CMPS 350 · Web Development Fundamentals

Lab 13 · Authentication and Server Actions[1]

Objective

- Using Next server actions/mutations, router, and headers
- Verifying and signing JSON Web Tokens (JWT)
- Authenticating using a one-time-password or an OAuth provider

Resources

- Server Actions: https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions
- Next environment variables: https://nextjs.org/docs/app/building-your-application/configuring/environment-variables
- Next headers: https://nextjs.org/docs/app/api-reference/functions/headers
- JSON Web Tokens (JWT): https://jwt.io
- Magic Auth: https://magic.link/auth
  - Hello world: https://magic.link/posts/hello-world
  - Magic JWT: https://magic.link/posts/magic-jwt

1. Server Actions

1. Create a Next application using: `npx create-next-app@latest .`
2. Update the configuration file, `next.config.js`, to enable (experimental) server actions:

   ```
   const nextConfig = { experimental: { serverActions: true } };
   ```
3. Create a Prisma schema for managing users and their ideas:
   3.1. A user has an email address, a required creation date, a registration date, and a list of ideas.
   3.2. An idea has a required title, a list of (predefined) tags, and a required creation date.
4. Create a page that allows a user to manage their ideas:
   4.1. Use server actions to retrieve, create, and delete ideas.
   4.2. Use local storage to identify a user using their model id.
   4.3. Do not store the ideas in local storage.
5. Create a data repository service to manage users and their ideas. A user should be able to retrieve their ideas, create a new idea, and delete an existing idea.
6. Create an API with multiple endpoints and methods that builds on top of the data repository service methods: `/api/users/[user]/ideas/[idea]`.
7. Test the application using multiple users. You can use a private/incognito tab for each user.

[1] Experimental alpha feature.

2. Magic Authentication

1. Register and create a Magic Auth application at https://magic.link:
   1.1. Install the Magic SDK packages and use them on the client and server side, respectively: `npm install magic-sdk @magic-sdk/admin`.
   1.2. Store the publishable key as an environment variable, accessible on the client side, `NEXT_PUBLIC_MAGIC_KEY`.
   1.3. Store the secret key as an environment variable, accessible only on the server side, `MAGIC_SECRET_KEY`.
   1.4. Store the keys in `.env.local` instead of `.env`.
2. Create a component that allows the user to login and logout using Magic Auth's email one-time password:
   2.1. Update the user's email address after a successful authentication.
   2.2. Use the token, after storing it in local storage, to authenticate and identify the users. The headers must be set and used correctly, both on the client and server side.
3. Add GitHub as a provider and update the page/schema/endpoints to display a user's name and picture.