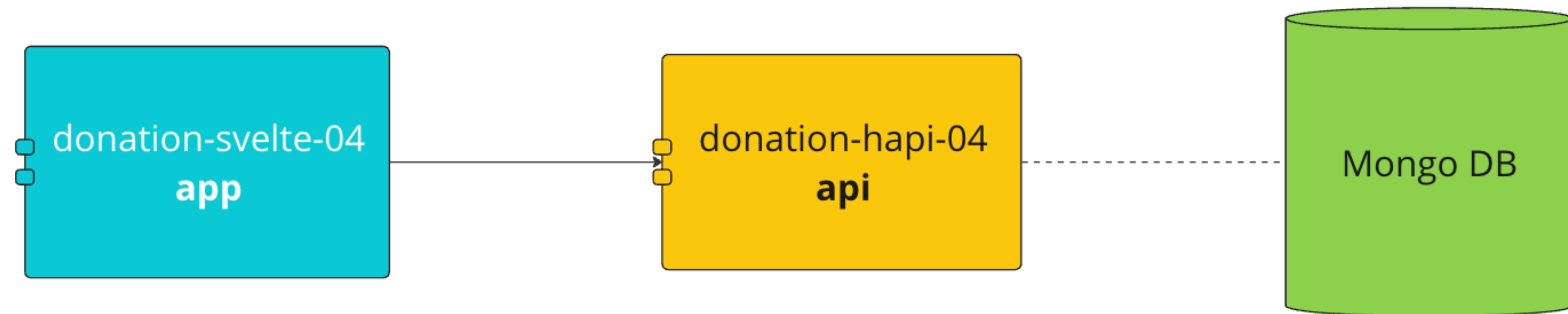# Svelte Authentication

Incorporate authentication into Svelte application
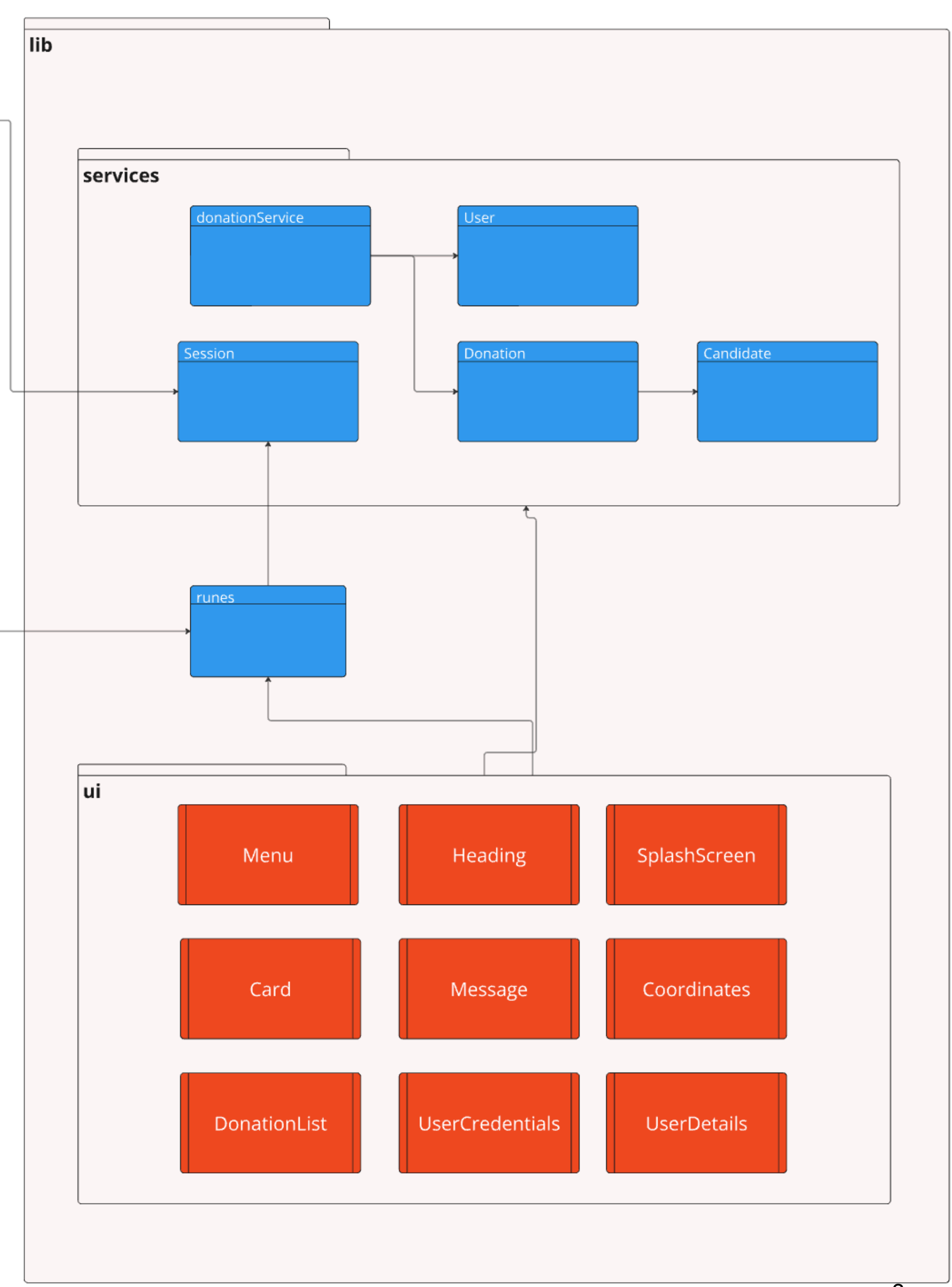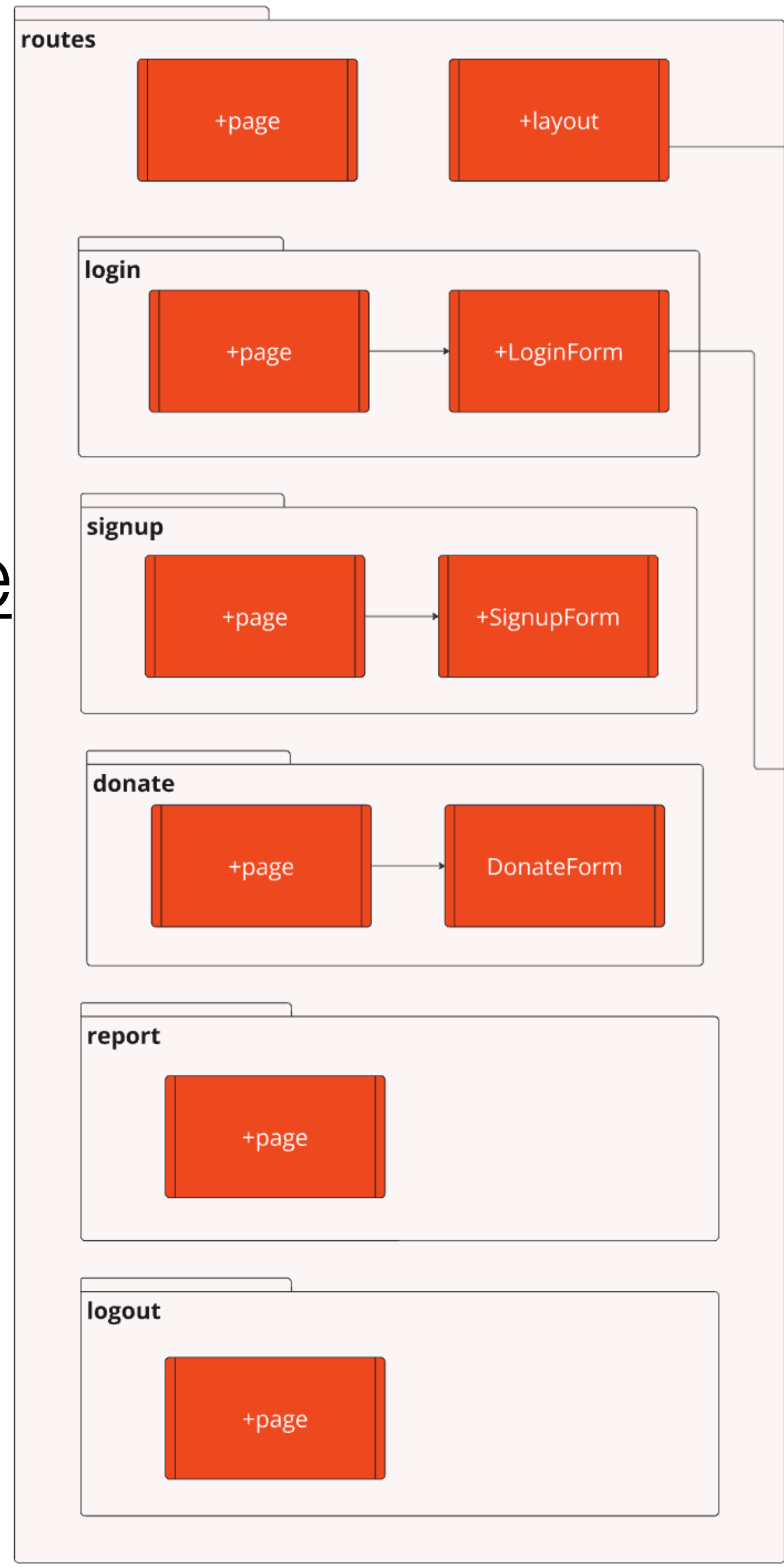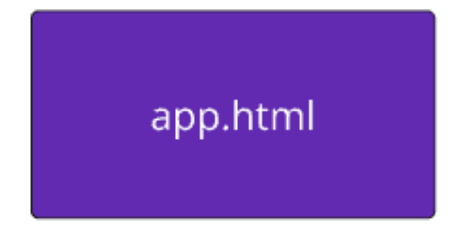
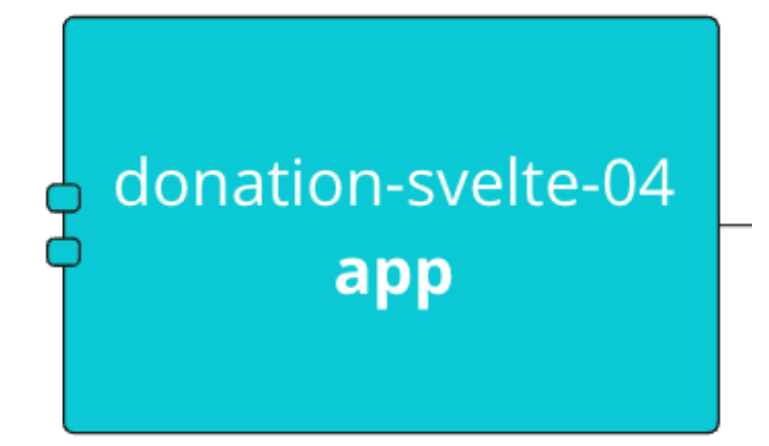# Svelte Authentication

- Attach to the Donation-svelte API

- Signup / Login using Users API

Architecture

Donation-types

```
src
  lib
    services
      TS donation-service.ts
    types
      TS donation-types.ts
    ui
    TS index.ts
    TS runes.svelte.ts
  routes
    donate
    login
    logout
    report
    signup
    +layout.svelte
    +page.svelte
  TS app.d.ts
  app.html
```
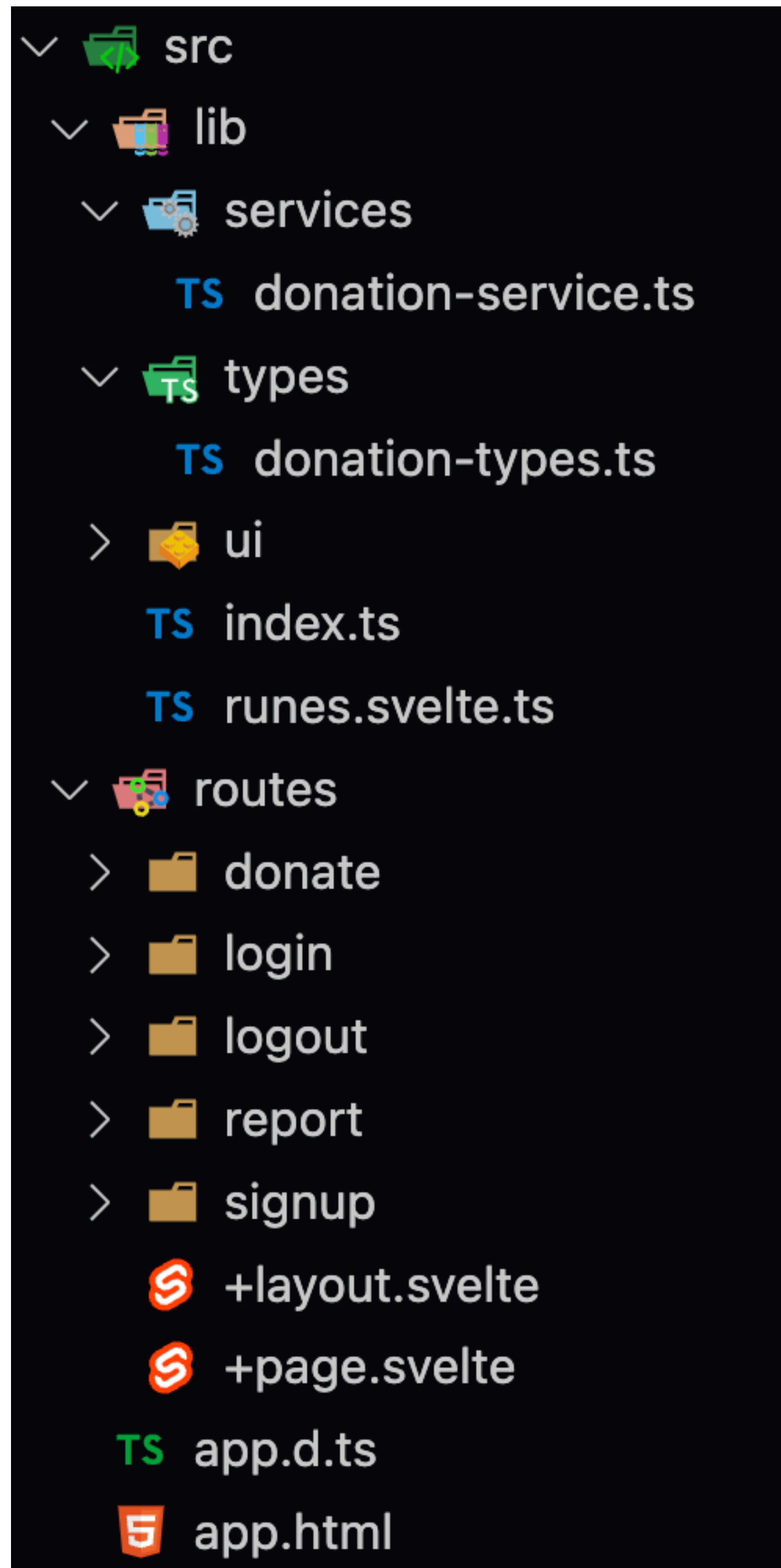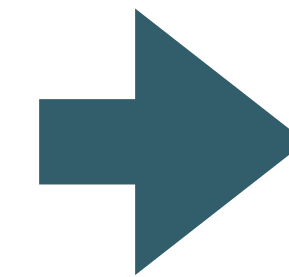
Session: logged in user name, id + JWT token
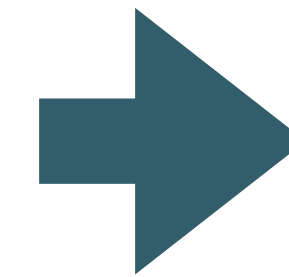
User: user details

```typescript
export interface Session {
  name: string;
  _id: string;
  token: string;
}

export interface User {
  firstName: string;
  lastName: string;
  email: string;
  password: string;
  _id: string;
}
```

donationService

- Encapsulate

- baseUrl defines url of remote donation service

- login: invoke authenticate endpoint, return session object if successful

- signup: invoke signup up API.

```typescript
import axios from "axios";
import type {  Session, User } from "./donation-types";

export const donationService = {
  baseUrl: "http://localhost:4000",

  async signup(user: User): Promise<boolean> {
    try {
      const response = await axios.post(`${this.baseUrl}/api/users`, user);
      return response.data.success === true;
    } catch (error) {
      console.log(error);
      return false;
    }
  },

  async login(email: string, password: string): Promise<Session | null> {
    try {
      const response = await axios.post(`${this.baseUrl}/api/users/authenticate`, { email, password });
      if (response.data.success) {
        axios.defaults.headers.common["Authorization"] = "Bearer " + response.data.token;
        const session: Session = {
          name: response.data.name,
          token: response.data.token,
          _id: response.data.id
        };
        return session;
      }
      return null;
    } catch (error) {
      console.log(error);
      return null;
    }
  },
}
```

Login

Login to DONATION

Email

✉ homer@simpson.com

Password

🔑 ••••••

Log In

```ts
<script lang="ts">
  import { goto } from '$app/navigation';
  import { loggedInUser } from '$lib/runes.svelte';
  import Message from '$lib/ui/Message.svelte';
  import UserCredentials from '$lib/ui/UserCredentials.svelte';

  let email = $state('');
  let password = $state('');
  let message = $state('');

  async function login() {
    const success = true;
    if (success) {
      loggedInUser.email = email;
      goto('/donate');
    } else {
      email = '';
      password = '';
      message = 'Invalid Credentials';
    }
  }
</script>
```

- Current version "goto" donate route when login button pressed

Login

Login to DONATION

Email

✉ homer@simpson.com

Password

🔑 ••••••

Log In

```ts
<script lang="ts">
  import { goto } from "$app/navigation";
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Message from "$lib/ui/Message.svelte";
  import UserCredentials from "$lib/ui/UserCredentials.svelte";

  let email = $state("");
  let password = $state("");
  let message = $state("");

  async function login() {
    console.log(`attempting to log in email: ${email} with password: ${password}`);
    let session = await donationService.login(email, password);
    if (session) {
      loggedInUser.email = email;
      loggedInUser.name = session.name;
      loggedInUser.token = session.token;
      loggedInUser._id = session._id;
      console.log(`Session: ${JSON.stringify(session)}`);
      goto("/donate");
    } else {
      email = "";
      password = "";
      message = "Invalid Credentials";
    }
  }
}
</script>
```

- Revised version to authenticate using Donation-hapi API

Import a references to the donationService object.

When login pressed, attempt login

If successful doing, display **Donate** page

Otherwise, reset email/password to bank + display error message

```ts
<script lang="ts">
  import { goto } from "$app/navigation";
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Message from "$lib/ui/Message.svelte";
  import UserCredentials from "$lib/ui/UserCredentials.svelte";

  let email = $state("");
  let password = $state("");
  let message = $state("");

  async function login() {
    console.log(`attempting to log in email: ${email} with password: ${password}`);
    let session = await donationService.login(email, password);
    if (session) {
      loggedInUser.email = email;
      loggedInUser.name = session.name;
      loggedInUser.token = session.token;
      loggedInUser._id = session._id;
      console.log(`Session: ${JSON.stringify(session)}`);
      goto("/donate");
    } else {
      email = "";
      password = "";
      message = "Invalid Credentials";
    }
  }
</script>
```

Signup for DONATION

Name

[Enter fi] [Enter la]

Email

[Enter email]

Password

[Enter Password]

Create Account

Import a reference to the donationService object.

Signup up new user

If successful doing, display **Main** page

```
let firstName = "";
let lastName = "";
let email = "";
let password = "";
let message = "";

async function signup() {
  const user: User = {
    firstName: firstName,
    lastName: lastName,
    email: email,
    password: password
  };
  let success = await donationService.signup(user);
  if (success) {
    goto("/login");
  } else {
    message = "Error Trying to sign up";
  }
}
```

Logged in User Indicator

♥ 🔲 Donation　　　　　　　　　　　Donate　　Report　　Logout [Homer Simpson]

- When user logged in…　　　… display user name in Menu bar

Logged in User Indicator

Donation   Donate   Report   Logout [Homer Simpson]

- When user logged in…   … display user name in Menu bar

```ts
export interface Session {
  name: string;
  _id: string;
  token: string;
}
```

```ts
export const loggedInUser = $state({
  email: "",
  name: "",
  token: "",
  _id: ""
});
```

- donationService creates the session object

- LoginForm writes the session info to the rune

```svelte
<script lang="ts">
  import { loggedInUser } from "$lib/runes.svelte";
</script>

<nav class="navbar is-full-width">
  <div class="container">
    <div class="navbar-brand">
      <a class="navbar-item" href="/dashboard">
        <span class="icon"> <i class="fas fa-map-marker-alt"></i></span> <span class="icon mr-1">
          <i class="far fa-money-bill-alt"></i></span>
        ><span><strong>Donation</strong> </span>
      </a>
    </div>
    <div id="navbarMenu" class="navbar-menu">
      <div class="navbar-end">
        <a class="navbar-item" href="/donate"> Donate </a>
        <a class="navbar-item" href="/report"> Report </a>
        <a class="navbar-item" href="/logout"> Logout [{loggedInUser.name}]</a>
      </div>
      <div></div>
    </div>
  </div>
</nav>
```

Logged in User Indicator

- ## When user logged in…

- ## … display user name in Menu bar

```ts
export const loggedInUser = $state({
    email: "",
    name: "",
    token: "",
    _id: ""
});
```

- donationService create the session object

- LoginForm writes session information to the rune
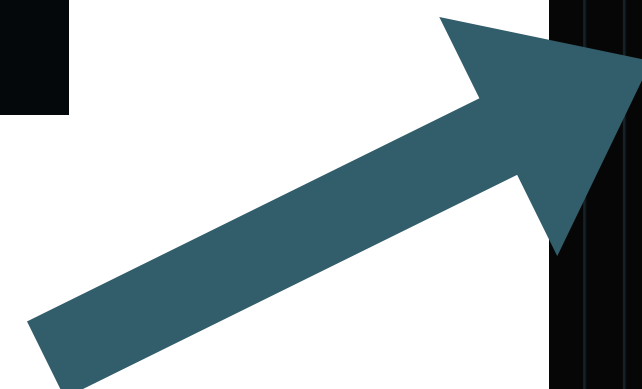
```ts
<script lang="ts">
  import { goto } from "$app/navigation";
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Message from "$lib/ui/Message.svelte";
  import UserCredentials from "$lib/ui/UserCredentials.svelte";

  let email = $state("");
  let password = $state("");
  let message = $state("");

  async function login() {
    console.log(`attempting to log in email: ${email} with password: ${password}`);
    let session = await donationService.login(email, password);
    if (session) {
      loggedInUser.email = email;
      loggedInUser.name = session.name;
      loggedInUser.token = session.token;
      loggedInUser._id = session._id;
      console.log(`Session: ${JSON.stringify(session)}`);
      goto("/donate");
    } else {
      email = "";
      password = "";
      message = "Invalid Credentials";
    }
  }
</script>

<div class="box">
  {#if message}
    <Message {message} />
  {/if}
  <UserCredentials bind:email bind:password />
  <button onclick={() => login()} class="button">Log In</button>
</div>
```

12

Svelte Authentication

Incorporate authentication into Svelte application