React Remix demo project where we:

- Create a simple **Remix app**
- Use **nested routes**
- Fetch data using **loader functions**
- Use **Tailwind CSS**
- Show a simple **quote generator** with buttons

1. Create the Remix App

```
npx create-remix@latest remix-demo
```

Choose:

- **Remix App Server** (or Express if you're deploying somewhere specific)
- **JavaScript or TypeScript** (I'll assume TypeScript)
- **Tailwind CSS** – Yes

Then:

```
cd remix-demo
npm install
npm run dev
```

You now have a working Remix app at `http://localhost:3000`.

3. App Structure Overview

```
remix-demo/
├── app/
│   ├── routes/
│   │   ├── index.tsx       <- Homepage
│   │   └── quotes.tsx      <- Quotes page
│   ├── root.tsx            <- Layout + tailwind
│   └── styles/tailwind.css
├── remix.config.js
```

4. Add Tailwind Styling (already scaffolded)

Inside `app/styles/tailwind.css`:

```
css

@tailwind base;
@tailwind components;
@tailwind utilities;
```

Ensure it's imported in `app/root.tsx`:

```
import styles from './styles/tailwind.css';
export function links() {
  return [{ rel: 'stylesheet', href: styles }];
}
```

5. Add `quotes.tsx` Page

In `app/routes/quotes.tsx`:

```
import { json, LoaderFunction } from '@remix-run/node';
import { useLoaderData } from '@remix-run/react';

const quotes = [
  "The only way to do great work is to love what you do. —
Steve Jobs",
  "Life is what happens when you're busy making other
plans. — John Lennon",
  "You miss 100% of the shots you don't take. — Wayne
Gretzky"
];

export const loader: LoaderFunction = () => {
  const random = quotes[Math.floor(Math.random() *
quotes.length)];
  return json({ quote: random });
};

export default function QuotesPage() {
  const data = useLoaderData<typeof loader>();
```

```
  return (
    <div className="p-10 text-center">
      <h1 className="text-2xl font-bold mb-4">Random
Quote</h1>
      <blockquote className="italic text-lg">{data.quote}</
blockquote>
      <form method="get" className="mt-6">
        <button
          type="submit"
          className="bg-blue-600 text-white px-4 py-2
rounded hover:bg-blue-800"
        >
          New Quote
        </button>
      </form>
    </div>
  );
}
```

6. Add Link on Homepage

Update `app/routes/index.tsx`:

```
import { Link } from '@remix-run/react';

export default function Index() {
  return (
    <div className="p-10 text-center">
      <h1 className="text-3xl font-bold mb-4">Welcome to
Remix!</h1>
      <Link
        to="/quotes"
        className="text-blue-600 underline hover:text-
blue-800"
      >
        View a random quote →
      </Link>
    </div>
```

```
  );
}
```

7. Run the App

```
npm run dev
```

- Go to `http://localhost:3000`
- Click **“View a random quote”**
- Each reload of the `/quotes` route shows a new quote (handled server-side via the loader)

Why Remix?

- **Loaders** give you **server-side data fetching**
- **No need for useEffect** for fetching
- Great for **SEO** and performance
- Built for **progressive enhancement**

Add authentication to your Remix app using sessions and a login form, all done the "Remix way" — no client-side libraries like Firebase, just secure server-side logic.

1. Create a login form
2. Authenticate against hardcoded user data (or fake DB)
3. Use session cookies for login/logout
4. Protect a route

1. Install Required Packages

```
npm install @remix-run/node @remix-run/react bcryptjs
cookie
```

2. Set Up Sessions

In `app/utils/session.server.ts`:

```
import { createCookieSessionStorage, redirect } from
"@remix-run/node";

const sessionSecret = process.env.SESSION_SECRET ||
"supersecret";

export const sessionStorage = createCookieSessionStorage({
  cookie: {
    name: "__session",
    secrets: [sessionSecret],
    sameSite: "lax",
    path: "/",
    httpOnly: true,
    secure: process.env.NODE_ENV === "production",
  },
});

export async function getUserSession(request: Request) {
  return
sessionStorage.getSession(request.headers.get("Cookie"));
}

export async function getUserId(request: Request) {
  const session = await getUserSession(request);
  return session.get("userId");
}

export async function requireUserId(request: Request) {
  const userId = await getUserId(request);
  if (!userId) throw redirect("/login");
  return userId;
}

export async function createUserSession(userId: string,
redirectTo: string) {
```

```
  const session = await sessionStorage.getSession();
  session.set("userId", userId);
  return redirect(redirectTo, {
    headers: {
      "Set-Cookie": await
sessionStorage.commitSession(session),
    },
  });
}

export async function logout(request: Request) {
  const session = await getUserSession(request);
  return redirect("/", {
    headers: {
      "Set-Cookie": await
sessionStorage.destroySession(session),
    },
  });
}
```

3. Create a Fake DB

In `app/utils/users.server.ts`:

```
ts

import bcrypt from "bcryptjs";

const users = [
  {
    id: "1",
    username: "admin",
    // bcrypt hash of "password"
    passwordHash: bcrypt.hashSync("password", 10),
  },
];

export async function authenticate(username: string,
password: string) {
```

```
  const user = users.find((u) => u.username === username);
  if (!user) return null;

  const isValid = await bcrypt.compare(password,
user.passwordHash);
  return isValid ? user : null;
}
```

4. Create Login Route

In `app/routes/login.tsx`:

```
import { json, redirect } from "@remix-run/node";
import { Form, useActionData } from "@remix-run/react";
import { authenticate } from "~/utils/users.server";
import { createUserSession } from "~/utils/session.server";

export async function action({ request }: { request:
Request }) {
  const form = await request.formData();
  const username = form.get("username") as string;
  const password = form.get("password") as string;

  const user = await authenticate(username, password);
  if (!user) {
    return json({ error: "Invalid credentials" }, { status:
400 });
  }

  return createUserSession(user.id, "/protected");
}

export default function Login() {
  const actionData = useActionData<typeof action>();

  return (
    <div className="p-10 max-w-md mx-auto">
      <h1 className="text-xl font-bold mb-4">Login</h1>
      <Form method="post">
```

```
        <input name="username" placeholder="Username"
className="border p-2 w-full mb-3" />
        <input name="password" type="password"
placeholder="Password" className="border p-2 w-full mb-3" /
>
        {actionData?.error && <p className="text-
red-500">{actionData.error}</p>}
        <button className="bg-blue-600 text-white px-4 py-2
rounded">Login</button>
      </Form>
    </div>
  );
}
```

5. Create a Protected Route

In `app/routes/protected.tsx`:

```
import { LoaderFunction } from "@remix-run/node";
import { requireUserId } from "~/utils/session.server";

export const loader: LoaderFunction = async ({ request })
=> {
  await requireUserId(request);
  return null;
};

export default function ProtectedPage() {
  return (
    <div className="p-10 text-center">
      <h1 className="text-2xl font-bold">Welcome to the
Protected Page!</h1>
      <form method="post" action="/logout"
className="mt-4">
        <button className="text-blue-600
underline">Logout</button>
      </form>
    </div>
  );
```

```
}
```

6. Add Logout Route

In `app/routes/logout.tsx`:

```
import { ActionFunction } from "@remix-run/node";
import { logout } from "~/utils/session.server";

export const action: ActionFunction = async ({ request })
=> {
  return logout(request);
};
```

To test:

1. Go to `/login`
2. Use: `admin` / `password`
3. You should be redirected to `/protected`
4. Try going to `/protected` without logging in → redirected to login