Fullstack Development

Authentication / Authorization

Part 4: SPA

Setup

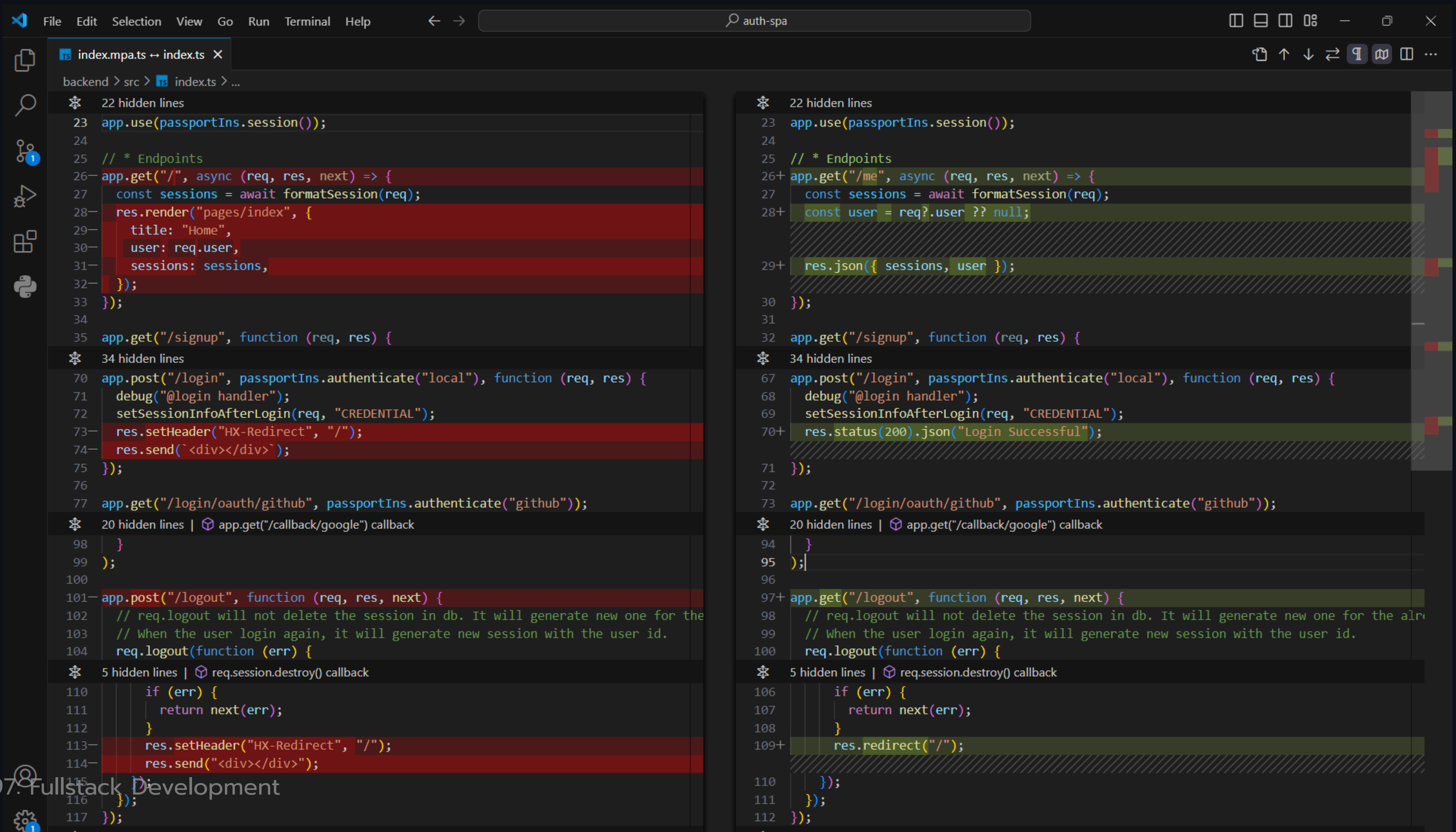
- `git clone https://github.com/fullstack-68/auth-spa.git`
- Backend
  - `cd backend`
  - Fill in `.env`
  - `pnpm install`
  - `pnpm run db:reset`
  - `pnpm run dev`

Setup

- Frontend
  - `cd frontend`
  - `pnpm install`
  - `pnpm run dev`

Backend

- Routes now return json / redirect header instead of html.
- Created `GET /me` route for clients to check their `auth`' states.
- Apart from that, there is very mininal change, surprisingly.



```
22 hidden lines
app.use(passportIns.session());

// * Endpoints
app.get("/", async (req, res, next) => {
  const sessions = await formatSession(req);
  res.render("pages/index", {
    title: "Home",
    user: req.user,
    sessions: sessions,
  });
});

app.get("/signup", function (req, res) {
34 hidden lines
app.post("/login", passportIns.authenticate("local"), function (req, res) {
  debug("@login handler");
  setSessionInfoAfterLogin(req, "CREDENTIAL");
  res.setHeader("HX-Redirect", "/");
  res.send(`<div></div>`);
});

app.get("/login/oauth/github", passportIns.authenticate("github"));
20 hidden lines | app.get("/callback/google") callback
  }
);

app.post("/logout", function (req, res, next) {
  // req.logout will not delete the session in db. It will generate new one for the
  // When the user login again, it will generate new session with the user id.
  req.logout(function (err) {
5 hidden lines | req.session.destroy() callback
      if (err) {
        return next(err);
      }
      res.setHeader("HX-Redirect", "/");
      res.send("<div></div>");
    });
  });
});
```

```
22 hidden lines
app.use(passportIns.session());

// * Endpoints
app.get("/me", async (req, res, next) => {
  const sessions = await formatSession(req);
  const user = req?.user ?? null;
  res.json({ sessions, user });
});

app.get("/signup", function (req, res) {
34 hidden lines
app.post("/login", passportIns.authenticate("local"), function (req, res) {
  debug("@login handler");
  setSessionInfoAfterLogin(req, "CREDENTIAL");
  res.status(200).json("Login Successful");
});

app.get("/login/oauth/github", passportIns.authenticate("github"));
20 hidden lines | app.get("/callback/google") callback
  }
);

app.get("/logout", function (req, res, next) {
  // req.logout will not delete the session in db. It will generate new one for the alr
  // When the user login again, it will generate new session with the user id.
  req.logout(function (err) {
5 hidden lines | req.session.destroy() callback
      if (err) {
        return next(err);
      }
      res.redirect("/");
    });
  });
});
```

Frontend

- Created (client-side) routing.
- Created logic to query/update `auth`'s state.
- Modified proxy server to take care of `Callback URL` (bypass client-routing).
- Created signup form/login and other UI.

Highlighted packages

```json
{
  "@tanstack/react-query": "^5.85.5",
  "react-router": "^7.8.2"
}
```

Client-side routing

`src/App.tsx`

```tsx
import { RouterProvider } from "react-router/dom";
import { createBrowserRouter } from "react-router";
const router = createBrowserRouter([
  {
    path: "/",
    element: <Layout />,
    children: [
      {
        path: "/",
        element: <Home />,
      },
    ],
  },
]);
```

Client-side routing

`src/App.tsx`

```tsx
function App() {
  return (
    // ...
    <RouterProvider router={router} />
    // ...
  );
}

export default App;
```

Getting auth state

`src/hooks/useAuth.ts`

```ts
import { useQuery } from "@tanstack/react-query";
// ...
function getMe() {
  return axios.get<AuthData>("/api/me");
}

function useAuth() {
  // Queries
  const { data, error, refetch } = useQuery({
    queryFn: getMe,
    // Other options
  });
  return { user: data?.user, sessions: data?.sessions, error, refetch };
}
```

Getting auth state

`src/components/Nav.tsx`

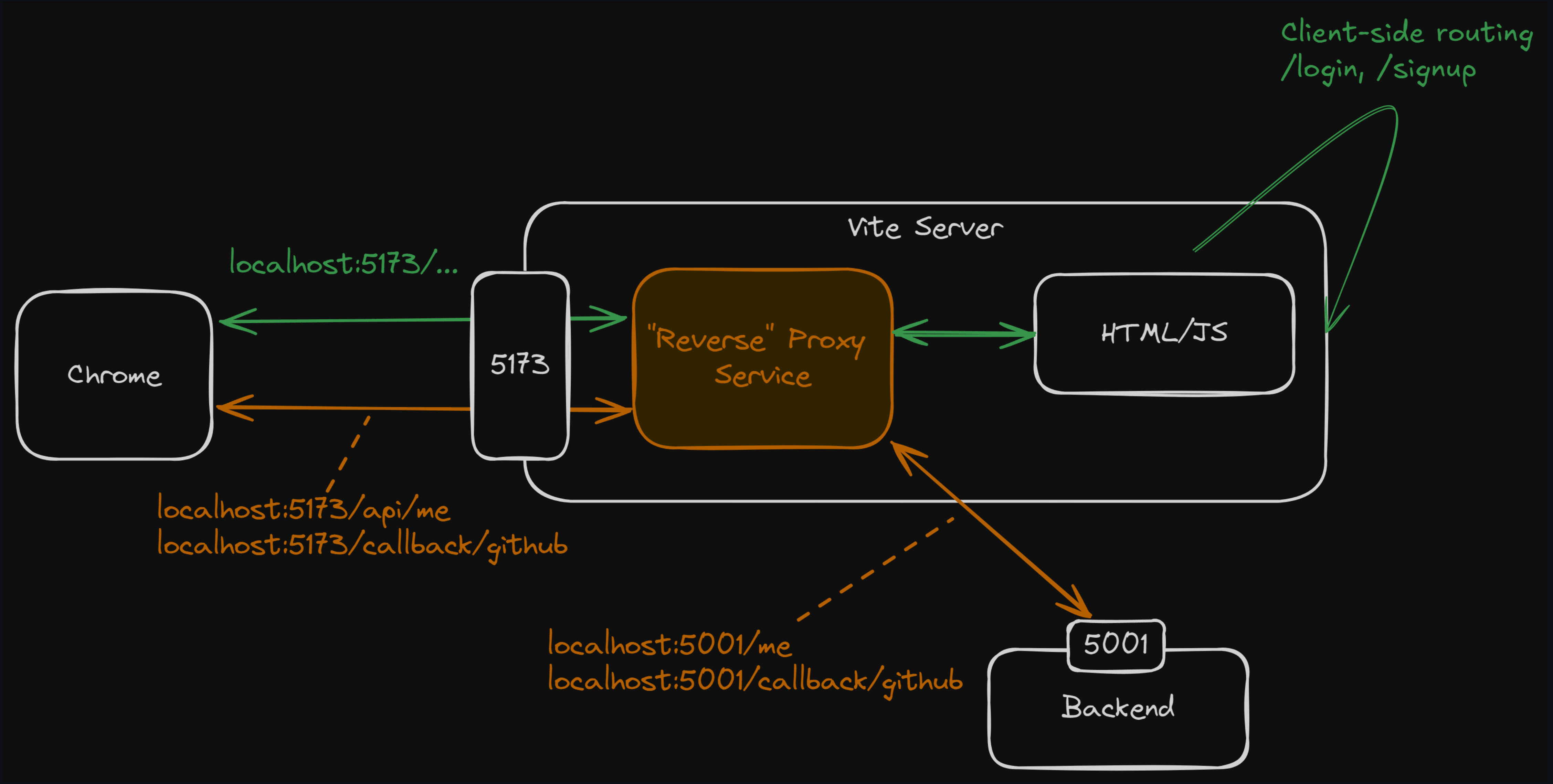
```
const Nav: FC = () => {
  const { user } = useAuth(); 👈👈👈
  // ...
  return (
    <nav>
      // ...
    </nav>
  );
};

export default Nav;
```

Handle `Callback URL`

`vite.config.ts`

```
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "/api": {
        // ...
      },
      "/callback": { 👉👉👉
        target: "http://localhost:5002",
      },
    },
  },
});
```


Client-side routing
/login, /signup
Vite Server
localhost:5173/...
Chrome
5173
"Reverse" Proxy Service
HTML/JS
localhost:5173/api/me
localhost:5173/callback/github
localhost:5001/me
localhost:5001/callback/github
5001
Backend

With proxy server

- The cookie will automatically sent to backend for all requests because it is the "same site".

If you don't have proxy server.

- Cookies are still automatically sent given the same host but different port.
- Need to set `withCredentials` to `true` on any AJAX request (via `fetch` or `axios` APIs)
    - Note that `httpOnly` cookie are sent by this technique.
- Need to allow CORS in the backend.

But we still need more

- More OAuth providers
- Forget password feature
- Email confirmation
- OTP link
- Two factor authentication
- Passkey
- ...

> When can I start building my stuff?

Third-party solution

- *On cloud*
  - Clerk
- *On premise*
  - AuthJS
    - I used this last year
  - Better-Auth

`Better-Auth`

Design

- `Better-Auth` (and `AuthJS`) **is not a middleware**.
  - Unlike `passport`
- It provides ready-made API endpoints.
- This architecture is more suitable for SPAs and fullstack frameworks (`NextJS`).
  - *Using it with MPA is quite unintuitive.*

Setup

- `git clone https://github.com/fullstack-68/auth-spa-better-auth.git`
- Backend
  - `cd backend`
  - Fill in `.env`
  - `pnpm install`
  - `pnpm run db:reset`
  - `pnpm run dev`

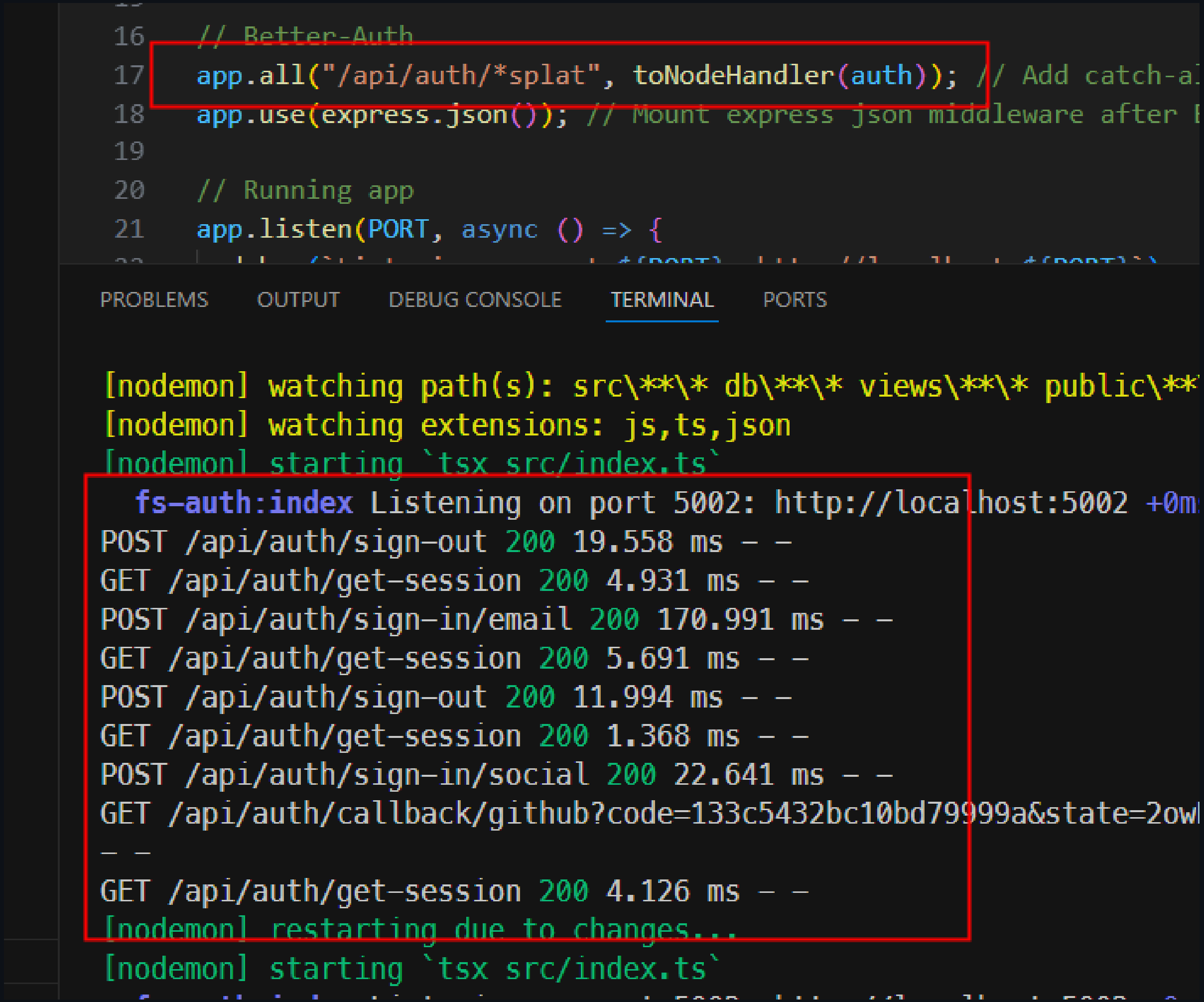
Setup

- Frontend
  - `cd frontend`
  - `pnpm install`
  - `pnpm run dev`

Note

- DB schema
- `npx @better-auth/cli@latest generate`

New routes (backend)



```
// Better-Auth
app.all("/api/auth/*splat", toNodeHandler(auth)); // Add catch-al
app.use(express.json()); // Mount express json middleware after

// Running app
app.listen(PORT, async () => {
```

```
[nodemon] watching path(s): src\**\* db\**\* views\**\* public\**
[nodemon] watching extensions: js,ts,json
[nodemon] starting `tsx src/index.ts`
  fs-auth:index Listening on port 5002: http://localhost:5002 +0m
POST /api/auth/sign-out 200 19.558 ms - -
GET /api/auth/get-session 200 4.931 ms - -
POST /api/auth/sign-in/email 200 170.991 ms - -
GET /api/auth/get-session 200 5.691 ms - -
POST /api/auth/sign-out 200 11.994 ms - -
GET /api/auth/get-session 200 1.368 ms - -
POST /api/auth/sign-in/social 200 22.641 ms - -
GET /api/auth/callback/github?code=133c5432bc10bd79999a&state=2ow
- -
GET /api/auth/get-session 200 4.126 ms - -
[nodemon] restarting due to changes...
[nodemon] starting `tsx src/index.ts`
```

Frontend client

`useAuth.ts`

```ts
import { useQuery } from "@tanstack/react-query";
import { authClient } from "../lib/auth-client";
function getMe() {
  return authClient.getSession();
}
```

Sign up

Signup.tsx

```
const res = await authClient.signUp.email({
  email,
  password,
  name,
  image: "logos/robot.png",
});
```

Login (email)

```
const res = await authClient.signIn.email({
  email,
  password,
  rememberMe: true,
});
```

Login (Social)

```
const signInGitHub = async () => {
  const data = await authClient.signIn.social({
    provider: "github",
  });
  console.log({ data });
};
```

More features

- Forget password
- Email confirmation
- OTP link
- Two factor