FSD Lab 6

Aim: Develop a set of REST API using Express and Node.

Objectives:

1. To define HTTP GET and POST operations.
2. To understand and make use of ‘REST’, ‘a REST endpoint’, ‘API Integration’, and ‘API Invocation’
3. To understand the use of a REST Client to make POST and GET requests to an API.

Theory:

1. What is REST API?

   REST (Representational State Transfer) is an architectural style for designing networked applications, with REST APIs serving as the communication mechanism between software systems. RESTful APIs use standard HTTP methods to manipulate resources identified by URIs. They follow a stateless model, where each request contains all the information needed for communication.

2. Main purpose of REST API.

   Main Purpose of REST API:

   The primary objectives of REST APIs are:

   1. Interoperability: Enable seamless communication between diverse applications and platforms.

   2. Scalability: Support the distribution of services across servers for horizontal scalability.

   3. Simplicity and Uniformity: Follow a straightforward and uniform approach using standard HTTP methods.

   4. Resource Manipulation: Focus on manipulating resources identified by URIs using standard CRUD operations.

   5. Statelessness: Embrace a stateless communication model, simplifying implementation and enhancing reliability.

   In essence, REST APIs facilitate standardized, scalable, and interoperable communication between different software systems over the web.

FAQ:

1. What are HTTP Request types?

HTTP (Hypertext Transfer Protocol) supports various request methods indicating the intended action on a resource:

1. GET: Retrieves data from a specified resource without modification.

2. POST: Submits data for processing, often creating a new resource on the server.

3. PUT: Updates a resource or creates a new one if it doesn't exist, replacing the entire resource.

4. PATCH: Applies partial modifications to a resource, updating it with provided changes.

5. DELETE: Requests the removal of a specified resource.

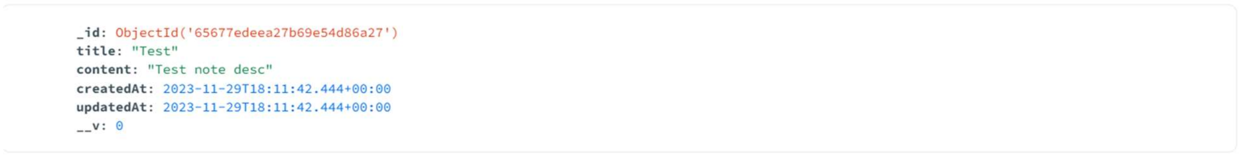
6. HEAD: Retrieves only the headers of a resource for status checks without downloading the content.

7. OPTIONS: Describes communication options for the target resource, allowing the client to determine allowed methods.

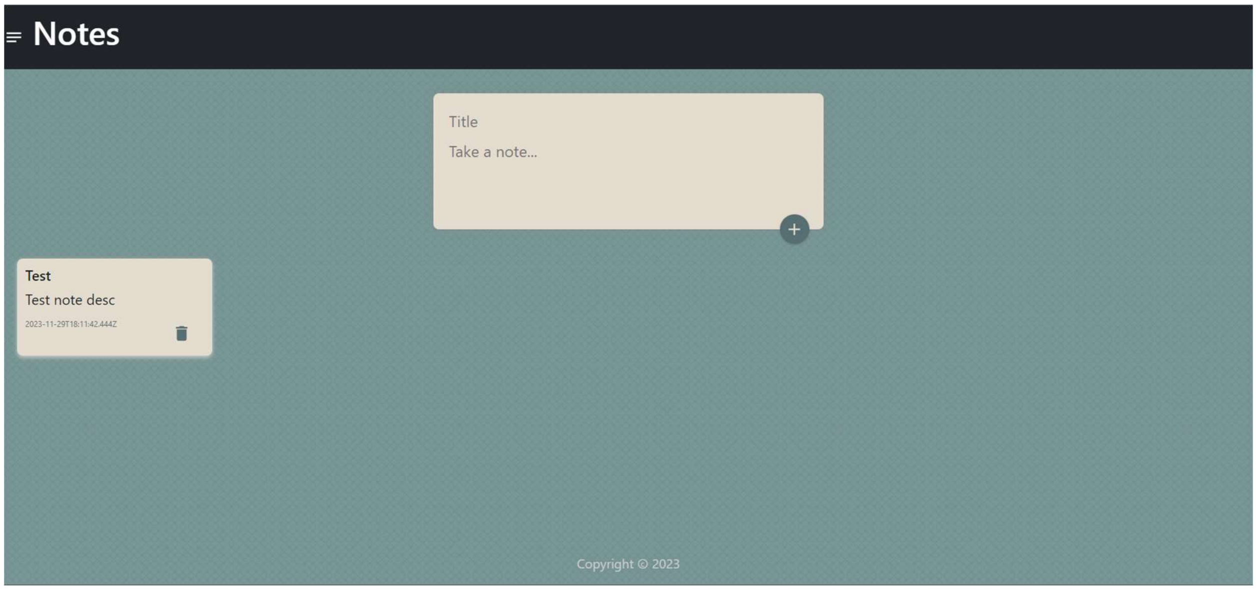
8. TRACE: Performs a message loop-back test for diagnostic purposes along the path to the target resource.

These HTTP request methods serve specific purposes, and the choice depends on the desired action and the nature of the operation.

SCREENSHOTS –



```
_id: ObjectId('65677edeea27b69e54d86a27')
title: "Test"
content: "Test note desc"
createdAt: 2023-11-29T18:11:42.444+00:00
updatedAt: 2023-11-29T18:11:42.444+00:00
__v: 0
```

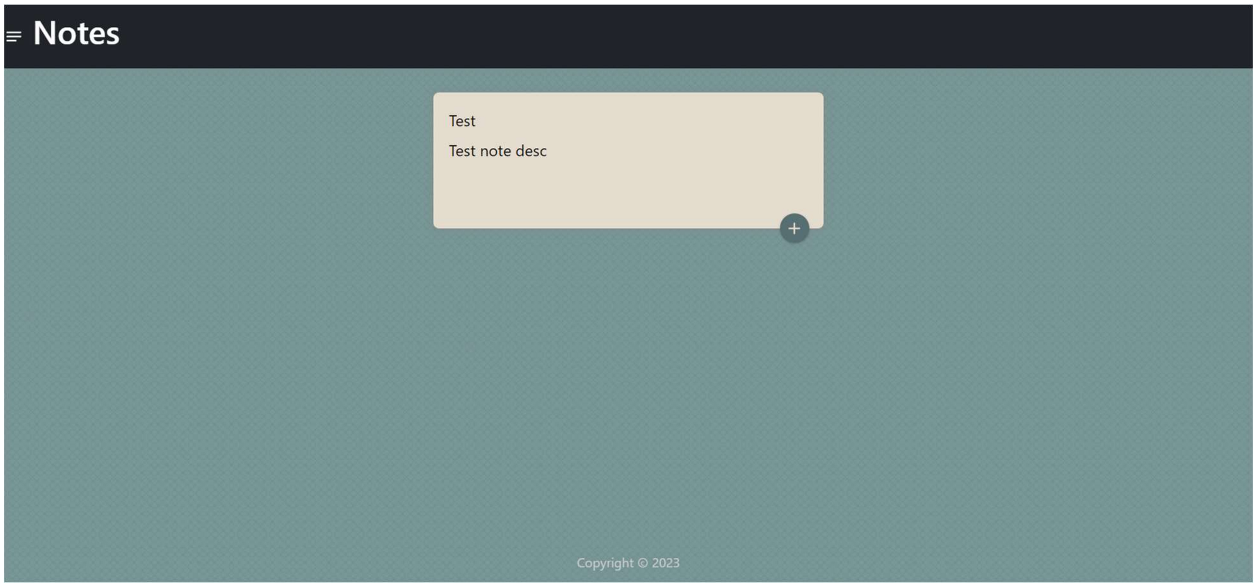


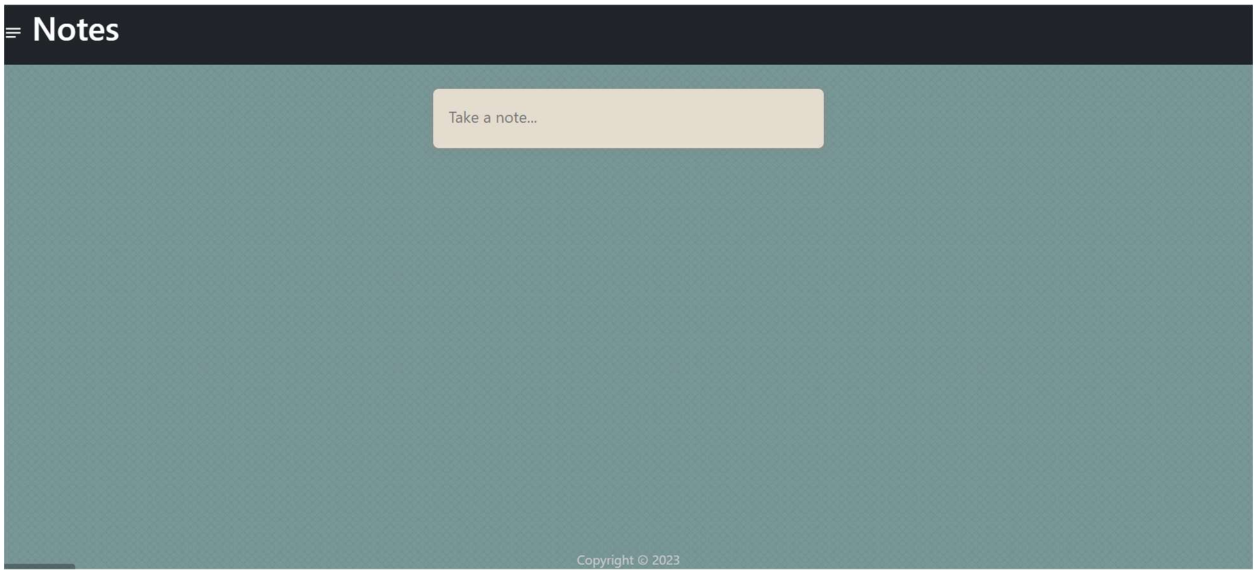
Test

Test note desc

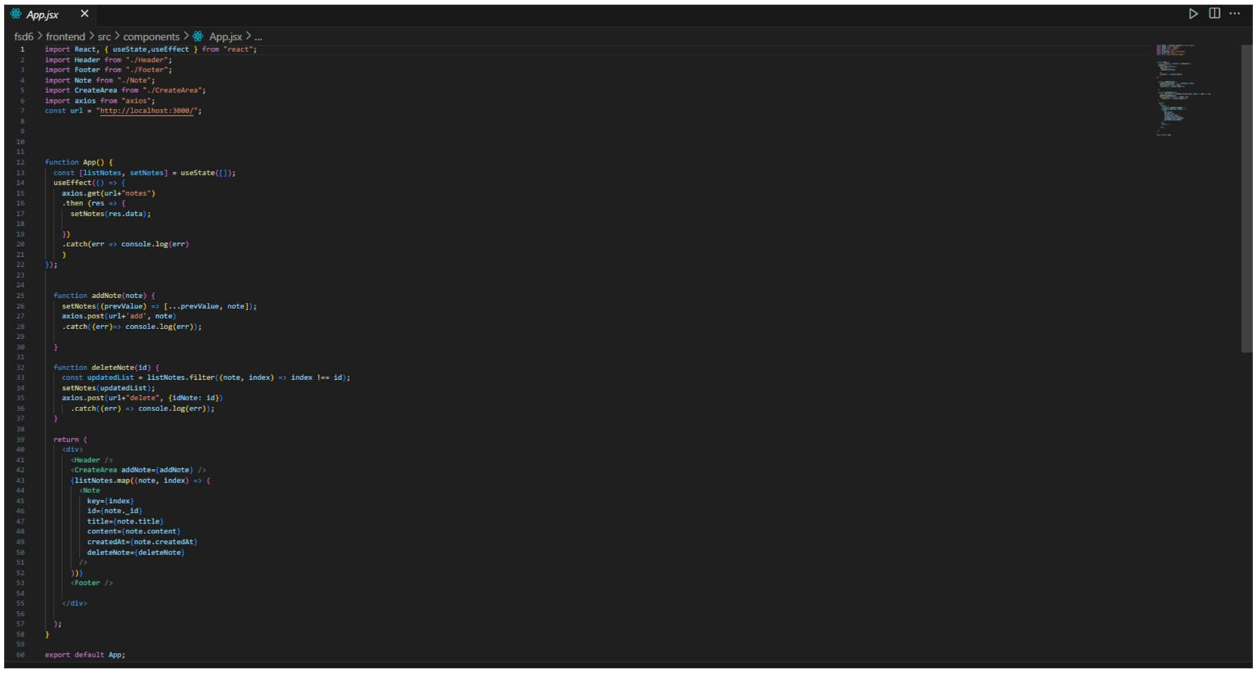
2023-11-29T18:11:42.444Z

Copyright © 2023

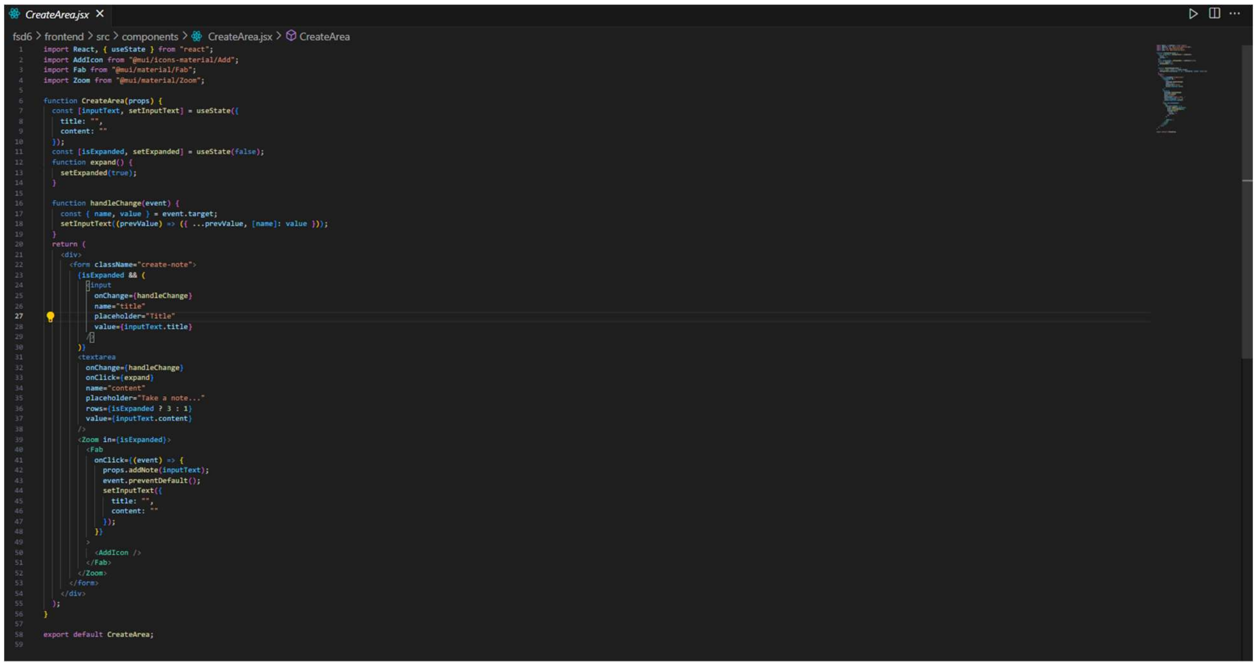

Notes
Test
Test note desc
Copyright © 2023


Notes
Take a note...
Copyright © 2023

CODES –



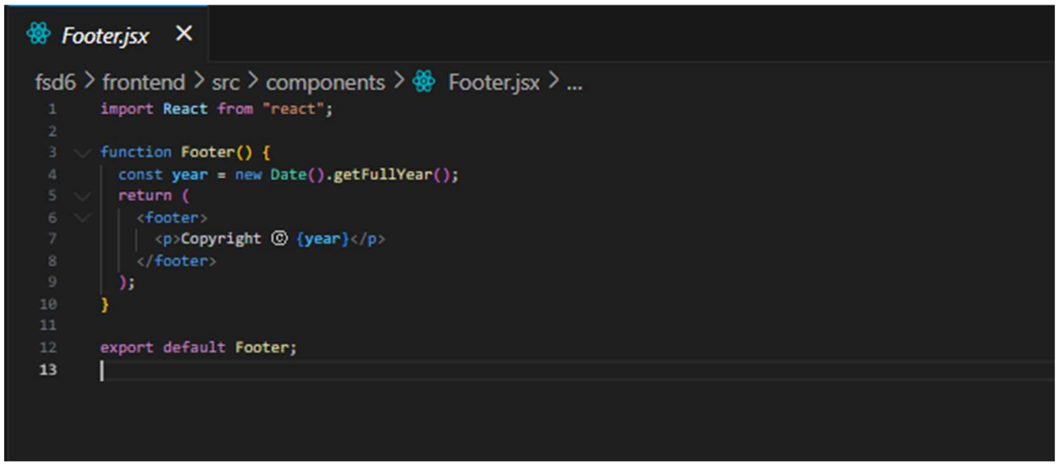
```jsx
import React, { useState,useEffect } from "react";
import Header from "./Header";
import Footer from "./Footer";
import Note from "./Note";
import CreateArea from "./CreateArea";
import axios from "axios";
const url = "http://localhost:3000/";




function App() {
  const [listNotes, setNotes] = useState([]);
  useEffect(() => {
    axios.get(url+"notes")
    .then (res => {
      setNotes(res.data);

    })
    .catch(err => console.log(err)
    )
});


  function addNote(note) {
    setNotes((prevValue) => [...prevValue, note]);
    axios.post(url+'add', note)
    .catch((err)=> console.log(err));

  }

  function deleteNote(id) {
    const updatedList = listNotes.filter((note, index) => index !== id);
    setNotes(updatedList);
    axios.post(url+"delete", {idNote: id})
      .catch((err) => console.log(err));
  }

  return (
    <div>
      <Header />
      <CreateArea addNote={addNote} />
      {listNotes.map((note, index) => (
        <Note
          key={index}
          id={note._id}
          title={note.title}
          content={note.content}
          createdAt={note.createdAt}
          deleteNote={deleteNote}
        />
      ))}
      <Footer />

    </div>

  );
}

export default App;
```



CreateArea.jsx

fsd6 > frontend > src > components > CreateArea.jsx > CreateArea

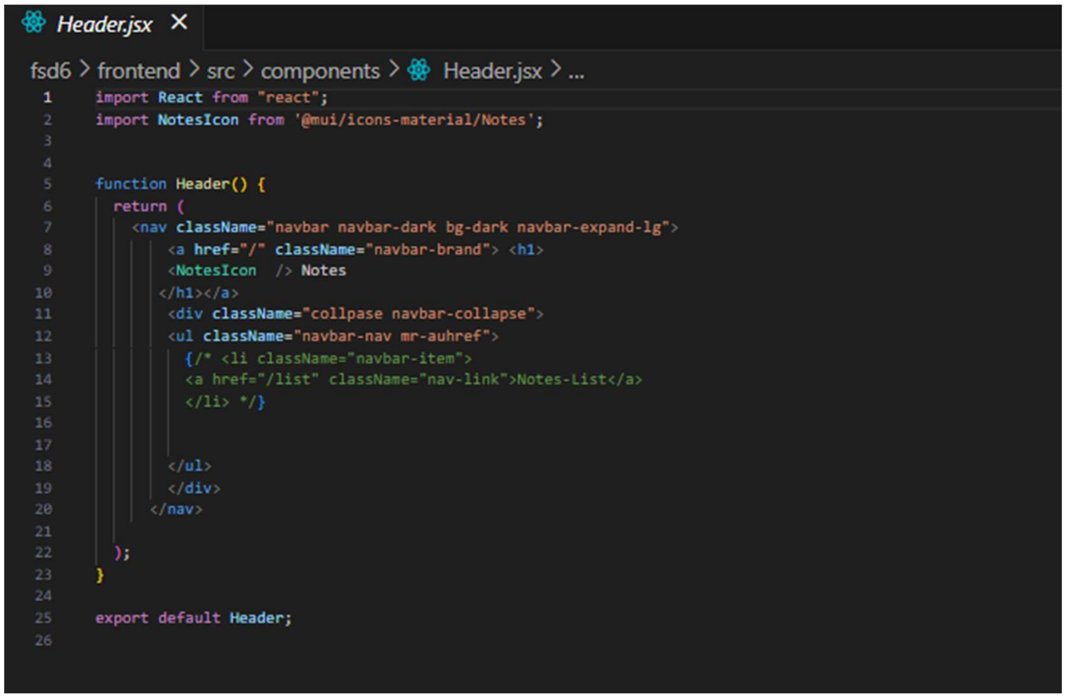
```jsx
import React, { useState } from "react";
import AddIcon from "@mui/icons-material/Add";
import Fab from "@mui/material/Fab";
import Zoom from "@mui/material/Zoom";

function CreateArea(props) {
  const [inputText, setInputText] = useState({
    title: "",
    content: ""
  });
  const [isExpanded, setExpanded] = useState(false);
  function expand() {
    setExpanded(true);
  }

  function handleChange(event) {
    const { name, value } = event.target;
    setInputText((prevValue) => ({ ...prevValue, [name]: value }));
  }
  return (
    <div>
      <form className="create-note">
        {isExpanded && (
          <input
            onChange={handleChange}
            name="title"
            placeholder="Title"
            value={inputText.title}
          />
        )}
        <textarea
          onChange={handleChange}
          onClick={expand}
          name="content"
          placeholder="Take a note..."
          rows={isExpanded ? 3 : 1}
          value={inputText.content}
        />
        <Zoom in={isExpanded}>
          <Fab
            onClick={(event) => {
              props.addNote(inputText);
              event.preventDefault();
              setInputText({
                title: "",
                content: ""
              });
            }}
          >
            <AddIcon />
          </Fab>
        </Zoom>
      </form>
    </div>
  );
}

export default CreateArea;
```



Footer.jsx

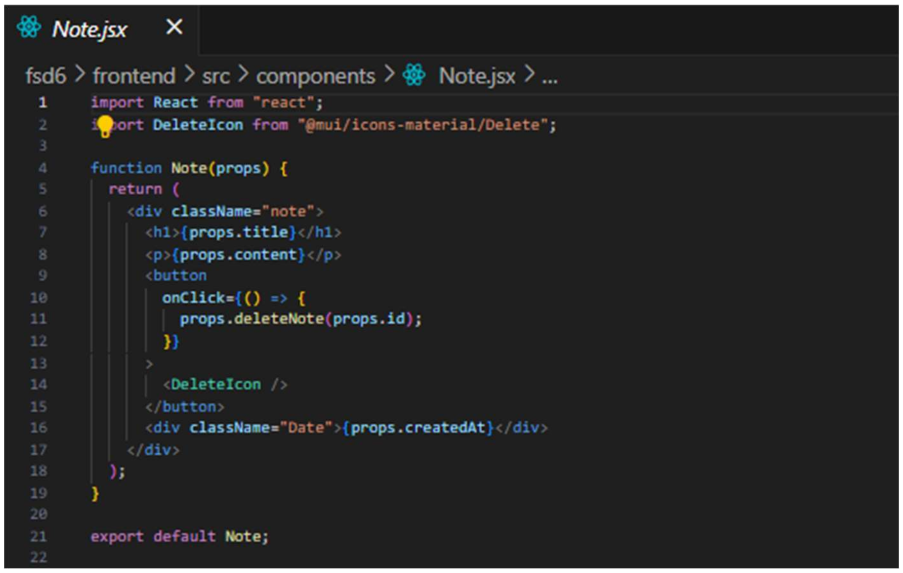
fsd6 > frontend > src > components > Footer.jsx > ...

```jsx
import React from "react";

function Footer() {
  const year = new Date().getFullYear();
  return (
    <footer>
      <p>Copyright © {year}</p>
    </footer>
  );
}

export default Footer;
```


Header.jsx

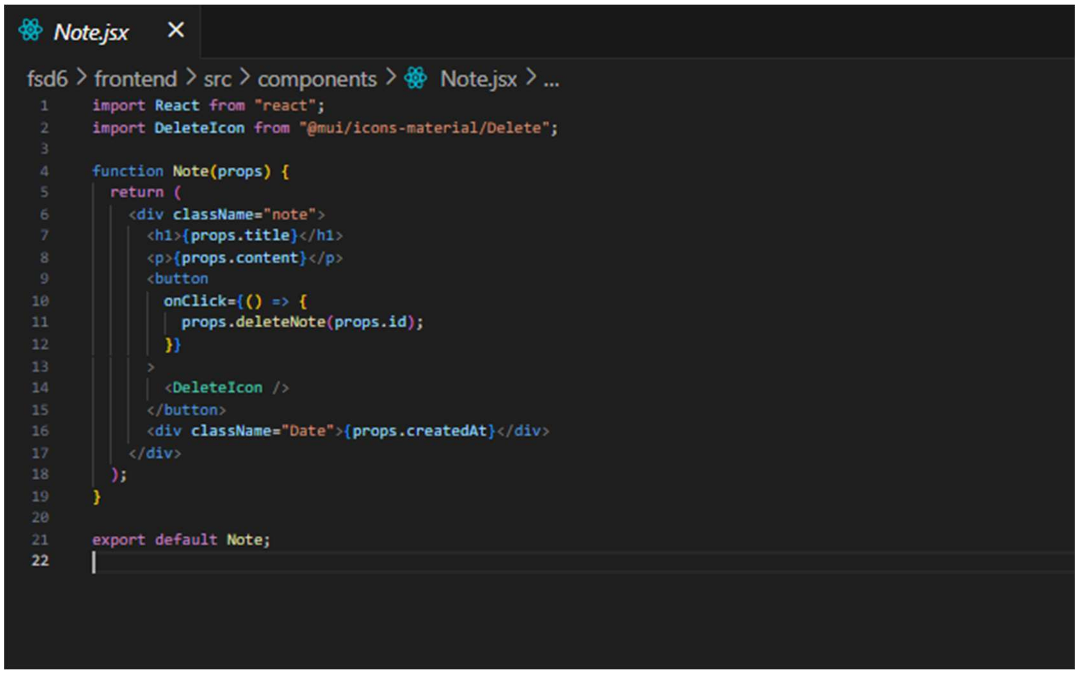
fsd6 > frontend > src > components > Header.jsx > ...

```jsx
import React from "react";
import NotesIcon from '@mui/icons-material/Notes';


function Header() {
  return (
    <nav className="navbar navbar-dark bg-dark navbar-expand-lg">
        <a href="/" className="navbar-brand"> <h1>
        <NotesIcon  /> Notes
       </h1></a>
        <div className="collpase navbar-collapse">
        <ul className="navbar-nav mr-auhref">
          {/* <li className="navbar-item">
          <a href="/list" className="nav-link">Notes-List</a>
          </li> */}


        </ul>
        </div>
      </nav>

  );
}

export default Header;

```


Note.jsx

fsd6 > frontend > src > components > Note.jsx > ...

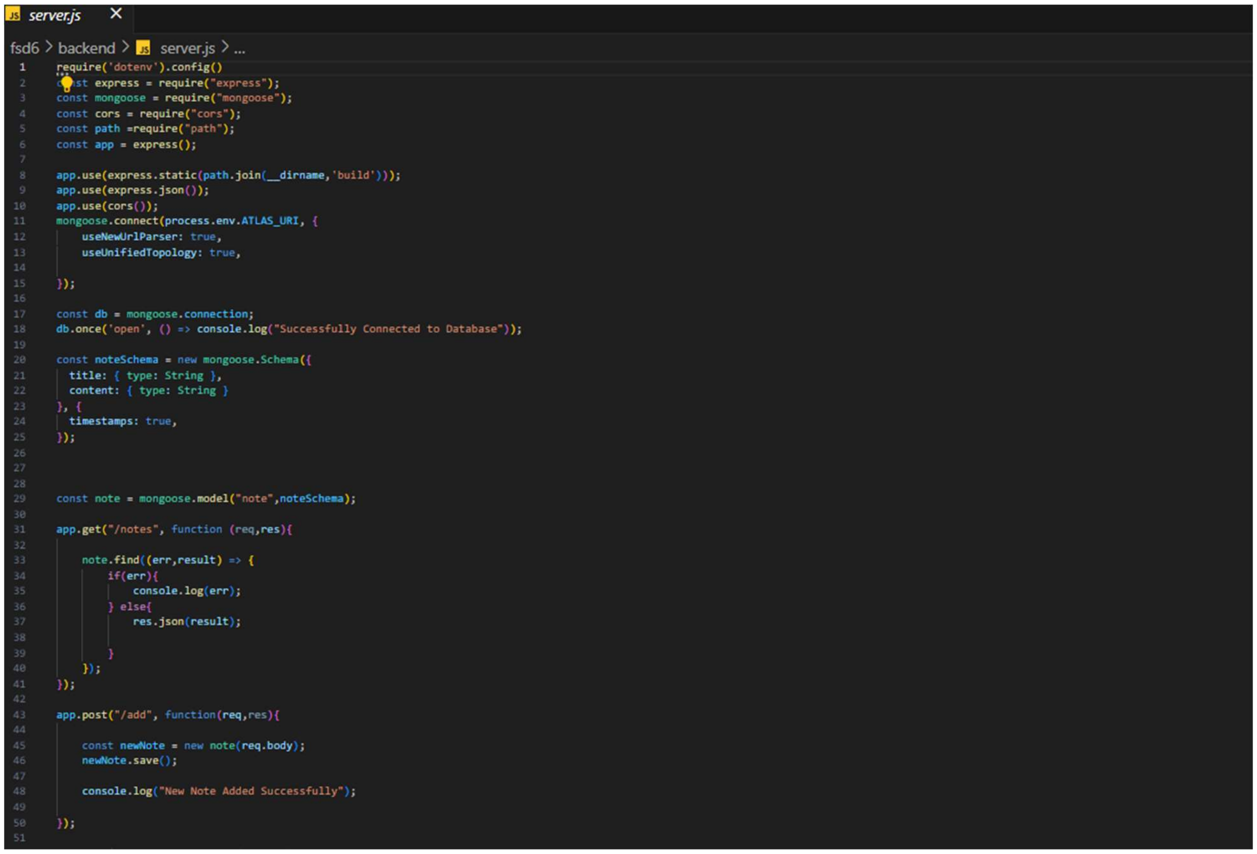
```jsx
import React from "react";
import DeleteIcon from "@mui/icons-material/Delete";

function Note(props) {
  return (
    <div className="note">
      <h1>{props.title}</h1>
      <p>{props.content}</p>
      <button
        onClick={() => {
          props.deleteNote(props.id);
        }}
      >
        <DeleteIcon />
      </button>
      <div className="Date">{props.createdAt}</div>
    </div>
  );
}

export default Note;

```



Note.jsx

fsd6 > frontend > src > components > Note.jsx > ...

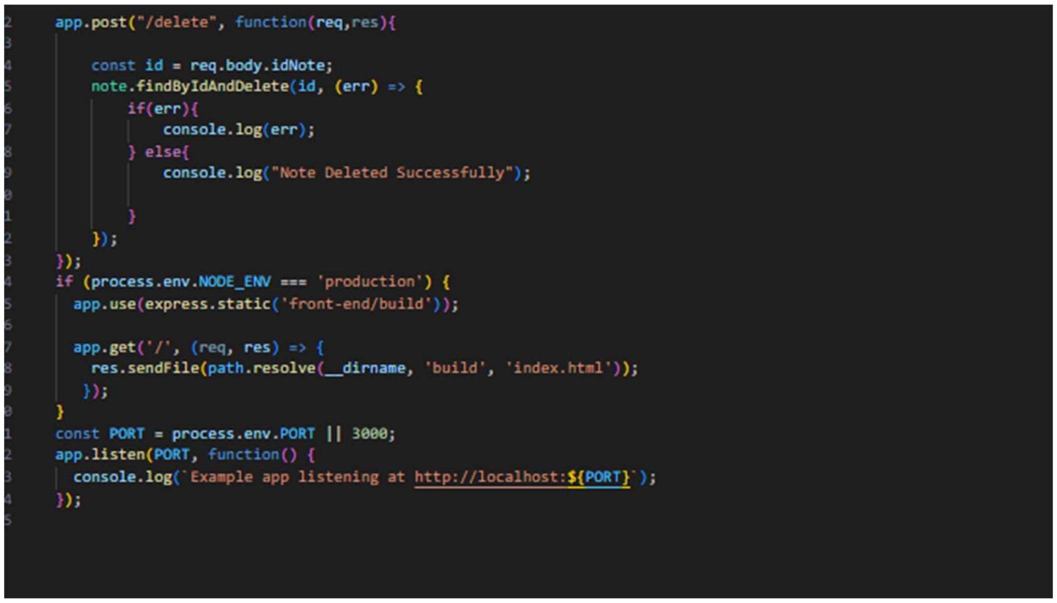
```jsx
import React from "react";
import DeleteIcon from "@mui/icons-material/Delete";

function Note(props) {
  return (
    <div className="note">
      <h1>{props.title}</h1>
      <p>{props.content}</p>
      <button
        onClick={() => {
          props.deleteNote(props.id);
        }}
      >
        <DeleteIcon />
      </button>
      <div className="Date">{props.createdAt}</div>
    </div>
  );
}

export default Note;

```



server.js

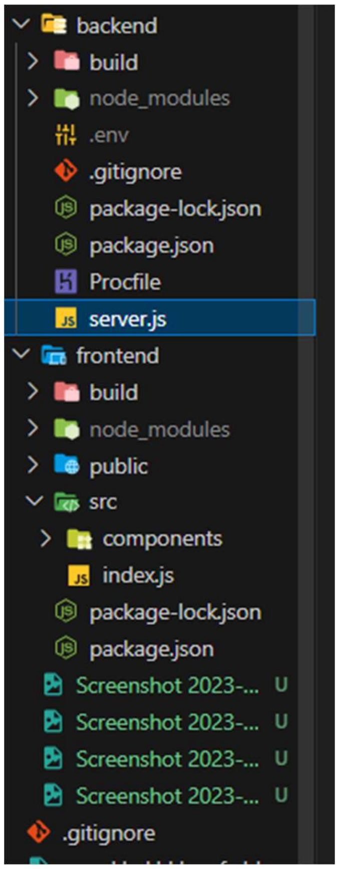
fsd6 > backend > server.js > ...

```javascript
require('dotenv').config()
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const path =require("path");
const app = express();

app.use(express.static(path.join(__dirname,'build')));
app.use(express.json());
app.use(cors());
mongoose.connect(process.env.ATLAS_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,

});

const db = mongoose.connection;
db.once('open', () => console.log("Successfully Connected to Database"));

const noteSchema = new mongoose.Schema({
  title: { type: String },
  content: { type: String }
}, {
  timestamps: true,
});



const note = mongoose.model("note",noteSchema);

app.get("/notes", function (req,res){

    note.find((err,result) => {
        if(err){
            console.log(err);
        } else{
            res.json(result);

        }
    });
});

app.post("/add", function(req,res){

    const newNote = new note(req.body);
    newNote.save();

    console.log("New Note Added Successfully");

});

```



```
app.post("/delete", function(req,res){

    const id = req.body.idNote;
    note.findByIdAndDelete(id, (err) => {
        if(err){
            console.log(err);
        } else{
            console.log("Note Deleted Successfully");

        }
    });
});
if (process.env.NODE_ENV === 'production') {
  app.use(express.static('front-end/build'));

  app.get('/', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'build', 'index.html'));
  });
}
const PORT = process.env.PORT || 3000;
app.listen(PORT, function() {
  console.log(`Example app listening at http://localhost:${PORT}`);
});
```

FILE STRUCTURE



```
backend
  build
  node_modules
  .env
  .gitignore
  package-lock.json
  package.json
  Procfile
  server.js
frontend
  build
  node_modules
  public
  src
    components
    index.js
  package-lock.json
  package.json
  Screenshot 2023-...  U
  Screenshot 2023-...  U
  Screenshot 2023-...  U
  Screenshot 2023-...  U
.gitignore
```