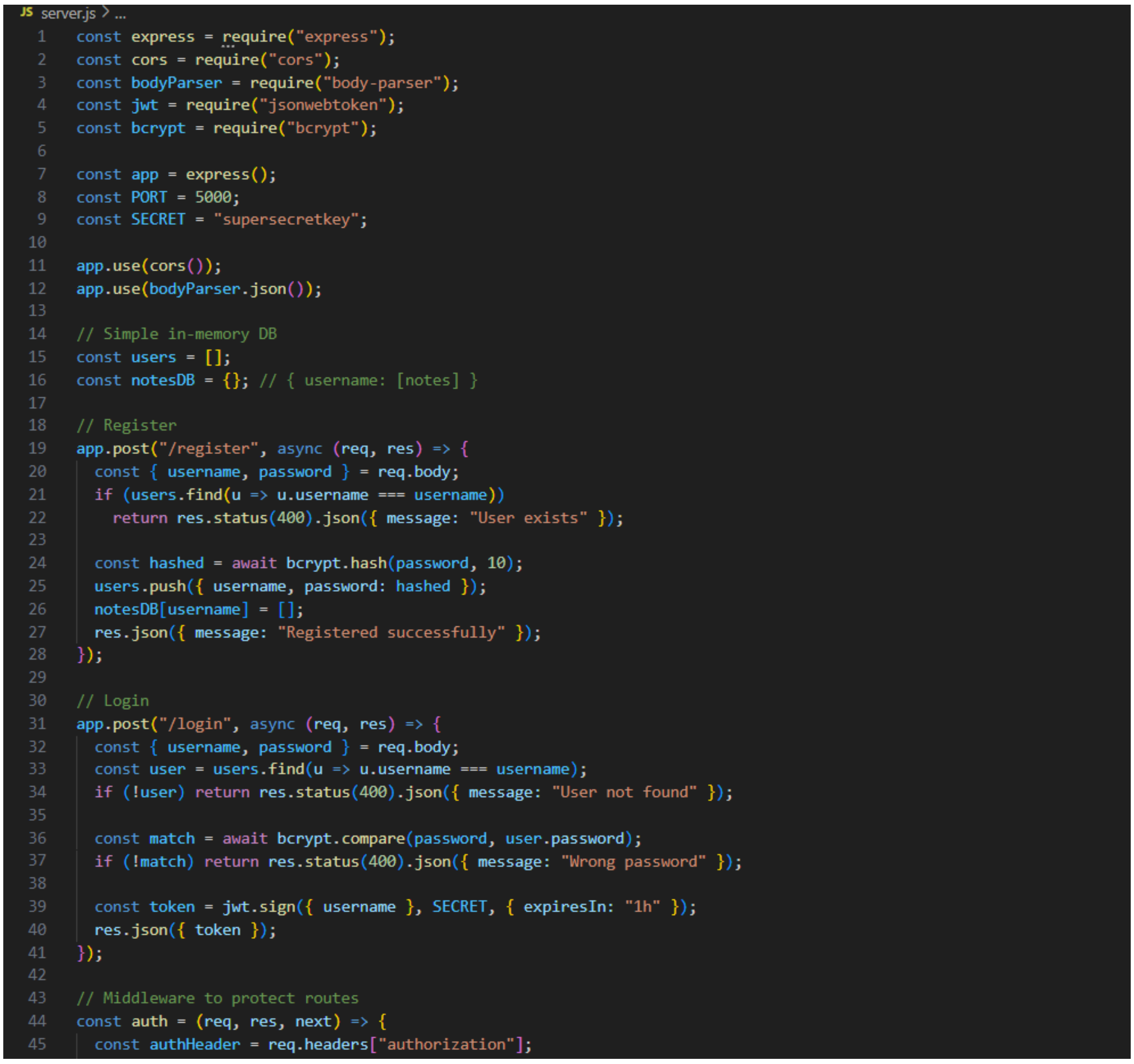
EXPERIMENT NO 5

D15A-04

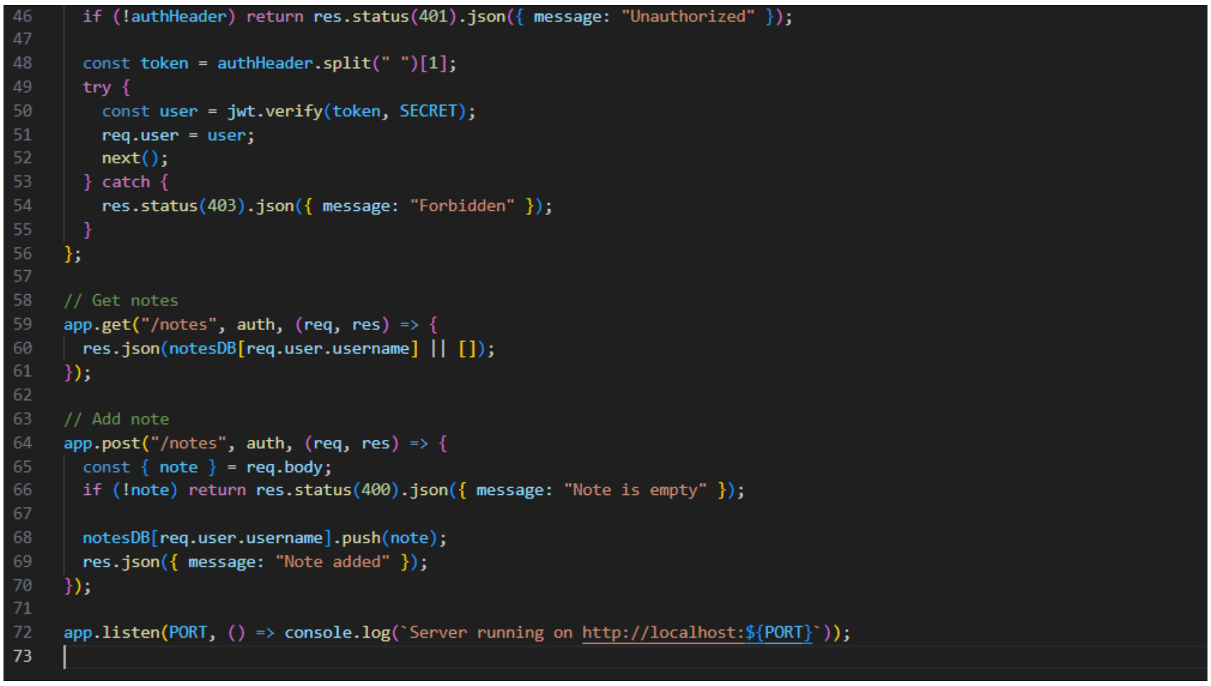
SOURCE CODE:

server.js



server.js > ...

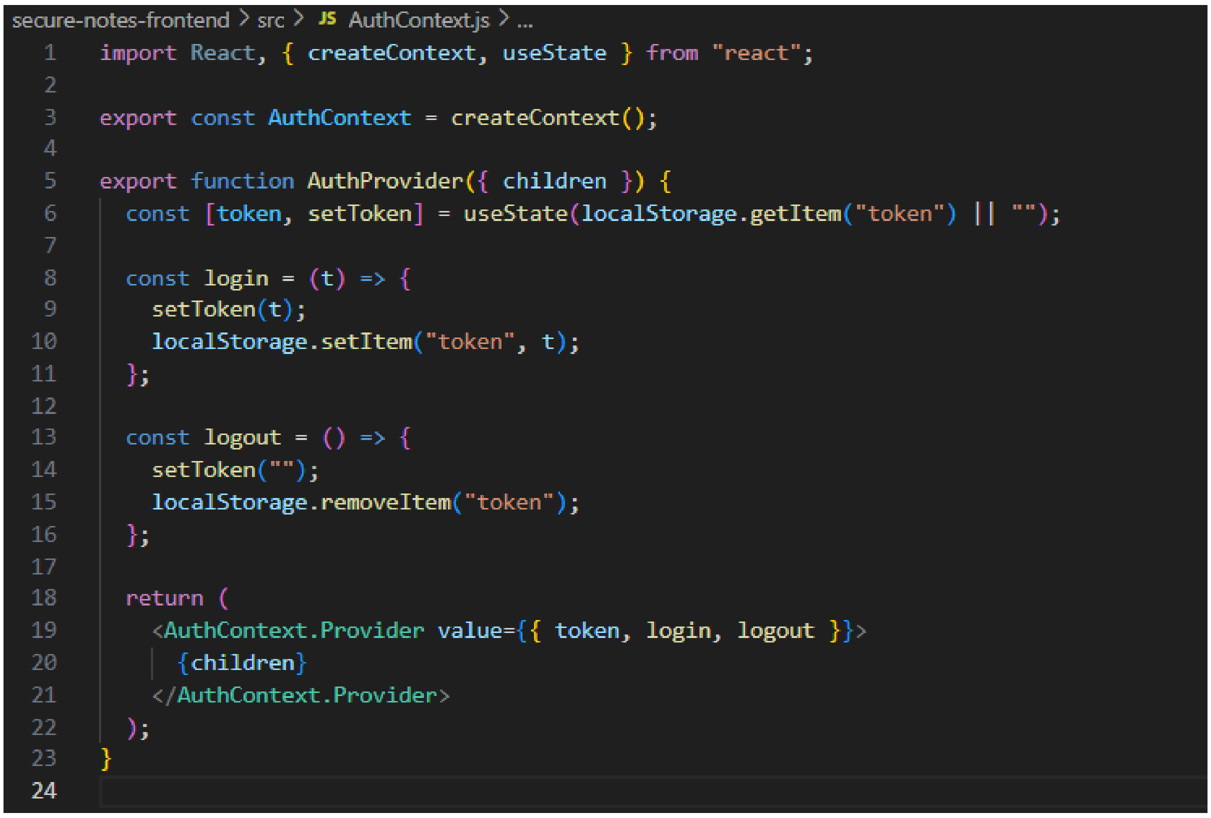
```javascript
const express = require("express");
const cors = require("cors");
const bodyParser = require("body-parser");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcrypt");

const app = express();
const PORT = 5000;
const SECRET = "supersecretkey";

app.use(cors());
app.use(bodyParser.json());

// Simple in-memory DB
const users = [];
const notesDB = {}; // { username: [notes] }

// Register
app.post("/register", async (req, res) => {
  const { username, password } = req.body;
  if (users.find(u => u.username === username))
    return res.status(400).json({ message: "User exists" });

  const hashed = await bcrypt.hash(password, 10);
  users.push({ username, password: hashed });
  notesDB[username] = [];
  res.json({ message: "Registered successfully" });
});

// Login
app.post("/login", async (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username);
  if (!user) return res.status(400).json({ message: "User not found" });

  const match = await bcrypt.compare(password, user.password);
  if (!match) return res.status(400).json({ message: "Wrong password" });

  const token = jwt.sign({ username }, SECRET, { expiresIn: "1h" });
  res.json({ token });
});

// Middleware to protect routes
const auth = (req, res, next) => {
  const authHeader = req.headers["authorization"];
```



```
  if (!authHeader) return res.status(401).json({ message: "Unauthorized" });

  const token = authHeader.split(" ")[1];
  try {
    const user = jwt.verify(token, SECRET);
    req.user = user;
    next();
  } catch {
    res.status(403).json({ message: "Forbidden" });
  }
};

// Get notes
app.get("/notes", auth, (req, res) => {
  res.json(notesDB[req.user.username] || []);
});

// Add note
app.post("/notes", auth, (req, res) => {
  const { note } = req.body;
  if (!note) return res.status(400).json({ message: "Note is empty" });

  notesDB[req.user.username].push(note);
  res.json({ message: "Note added" });
});

app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));

```

Frontend

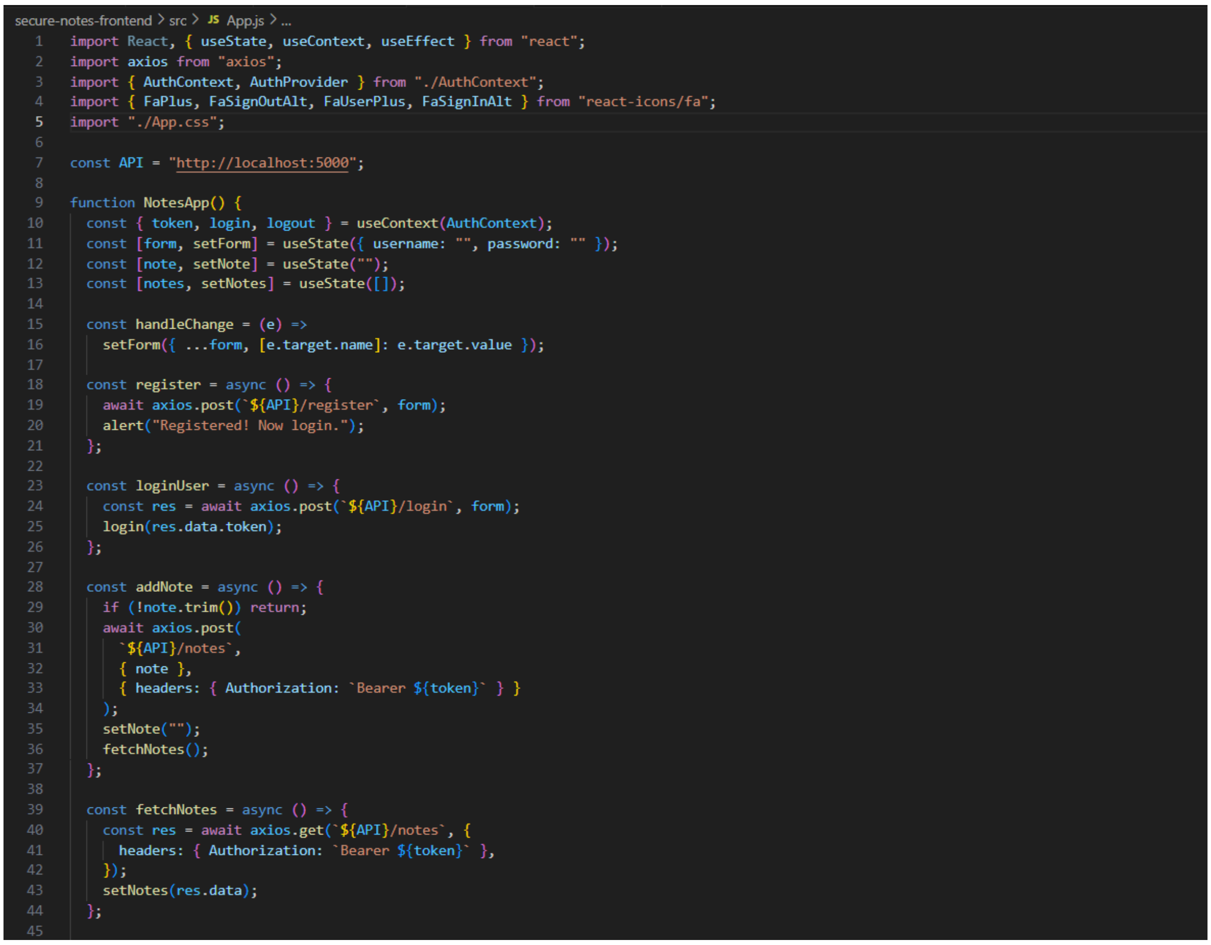
AuthContext.js



secure-notes-frontend > src > JS AuthContext.js > ...

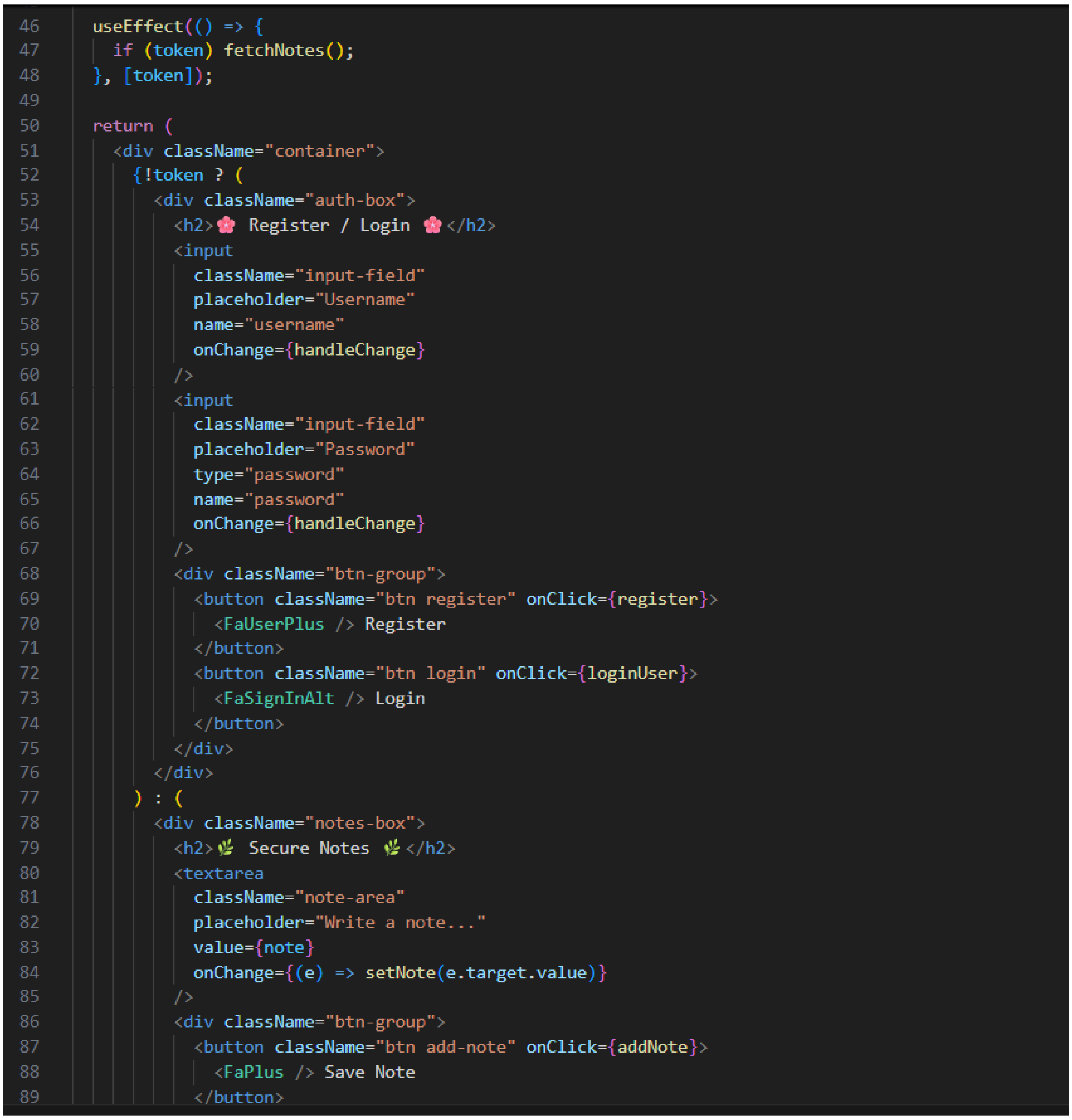
```
import React, { createContext, useState } from "react";

export const AuthContext = createContext();

export function AuthProvider({ children }) {
  const [token, setToken] = useState(localStorage.getItem("token") || "");

  const login = (t) => {
    setToken(t);
    localStorage.setItem("token", t);
  };

  const logout = () => {
    setToken("");
    localStorage.removeItem("token");
  };

  return (
    <AuthContext.Provider value={{ token, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
}

```

App.js

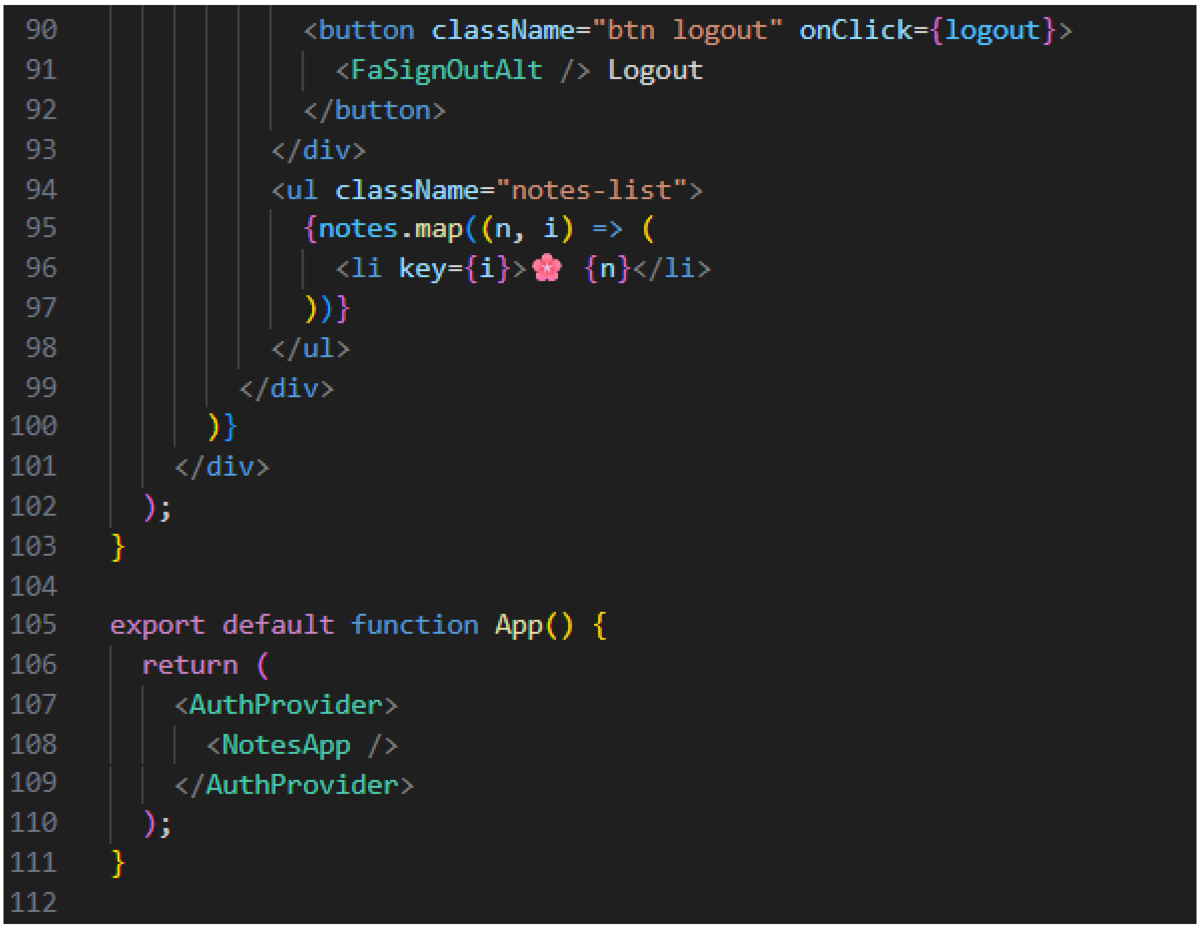


secure-notes-frontend > src > JS App.js > ...

```js
import React, { useState, useContext, useEffect } from "react";
import axios from "axios";
import { AuthContext, AuthProvider } from "./AuthContext";
import { FaPlus, FaSignOutAlt, FaUserPlus, FaSignInAlt } from "react-icons/fa";
import "./App.css";

const API = "http://localhost:5000";

function NotesApp() {
  const { token, login, logout } = useContext(AuthContext);
  const [form, setForm] = useState({ username: "", password: "" });
  const [note, setNote] = useState("");
  const [notes, setNotes] = useState([]);

  const handleChange = (e) =>
    setForm({ ...form, [e.target.name]: e.target.value });

  const register = async () => {
    await axios.post(`${API}/register`, form);
    alert("Registered! Now login.");
  };

  const loginUser = async () => {
    const res = await axios.post(`${API}/login`, form);
    login(res.data.token);
  };

  const addNote = async () => {
    if (!note.trim()) return;
    await axios.post(
      `${API}/notes`,
      { note },
      { headers: { Authorization: `Bearer ${token}` } }
    );
    setNote("");
    fetchNotes();
  };

  const fetchNotes = async () => {
    const res = await axios.get(`${API}/notes`, {
      headers: { Authorization: `Bearer ${token}` },
    });
    setNotes(res.data);
  };

```



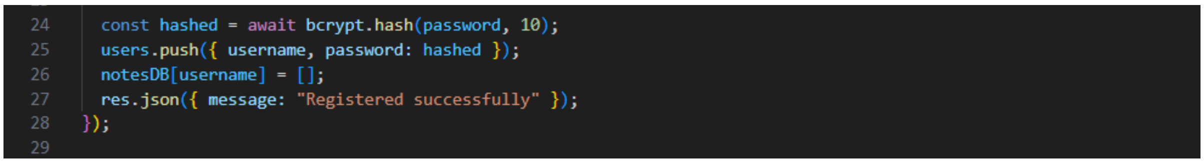
```
  useEffect(() => {
    if (token) fetchNotes();
  }, [token]);

  return (
    <div className="container">
      {!token ? (
        <div className="auth-box">
          <h2>🌸 Register / Login 🌸</h2>
          <input
            className="input-field"
            placeholder="Username"
            name="username"
            onChange={handleChange}
          />
          <input
            className="input-field"
            placeholder="Password"
            type="password"
            name="password"
            onChange={handleChange}
          />
          <div className="btn-group">
            <button className="btn register" onClick={register}>
              <FaUserPlus /> Register
            </button>
            <button className="btn login" onClick={loginUser}>
              <FaSignInAlt /> Login
            </button>
          </div>
        </div>
      ) : (
        <div className="notes-box">
          <h2>🌿 Secure Notes 🌿</h2>
          <textarea
            className="note-area"
            placeholder="Write a note..."
            value={note}
            onChange={(e) => setNote(e.target.value)}
          />
          <div className="btn-group">
            <button className="btn add-note" onClick={addNote}>
              <FaPlus /> Save Note
            </button>
```



```jsx
            <button className="btn logout" onClick={logout}>
              <FaSignOutAlt /> Logout
            </button>
          </div>
          <ul className="notes-list">
            {notes.map((n, i) => (
              <li key={i}>🌸 {n}</li>
            ))}
          </ul>
        </div>
      )}
    </div>
  );
}

export default function App() {
  return (
    <AuthProvider>
      <NotesApp />
    </AuthProvider>
  );
}
```

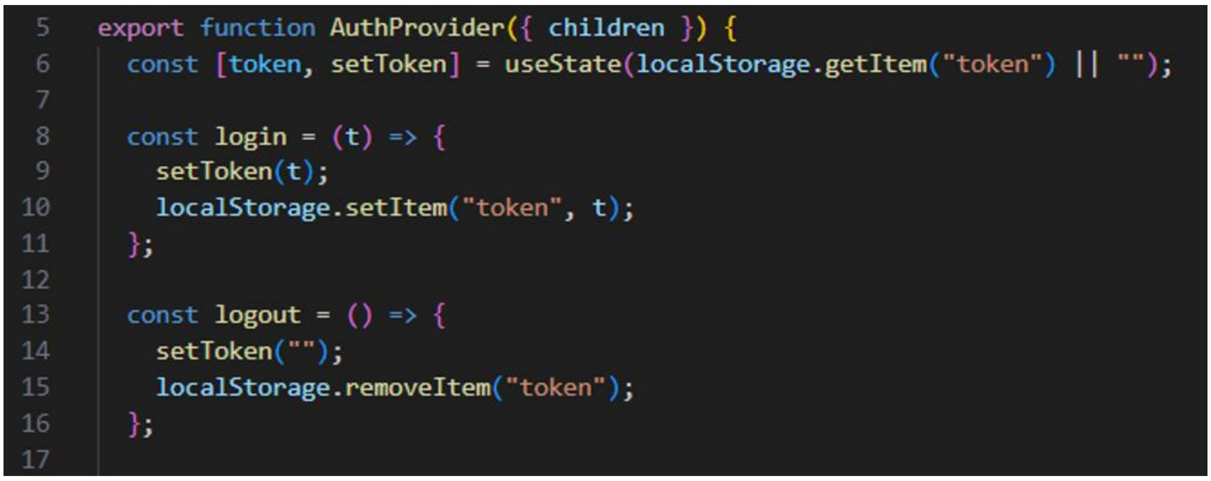
EXTRA 30%

Password hashing



```js
  const hashed = await bcrypt.hash(password, 10);
  users.push({ username, password: hashed });
  notesDB[username] = [];
  res.json({ message: "Registered successfully" });
});
```

Local storage



```js
export function AuthProvider({ children }) {
  const [token, setToken] = useState(localStorage.getItem("token") || "");

  const login = (t) => {
    setToken(t);
    localStorage.setItem("token", t);
  };

  const logout = () => {
    setToken("");
    localStorage.removeItem("token");
  };

```

OUTPUT:

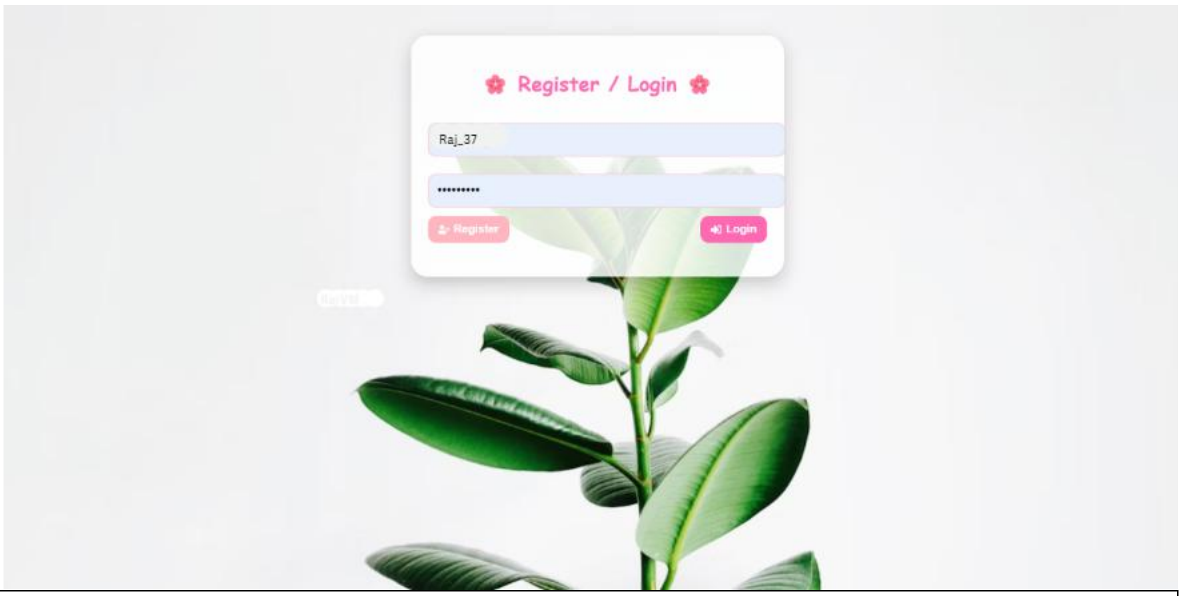
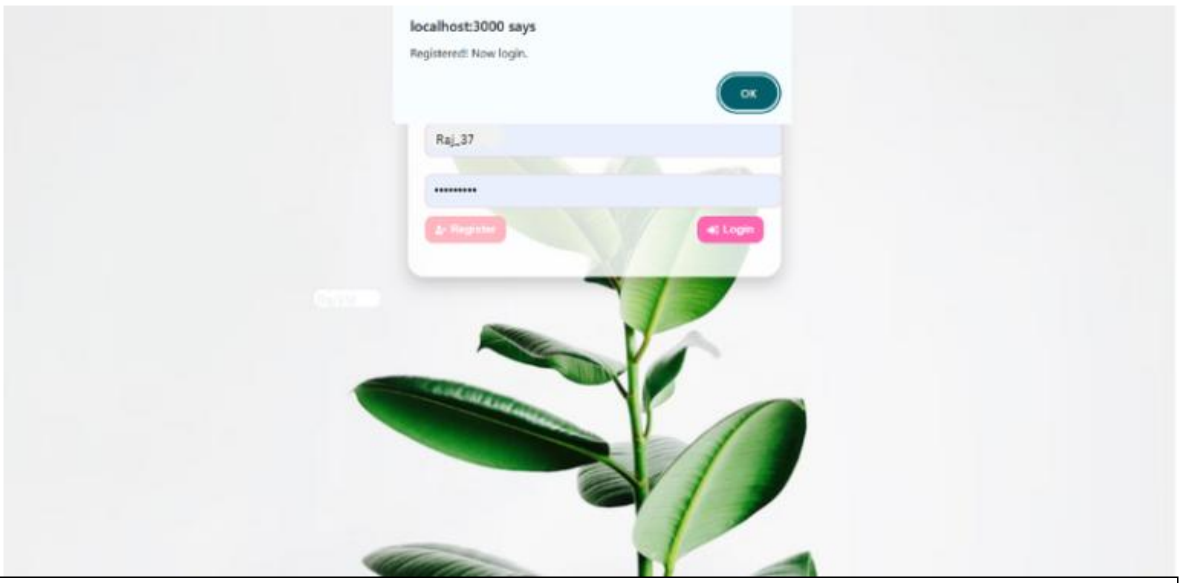


FIGURE 1: SecureNotes Login/ Registration page


localhost:3000 says

Registered! Now login.

FIGURE 2: After registering successfully on the website

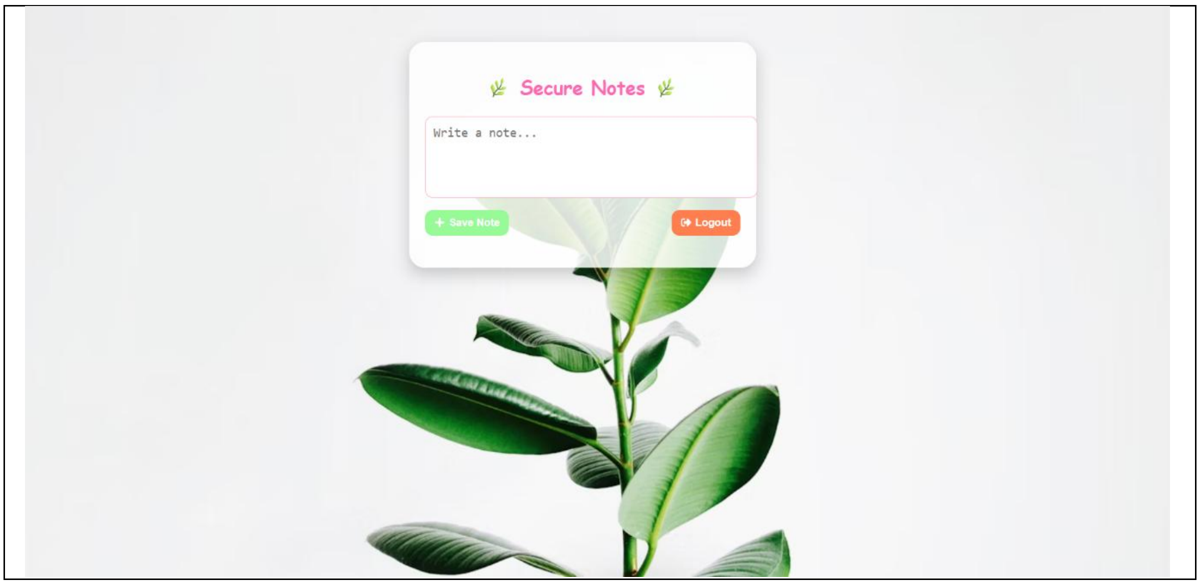
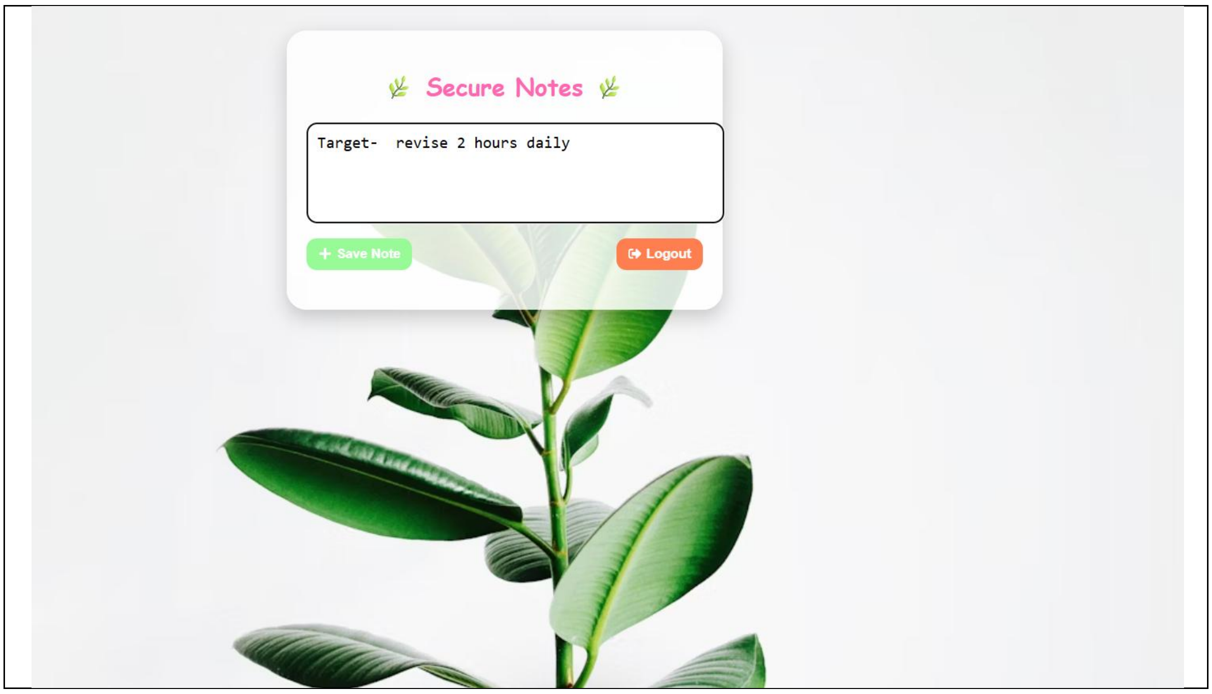
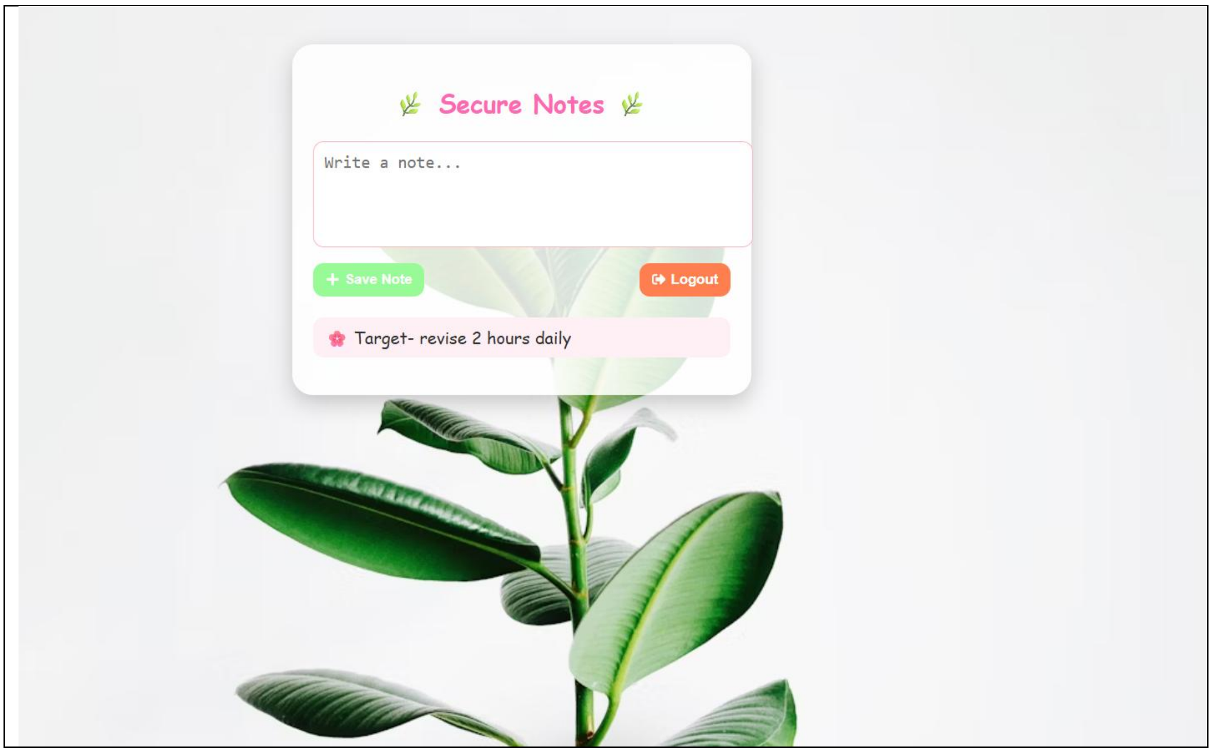


FIGURE 3: After logging in, user can create their secure notes



Target- revise 2 hours daily

FIGURE 4: User types in their notes



Target- revise 2 hours daily

FIGURE 5: After clicking on save note, the notes get logged