user_email_summary > [O] MAILING
1255
1256 const user_find_flight_details = async(req,res)=>{
1257   console.log(req.params.id)
1258   try {
1259     Flights.find({Flight_number:req.params.id}).then(result =>{
1260       console.log(result)
1261       res.send(result)
1262     })
1263   }
1264   catch(err) {
1265     console.log(err)
1266   }
1267 }
1268 //
1269
1270 const user_email_summary = async(req,res)=>{
1271   try {
1272     var x = []
1273     const z = await Reservations.find({_id:req.params.id}).then(result =>{
1274       x = result[0]
1275     })
1276     const MAILING = await Users.find({_id:req.params.user_id}).then(result => {
1277       mail_user = result[0].Email
1278     })
1279   }
1280 }
1281 console.log("hello")
1282 var transporter = nodemailer.createTransport({
1283   host: "smtp-mail.outlook.com", // hostname
1284   secureConnection: false, // TLS requires secureConnection to be false
1285   port: 587, // port for secure SMTP
1286 })
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
src/Usercomponents/UserSearchResults.js
Line 3:10: 'Link' is defined but never used no-unused-vars
Line 37:68: Expected '=' and instead saw ''= eqeqeq

src/Usercomponents/UserSearchReturn.js
Line 3:10: 'Link' is defined but never used no-unused-vars
Line 32:68: Expected '=' and instead saw ''= eqeqeq

src/Usercomponents/User_Home.js
Line 12:3: 'Typography' is defined but never used no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

Ln 1279, Col 5 Spaces: 2 UTF-8 LF JavaScript Go Live Prettier

Tech/Frame-Work used

- This website was fully developed using MERNstack, consisting of MongoDB, ExpressJS, React, and NodeJS.

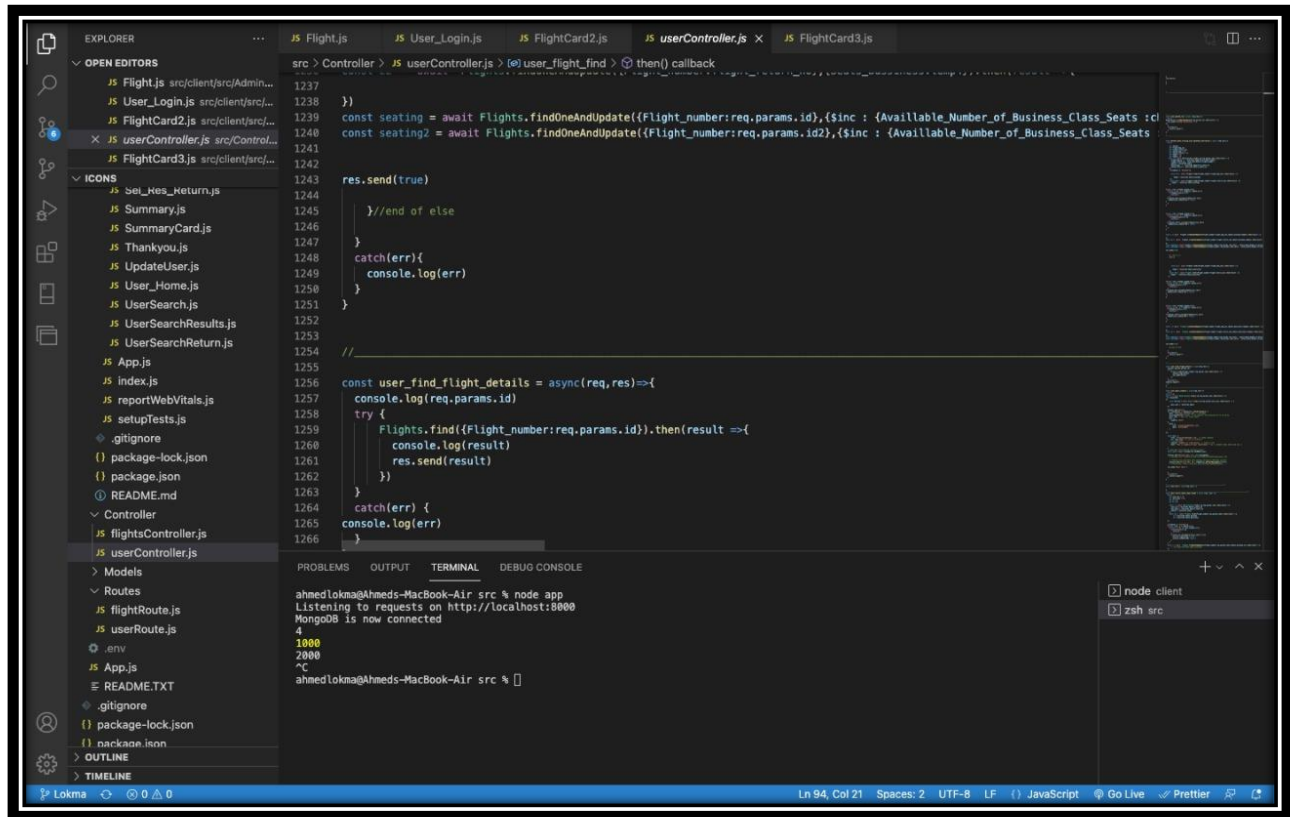
Installation

- The reader should install MongoDB, Postman, and NodeJS along with Visual Studio libraries such as React and Express to properly work with this project. Don't forget to also install react UI material packages.

Api References	
Home page:	(http://localhost:3000/)
Edit information:	(http://localhost:3000/update_info)
Search flights:	http://localhost:3000/search
Sign Up:	(http://localhost:3000/sign-up)
Login:	(http://localhost:3000/login)

Code examples:

1- Example of user login and how token is generated:



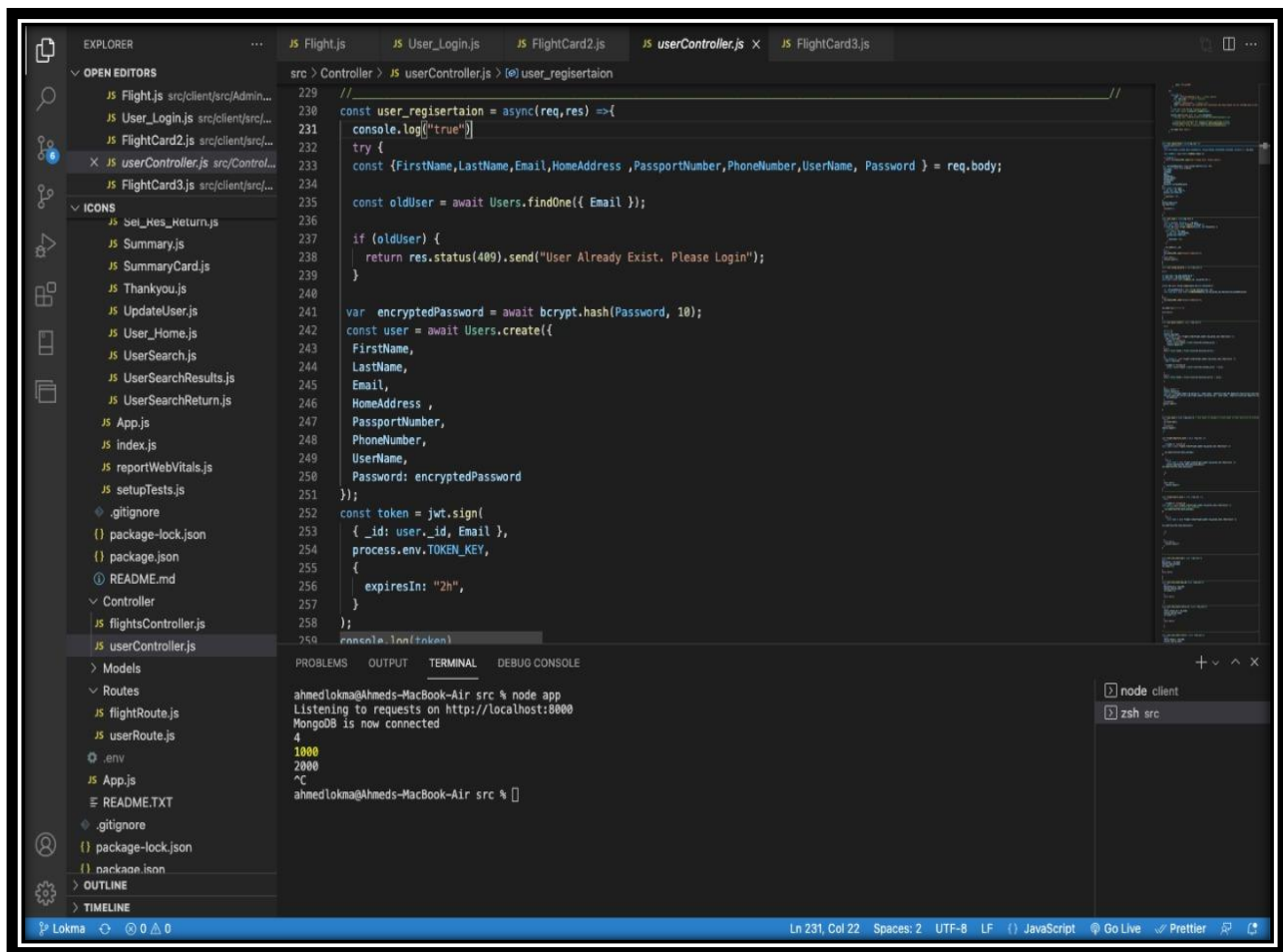
The screenshot shows a VS Code editor with the following components:

- EXPLORER:** A file tree on the left showing the project structure. The 'Controller' folder is expanded, showing files like `flightController.js`, `userController.js`, and `flightsController.js`.
- EDITOR:** The main workspace showing the `userController.js` file. The code includes a `user_login_find` function that handles user login and flight details. It uses `Flights.findOneAndUpdate` to update flight information and `Flights.find` to retrieve flight details. The code also includes error handling with `catch(err)` and `console.log(err)`.
- TERMINAL:** A terminal window at the bottom showing the command `node app` being executed. The output indicates that the application is listening on `http://localhost:8000` and that MongoDB is now connected.

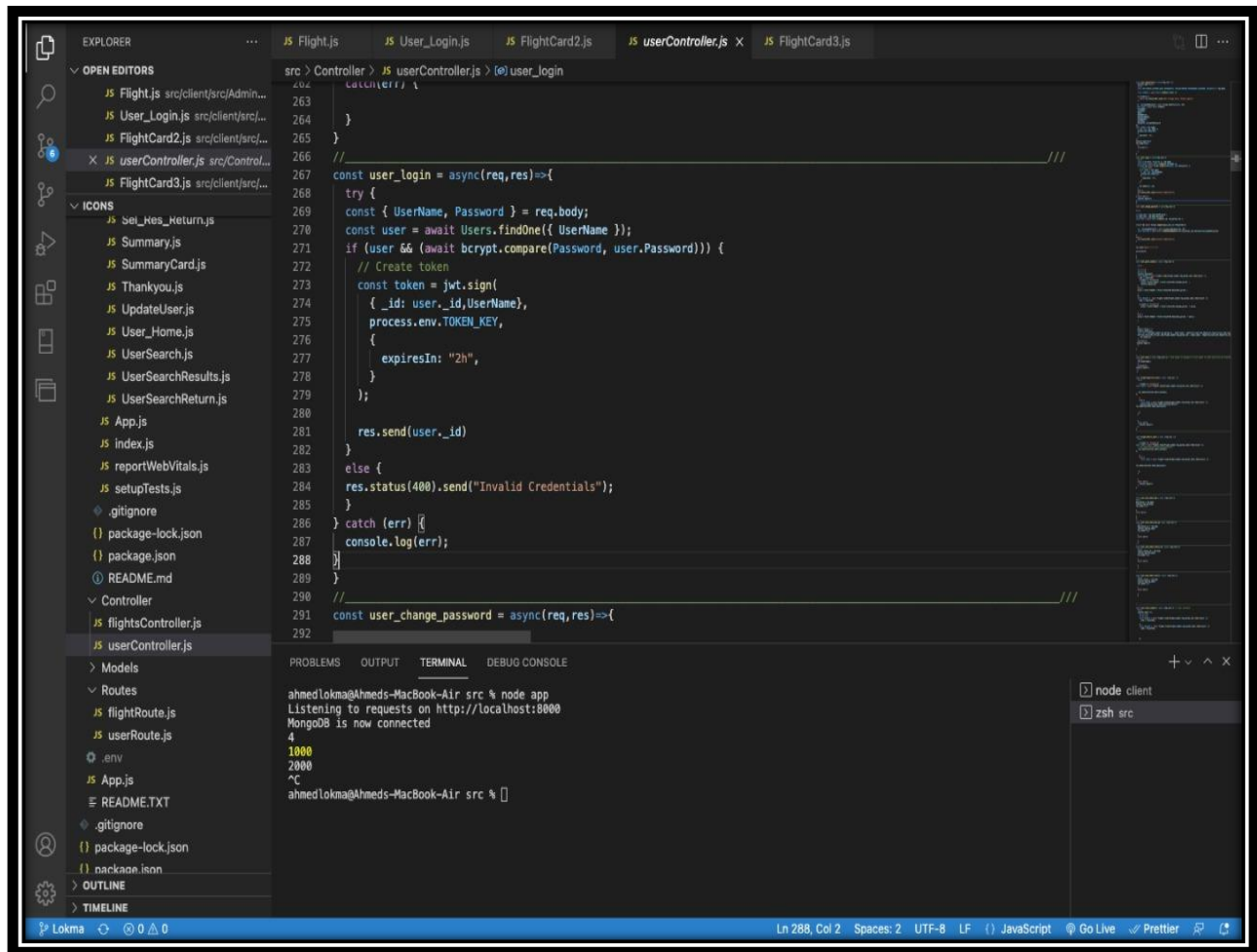
```
src > Controller > JS userController.js > @user_flight_find > then() callback
1237
1238
1239 const seating = await Flights.findOneAndUpdate({Flight_number:req.params.id},{sync : {Available_Number_of_Business_Class_Seats :cl
1240 const seating2 = await Flights.findOneAndUpdate({Flight_number:req.params.id2},{sync : {Available_Number_of_Business_Class_Seats
1241
1242
1243 res.send(true)
1244
1245 //end of else
1246
1247 catch(err){
1248 console.log(err)
1249 }
1250
1251 //
1252
1253 const user_find_flight_details = async(req,res)=>{
1254 console.log(req.params.id)
1255 try {
1256 Flights.find({Flight_number:req.params.id}).then(result =>{
1257 console.log(result)
1258 res.send(result)
1259 })
1260 catch(err) {
1261 console.log(err)
1262 }
1263
1264
1265
1266
```

ahmedlokma@Ahmeds-MacBook-Air src % node app
Listening to requests on http://localhost:8000
MongoDB is now connected
4
1000
2000
^C
ahmedlokma@Ahmeds-MacBook-Air src %

2- Example of user sign up and generating token:



3- Example of finding specific flight detail:



Tests

- There's no test provided

How to use?

First of all download the project then run backend terminal on port 8000(cd src => node app) and front end terminal on port 3000(cd src => cd client => npm run start)

- you will be directed to homepage
- click on the button search to be redirected to search
- you can choose departure and return flight , but you must be existing user to choose seats
- all the process is sequential

Features:

There are numerous facilities and features that the user benefits from while using the website, such as having the seats visible to their eyes. The user is able to select the cabin. All of the requirements and necessities that the user would need to acquire information about are present on the web-site.

Contribute

- Any contributions in the **UI** will be very useful and helpful

Credits:

- Professor Mervat
- Eng. Nada Sharawy
- Eng. Nada Hesham
- Eng. Ahmed Helmy