```
// App.js
import React from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./Home";
import Login from "./Login";
import Register from "./Register";
import Dashboard from "./Dashboard";
import Explore from "./Explore";
import NotFound from "./NotFound";
import PrivateRoute from "./PrivateRoute";
import Profile from "./Profile";

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/explore" element={<Explore />} />
        <Route path="*" element={<NotFound />} />
        <Route path="/dashboard" element={<PrivateRoute><Dashboard /></PrivateRoute>} />
        <Route path="/explore" element={<PrivateRoute><Explore /></PrivateRoute>} />
        <Route path="/profile" element={<Profile />} />


      </Routes>
    </BrowserRouter>
  );
}
```

```
export default App;




import { createContext, useEffect, useState } from "react";

export const AuthContext = createContext({
  user: null,
  login: () => {},
  logout: () => {},
  loading: true,
});
export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    try {
      const token = localStorage.getItem("token");
      if (token) {
        setUser(token);
      }
    } catch (error) {
      console.error("Failed to access localStorage:", error);
    } finally {
      setLoading(false);
    }
  }, []);

  const login = (token) => {
    localStorage.setItem("token", token);
    const user = parseToken(token); // Example function to decode token
    setUser(user);
  };

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

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

```
import React, { useEffect, useState, useContext } from "react";
import { AuthContext } from "./AuthContext";
import { useNavigate } from "react-router-dom";
import "./Dashboard.css"; // 👈 Import dashboard-specific CSS
import Navbar from "./Navbar";

const Dashboard = () => {
  const [dashboardData, setDashboardData] = useState(null);
  const { logout } = useContext(AuthContext);
  const navigate = useNavigate();
const token = localStorage.getItem("token");

useEffect(() => {
  if (token) {
    fetch("/api/dashboard", {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then((res) => {
        if (!res.ok) throw new Error("Unauthorized");
        return res.json();
      })
      .then((data) => setDashboardData(data))
      .catch((err) => {
        console.error("Error:", err);
        logout();
        navigate("/login");
      });
  }
}, [token, logout, navigate]);

return (
  <>
    <Navbar />
    <div className="dashboard-container">
  <div className="dashboard-card">
  <div className="dashboard-container">
    <div className="dashboard-card">
      {dashboardData ? (
        <>
          <h2>👋 Welcome, {dashboardData.name}!</h2>
          <p><strong>Email:</strong> {dashboardData.email}</p>
```

```
            <p><strong>GitHub:</strong> <a href={dashboardData.github} target="_blank" rel="noopener noreferrer">{dashboardData.github}</a></p>
            <button className="logout-btn" onClick={logout}>Logout</button>
          </>
        ) : (
          <p className="loading-text">Loading or unauthorized...</p>
        )}
        </div>
      </div>
      </div>
      </div>
    </>
  );
};

export default Dashboard;








import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { AuthProvider } from "./AuthContext";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <AuthProvider>
    <App />
  </AuthProvider>
);




import React, { useState, useContext } from "react";
import { AuthContext } from "./AuthContext";
import { useNavigate } from "react-router-dom";
import "./AuthForm.css"; // 👈 Import shared CSS

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const { login } = useContext(AuthContext);
  const navigate = useNavigate();

  const handleLogin = async (e) => {
    e.preventDefault();
    const res = await fetch("/api/auth/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, password }),
    });

    const data = await res.json();

    if (res.ok) {
      localStorage.setItem("token", data.token);
      login(data.token); // ✅ THIS updates the context state
      navigate("/dashboard"); // ✅ This should work now
    } else {
      alert(data.message || "Login failed");
    }
  };


  return (
    <div className="auth-container">
      <form className="auth-form" onSubmit={handleLogin}>
        <h2>Developer Login 🚀</h2>
        <input
          type="email"
          placeholder="you@devmail.com"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <input
          type="password"
          placeholder="••••••••"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
        <button type="submit">Login</button>
        <p>
          Don't have an account? <a href="/register">Register</a>
        </p>
      </form>
    </div>
  );
};

export default Login;
```

```
// src/Navbar.js
import React, { useContext } from "react";
import { Link } from "react-router-dom";
import { AuthContext } from "./AuthContext";
import "./Navbar.css";

const Navbar = () => {
  const { logout } = useContext(AuthContext);

  return (
    <nav className="navbar">
      <div className="navbar-logo">🚀 DevConnect</div>
      <div className="navbar-links">
        <Link to="/dashboard">Dashboard</Link>
        <button onClick={logout}>Logout</button>
      </div>
    </nav>
  );
};
```

```
export default Navbar;
```

```
import React, { useContext } from "react";
import { Navigate } from "react-router-dom";
import { AuthContext } from "./AuthContext";

const PrivateRoute = ({ children }) => {
  const { user, loading } = useContext(AuthContext);

  if (loading) {
    return <div>Loading...</div>; // Or a better loader if you want
  }

  return user ? children : <Navigate to="/login" />;
};

export default PrivateRoute;
```

```
// src/pages/Profile.js
```

```
import React, { useEffect, useState } from "react";
import "./Profile.css"; // 👈 CSS file

const Profile = () => {
  const [profile, setProfile] = useState({
    name: "",
    email: "",
    github: "",
    bio: "",
    skills: "",
  });

  const token = localStorage.getItem("token");

  useEffect(() => {
    fetch("/api/profile", {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then((res) => res.json())
      .then((data) => setProfile(data))
      .catch((err) => console.error("Profile fetch error:", err));
  }, [token]);

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

  const handleSave = async () => {
    try {
      const res = await fetch("/api/profile", {
        method: "PUT",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${token}`,
        },
        body: JSON.stringify(profile),
      });
      if (res.ok) {
        alert("Profile updated!");
      } else {
        alert("Error updating profile.");
      }
    } catch (err) {
      console.error(err);
      alert("Something went wrong.");
    }
  };

  return (
    <div className="profile-container">
      <h2>Developer Profile</h2>
      <input name="name" value={profile.name} onChange={handleChange} placeholder="Name" />
      <input name="email" value={profile.email} onChange={handleChange} placeholder="Email" />
      <input name="github" value={profile.github} onChange={handleChange} placeholder="GitHub
URL" />
      <textarea name="bio" value={profile.bio} onChange={handleChange} placeholder="Bio" rows="3"
/>
      <input name="skills" value={profile.skills} onChange={handleChange} placeholder="Skills (comma
separated)" />
      <button onClick={handleSave}>Save</button>
    </div>
  );
};
```

```
export default Profile;
```

```
import React from "react";
import { Navigate } from "react-router-dom";

const ProtectedRoute = ({ children }) => {
  const token = localStorage.getItem("token");
  return token ? children : <Navigate to="/login" />;
};

export default ProtectedRoute;
```

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import "./AuthForm.css";

const Register = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
const [github, setGithub] = useState("");
const [password, setPassword] = useState("");
const navigate = useNavigate();

const handleRegister = async (e) => {
  e.preventDefault();
  const res = await fetch("/api/auth/register", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      name,
      email,
      password,
      github,
    }),
  });

  const data = await res.json();
  if (res.ok) {
    alert("Registered successfully!");
    navigate("/login");
  } else {
    alert(data.error || "Registration failed");
  }
};


return (
  <div className="auth-container">
    <form className="auth-form" onSubmit={handleRegister}>
      <h2>Developer Register 🛠️</h2>
```

```
<input
  type="text"
  placeholder="Your Full Name"
  value={name}
  onChange={(e) => setName(e.target.value)}
  required
/>
<input
  type="email"
  placeholder="you@devmail.com"
  value={email}
  onChange={(e) => setEmail(e.target.value)}
  required
/>
<input
  type="url"
  placeholder="GitHub Profile URL"
  value={github}
  onChange={(e) => setGithub(e.target.value)}
  required
/>
<input
  type="password"
  placeholder="••••••••"
  value={password}
  onChange={(e) => setPassword(e.target.value)}
  required
/>
<button type="submit">Register</button>
<p>
  Already have an account? <a href="/login">Login</a>
```

```
        </p>
      </form>
    </div>
  );
};

export default Register;





// src/components/UserList.js
import React, { useEffect, useState } from "react";

const UserList = () => {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch("/users") // or "/test-user" based on your route
      .then((res) => res.json())
      .then((data) => setUsers(data))
      .catch((err) => console.error("Error:", err));
  }, []);

  return (
    <div>
      <h2>Registered Users</h2>
      <ul>
        {users.map((user) => (
          <li key={user._id}>
          <strong>{user.name}</strong> - {user.email}
        </li>
      ))}
    </ul>
  </div>
 );
};

export default UserList;
```

```
module.exports = {
  presets: ["@babel/preset-env", "@babel/preset-react"],
 };
```

```
{
 "name": "client",
 "version": "0.1.0",
 "private": true,
"proxy": "http://localhost:5000",
"dependencies": {
  "bcryptjs": "^3.0.2",
  "react": "^19.1.0",
  "react-dom": "^19.1.0",
  "react-router-dom": "^7.5.3",
  "react-scripts": "^5.0.1"
},
"scripts": {
  "start": "react-scripts start",
  "dev": "npm start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"devDependencies": {
  "@babel/core": "^7.22.20",
  "@babel/preset-env": "^7.22.20",
  "@babel/preset-react": "^7.22.5",
  "babel-loader": "^9.1.3",
  "html-webpack-plugin": "^5.6.3",
  "webpack": "^5.88.0",
  "webpack-cli": "^5.1.4",
  "webpack-dev-server": "^4.15.0"
},
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

```
const HtmlWebpackPlugin = require("html-webpack-plugin");
const path = require("path");

module.exports = {
  entry: "./src/index.js",
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
    clean: true,
  },
  devServer: {
    static: "./dist",
    port: 3000,
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: "./public/index.html",
    }),
  ],
  module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        exclude: /node_modules/,
        use: ["babel-loader"],
      },
    ],
  },
  resolve: {
    extensions: [".js", ".jsx"],
  },
};
```

// This Webpack configuration file is set up to bundle a React application. It specifies the entry point, output file, and development server settings. The Babel loader is used to transpile JavaScript and JSX files, excluding the node_modules directory. The configuration also resolves file extensions for easier imports.

// The development server serves static files from the "public" directory and runs on port 3000. This setup is typical for a React application using Webpack for bundling and Babel for transpilation.

```
const mongoose = require('mongoose');


const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log('✅ MongoDB connected');
  } catch (error) {
    console.error('❌ MongoDB connection failed:', error.message);
    process.exit(1);
  }
};

module.exports = connectDB;




const jwt = require("jsonwebtoken");

const authenticateToken = (req, res, next) => {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1]; // Bearer <token>

  if (!token) {
    return res.status(401).json({ message: "No token provided" });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ message: "Invalid token" });
    req.user = user;
    next();
  });
};

module.exports = authenticateToken;
```

```
const mongoose = require("mongoose");

const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    unique: true,
    lowercase: true
  },
  password: {
    type: String,
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

// ✅ Prevent OverwriteModelError
module.exports = mongoose.models.User || mongoose.model("User", UserSchema);
```

```
// server/routes/auth.js
const express = require("express");
const router = express.Router();
const bcrypt = require("bcryptjs");
const User = require("../models/User");
const jwt = require("jsonwebtoken");

// Register User
router.post("/register", async (req, res) => {
  const { name, email, password } = req.body;

  try {
    // Check if user exists
    let user = await User.findOne({ email });
    if (user) return res.status(400).json({ msg: "User already exists" });

    // Hash password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Create and save new user
    user = new User({ name, email, password: hashedPassword });
    await user.save();

    res.status(201).json({ msg: "User registered successfully!" });
  } catch (err) {
    console.error(err);
    res.status(500).send("Server Error");
  }
});

//Login User
```

```
router.post("/login", async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(404).json({ message: "User not found" });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(401).json({ message: "Invalid credentials" });

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "1h" });
    const verified = jwt.verify(token, process.env.JWT_SECRET);

    res.json({ token, user: { id: user._id, name: user.name, email: user.email } });
  } catch (err) {
    res.status(500).json({ message: "Server error" });
  }
 });


module.exports = router;





const express = require("express");
const router = express.Router();
const verifyToken = require("../middleware/verifyToken");

router.get("/", verifyToken, (req, res) => {
  res.json({ message: `Welcome to your dashboard, ${req.user.name}!` });
});

module.exports = router;
```

```
const express = require("express");
const router = express.Router();
const authenticateToken = require("../middleware/authMiddleware"); // adjust path if needed

router.get("/api/dashboard", authenticateToken, (req, res) => {
  res.json({
    message: "Welcome to the protected dashboard!",
    user: req.user,
  });
});

module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const User = require('../models/User');
```

```
// Register new user
router.post('/register', async (req, res) => {
  try {
    const newUser = new User(req.body);
    const saved = await newUser.save();
    res.status(201).json(saved);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Get all users
router.get('/', async (req, res) => {
  try {
    const users = await User.find();
    res.json(users);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

```
const express = require('express');

const mongoose = require('mongoose');

const dotenv = require('dotenv');

const cors = require('cors');

const User = require('./models/User.js');

require('dotenv').config();

const connectDB = require('./config/db.js');

const authRoutes = require("./routes/auth");

const protectedRoutes = require("./routes/protected");



dotenv.config();


const app = express();

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


// Middleware

app.use(cors());

app.use(express.json());


// MongoDB connection

mongoose.connect(process.env.MONGO_URI, {

  useNewUrlParser: true,

  useUnifiedTopology: true

})

.then(() => console.log('✅ MongoDB connected'))

.catch((err) => console.log('❌ DB connection error:', err));


//Routes

app.use("/api/auth", authRoutes);
```

```
// Test route
app.get('/', (req, res) => {
  res.send('🚀 Backend is running!');
});

app.use("/", protectedRoutes);

// Optional: Create route to test model
app.post("/api/register", async (req, res) => {
  const { name, email, password, github } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ name, email, password: hashedPassword, github });
    await user.save();
    res.status(201).json({ message: "User registered" });
  } catch (err) {
    res.status(400).json({ error: "User already exists or error" });
  }
});



// Route to get all users
app.get('/users', async (req, res) => {
  try {
    const users = await User.find(); // Fetches all users
    res.status(200).json(users);     // Sends them as JSON
  } catch (err) {
    res.status(500).json({ error: 'Server error' });
  }
});
```

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

Server side package.js

```
{
  "name": "devconnect",
  "version": "1.0.0",
  "main": "index.js",
  "proxy": "http://localhost:5000",
  "scripts": {
    "client": "cd client && npm start",
    "server": "cd server && npm run dev",
    "dev": "concurrently \"npm run server\" \"npm run client\""
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "concurrently": "^9.1.2"
  },
  "dependencies": {
    "bcryptjs": "^3.0.2",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.5.3"
  }
}
```