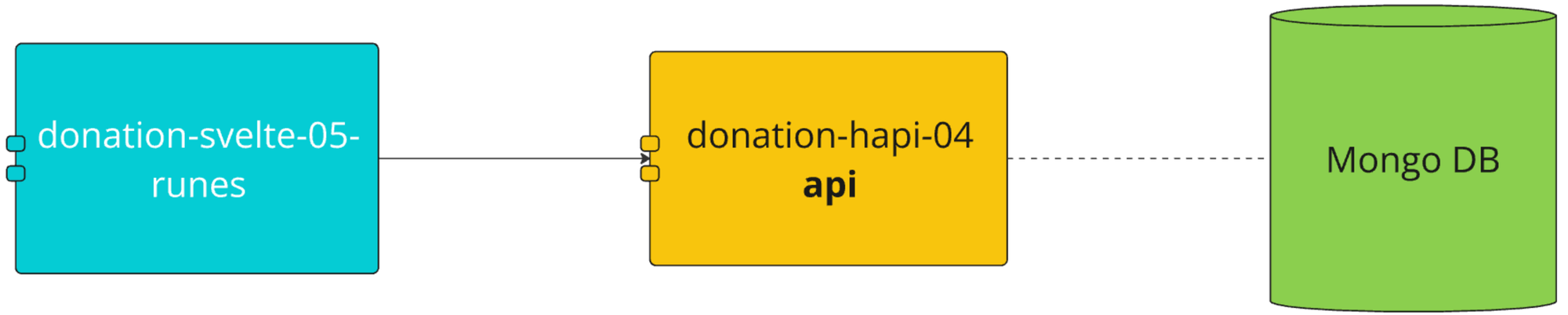
SvelteKit Sessions

Preserve the SvelteKit sessions between page reloads

Svelte Authentication



- Session/jwt token management
- State synchronisation across client & server

LoginForm.svelte

```
async function login() {
  console.log(`attempting to log in email: ${email} with password: ${password}`);
  let session = await donationService.login(email, password);
  if (session) {
```

donation-service.ts

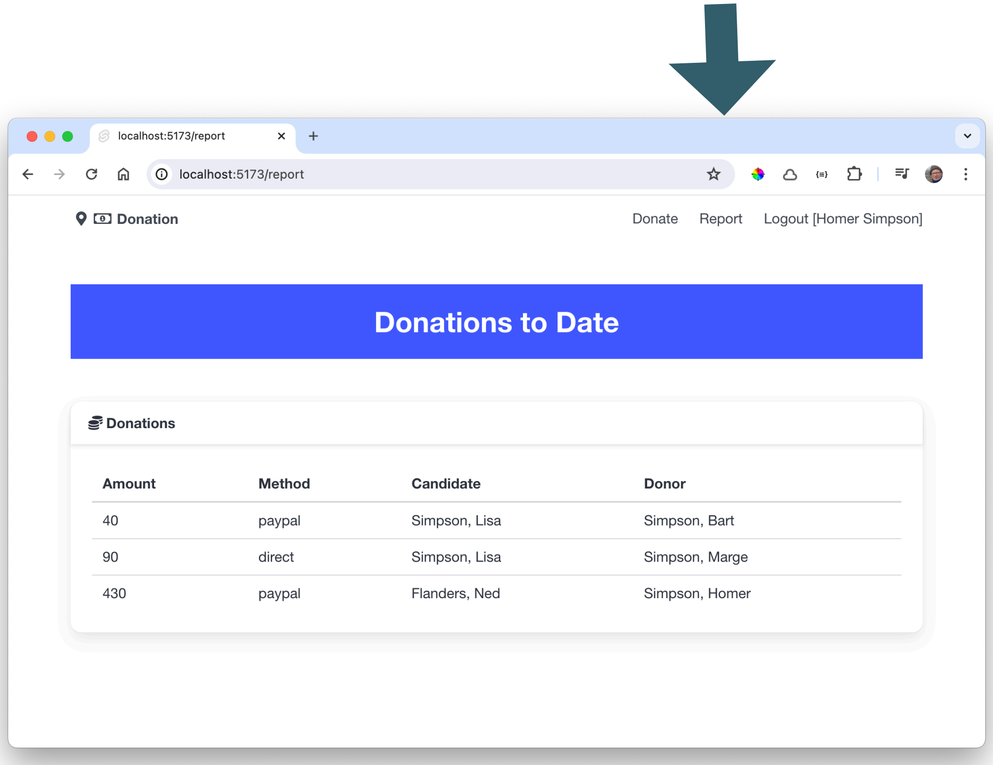
```
async login(email: string, password: string): Promise<Session | null> {
  try {
    const response = await axios.post(`${this.baseUrl}/api/users/authenticate`, {
      email,
      password
    });
    if (response.data.success) {
      axios.defaults.headers.common["Authorization"] = "Bearer " +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

- Logged in user name displayed in menu

```
<a class="navbar-item" href="/logout"> Logout [{loggedInUser.name}]</a>
```

LoginForm.svelte

```
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;
    goto("/donate");
  } else {
    email = "";
    password = "";
    message = "Invalid Credentials";
  }
}
```

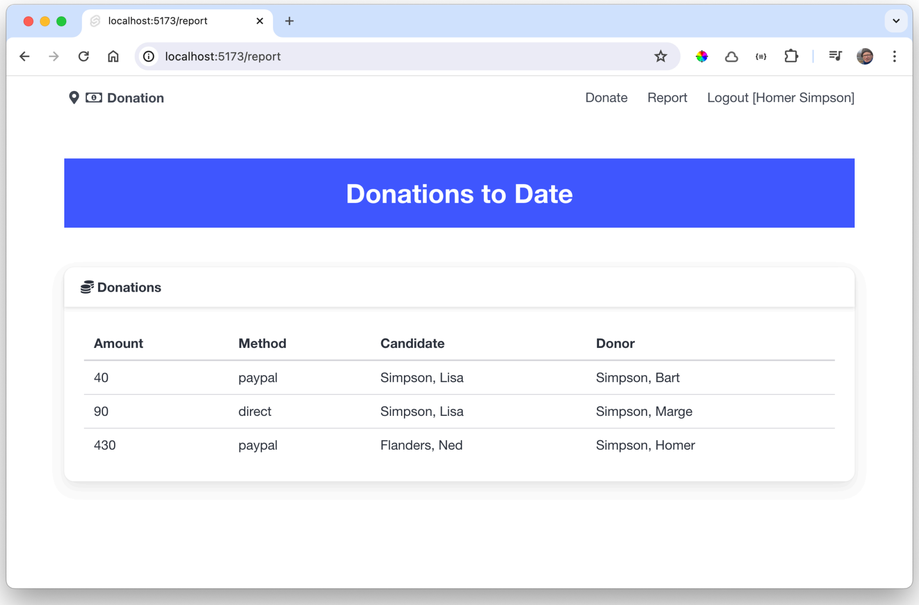


runes.svelte.ts

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

routes/+layout.svelte

```svelte
<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

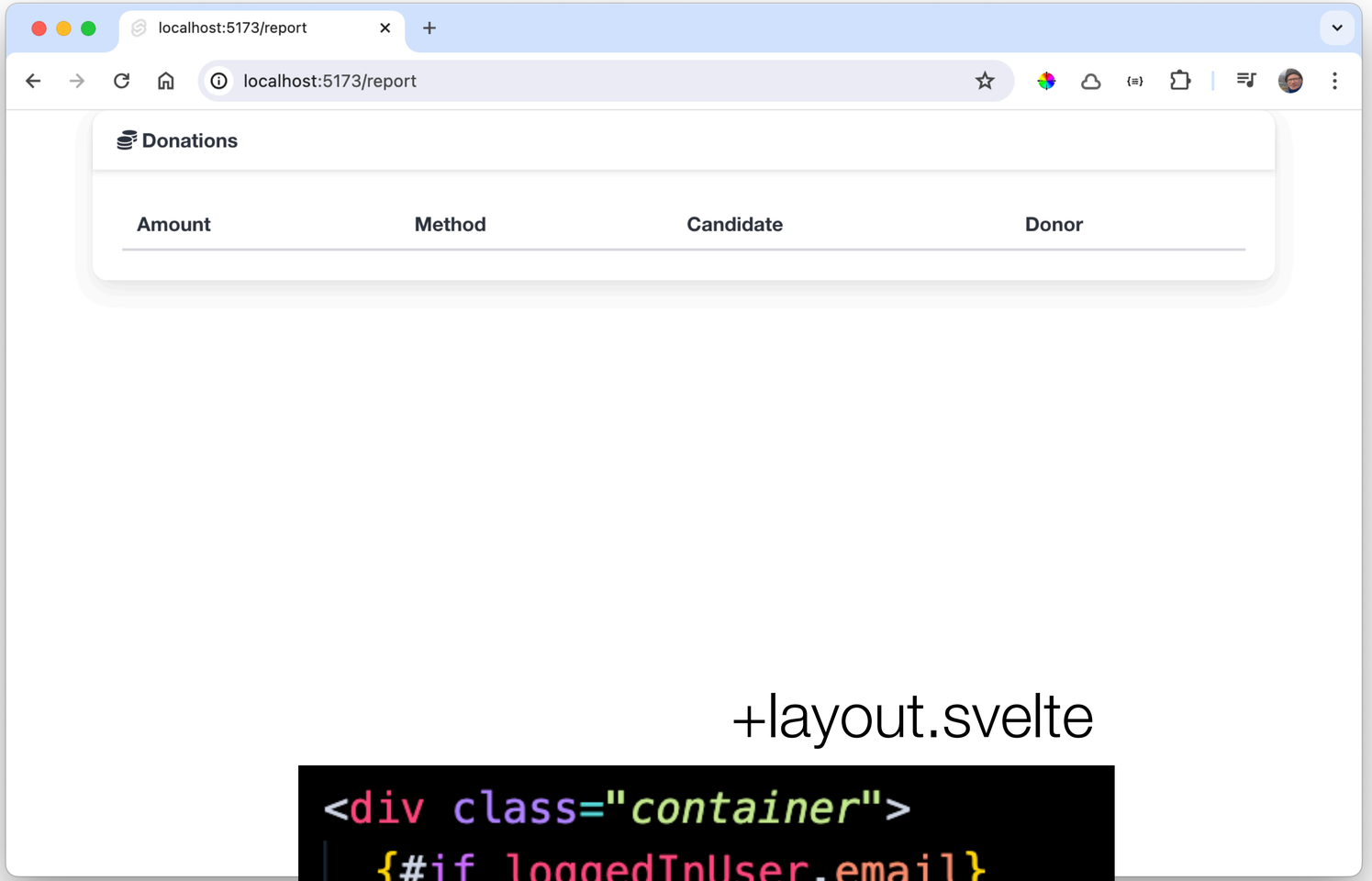


routes/report/+page.svelte

```svelte
<script lang="ts">
  import Card from "$lib/ui/Card.svelte";
  import DonationList from "$lib/ui/DonationList.svelte";
  import { loggedInUser, subTitle } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import { onMount } from "svelte";
  import type { Donation } from "$lib/types/donation-types";

  subTitle.text = "Donation to Date";

  let donations: Donation[] = [];
  onMount(async () => {
    donations = await donationService.getDonations(loggedInUser.token);
  });
</script>

<Card title="Donations">
  <DonationList {donations} />
</Card>
```

Browser Reload Button

- If browser reload button pressed:
  - Menu + Header disappears
  - List of donations no longer displayed
- When browser reloads:
  - All state lost, including runes
  - Session cleared - so layout will remove menu + Header



+layout.svelte

```
<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

Window.localStorage

The `localStorage` read-only property of the `window` interface allows you to access a `Storage` object for the `Document` 's origin; the stored data is saved across browser sessions.

Examples

The following snippet accesses the current domain's local `Storage` object and adds a data item to it using `Storage.setItem()` .

```
localStorage.setItem('myCat', 'Tom');
```

The syntax for reading the `localStorage` item is as follows:

```
const cat = localStorage.getItem('myCat');
```

The syntax for removing the `localStorage` item is as follows:

```
localStorage.removeItem('myCat');
```

The syntax for removing all the `localStorage` items is as follows:

```
localStorage.clear();
```

https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

- Utility method to save a session to Local Storage

- This will not be cleared on page reload

```
saveSession(session: Session, email: string) {
  loggedInUser.email = email;
  loggedInUser.name = session.name;
  loggedInUser.token = session.token;
  loggedInUser._id = session._id;
  localStorage.donation = JSON.stringify(loggedInUser);
},
```

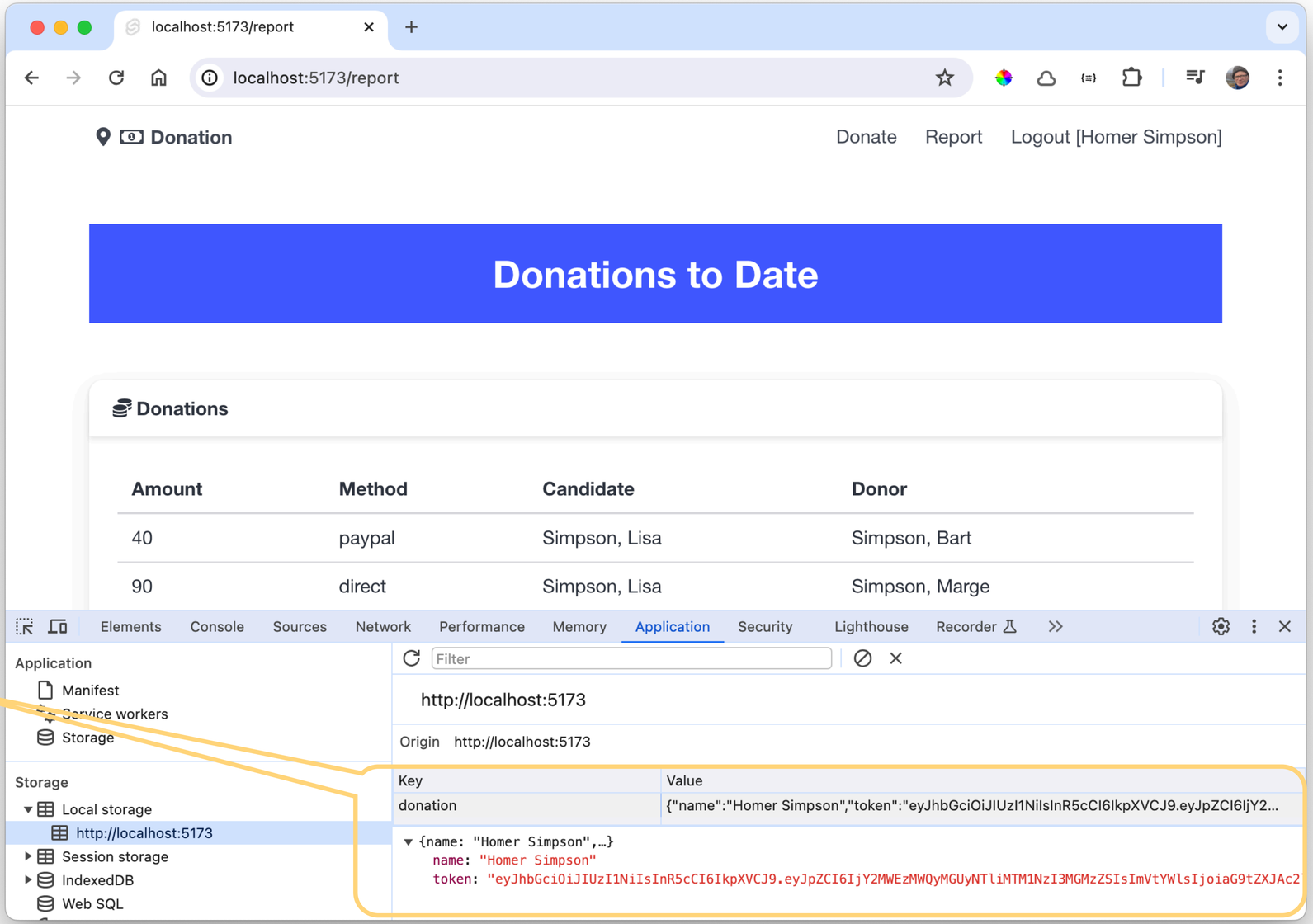
- Call this method when logging in to save session information

```
async login(email: string, password: string): Promise<Session | null> {
    const response = await axios.post(`${this.baseUrl}/api/users/authenticate`, {
      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
      };
      this.saveSession(session, email);
      return session;
    }
    return null;
  } catch (error) {
    console.log(error);
    return null;
  }
},
```

Login

```
saveSession(session: Session, email: string) {
  loggedInUser.email = email;
  loggedInUser.name = session.name;
  loggedInUser.token = session.token;
  loggedInUser._id = session._id;
  localStorage.donation = JSON.stringify(loggedInUser);
},
```

- This will persist in the browser until explicitly cleared



Clear Session

- Clears the loggedInUser rune
- Removes the contents from local storage

```
clearSession() {
  currentDonations.donations = [];
  currentCandidates.candidates = [];
  loggedInUser.email = "";
  loggedInUser.name = "";
  loggedInUser.token = "";
  loggedInUser._id = "";
  localStorage.removeItem("donation");
},
```

Logout


```
<a class="navbar-item" href="/logout"> Logout [{loggedInUser.name}]</a>
```

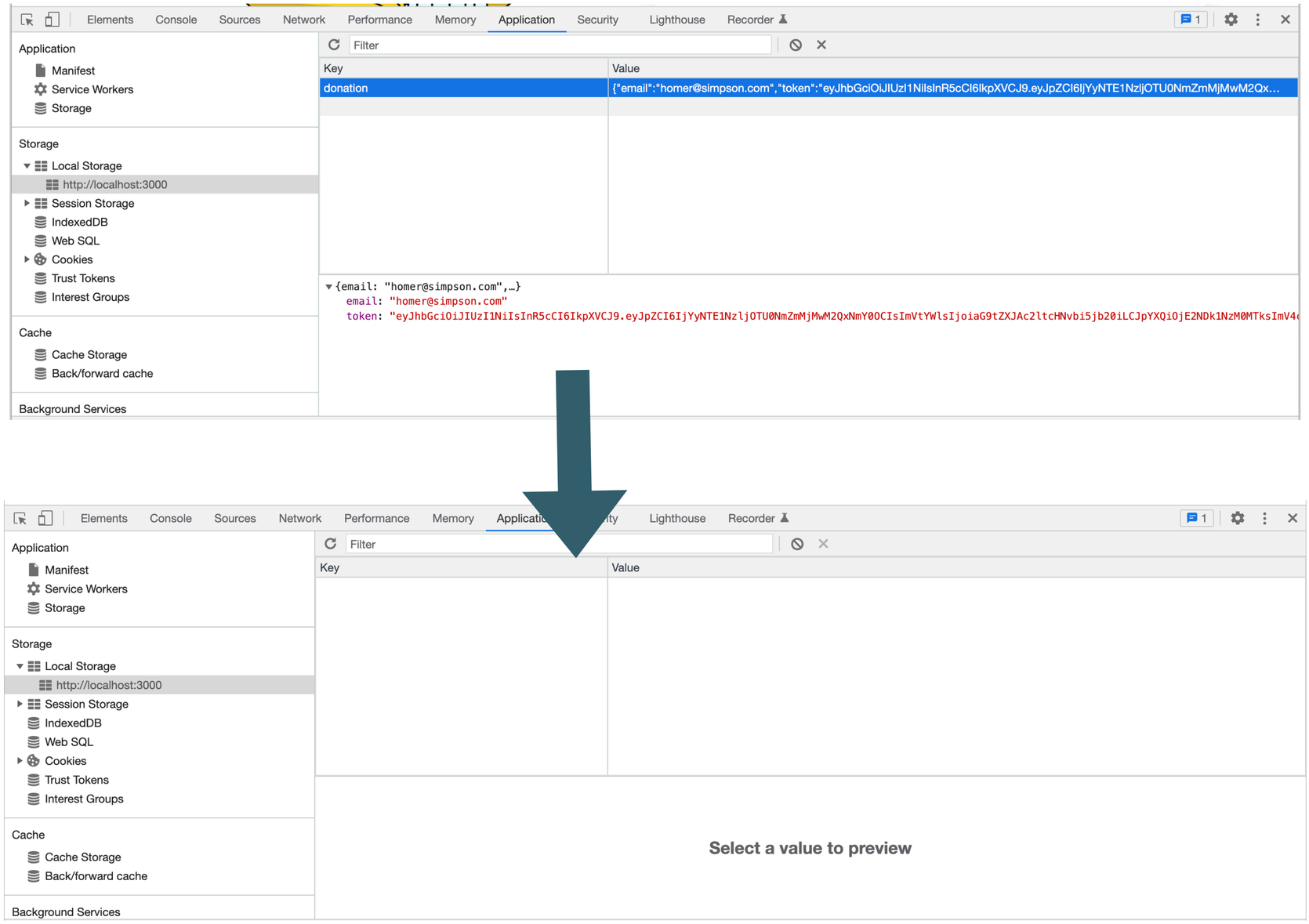
/src/routes/logout/+page.svelte

- Logout route:
  - Clears the session
  - Navigate to the start page

```
<script lang="ts">
  import { goto } from "$app/navigation";
  import { donationService } from "$lib/services/donation-service";

  donationService.clearSession();
  goto("/");
</script>
```

Logout

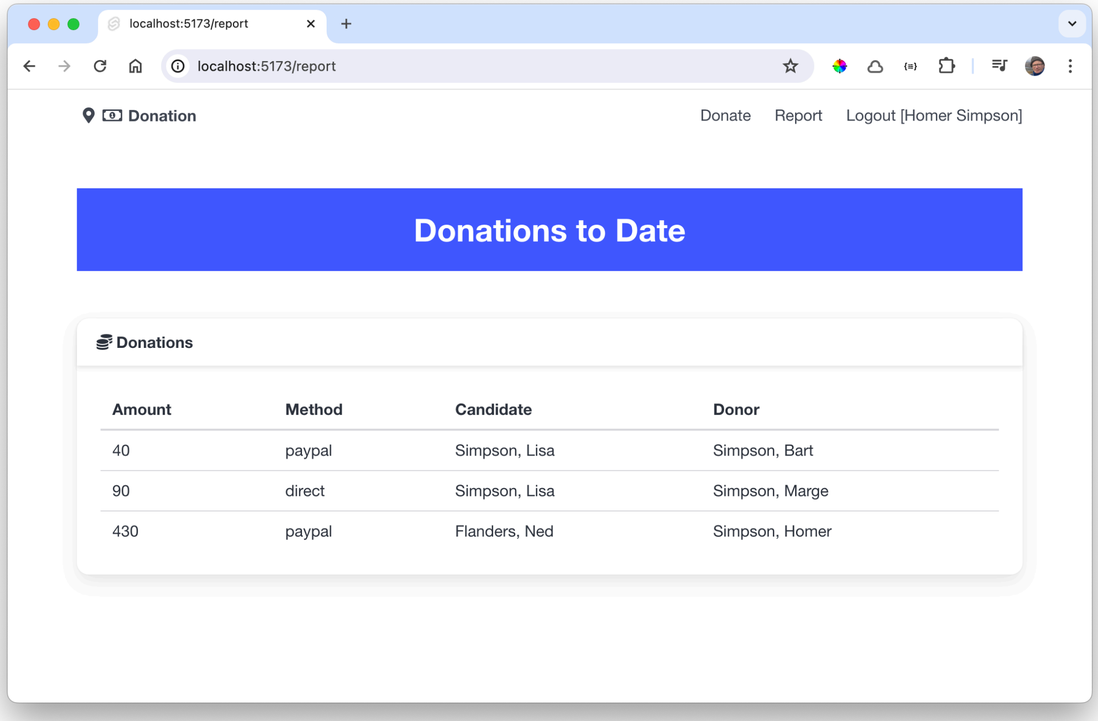
- When Logout button pressed, local storage will be cleared.



Browser Reload Button

+layout.svelte

- There should be no change in application status




```svelte
<script lang="ts">
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";
  import { onMount } from "svelte";

  onMount(async () => {
    await donationService.restoreSession();
  });
</script>

<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

Browser Reload Button

+layout.svelte

- A page in the application is being loaded for the first time (or a reload page is activated).
- Restore Session (if appropriate)

```
async restoreSession() {
  const savedLoggedInUser = localStorage.donation;
  if (savedLoggedInUser) {
    const session = JSON.parse(savedLoggedInUser);
    loggedInUser.email = session.email;
    loggedInUser.name = session.name;
    loggedInUser.token = session.token;
    loggedInUser._id = session._id;
  }
},
```

```
<script lang="ts">
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";
  import { onMount } from "svelte";

  onMount(async () => {
    await donationService.restoreSession();
  });
</script>

<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

```
async restoreSession() {
  const savedLoggedInUser = localStorage.donation;
  if (savedLoggedInUser) {
    const session = JSON.parse(savedLoggedInUser);
    loggedInUser.email = session.email;
    loggedInUser.name = session.name;
    loggedInUser.token = session.token;
    loggedInUser._id = session._id;
  }
  await this.refreshDonationInfo();
},
```

Restore Session

- Fetch latest donations + candidates

```
async refreshDonationInfo() {
  if (loggedInUser.token) {
    currentDonations.donations = await this.getDonations(loggedInUser.token);
    currentCandidates.candidates = await this.getCandidates(loggedInUser.token);
  }
},
```

- Fetch latest donations + candidates

```
async refreshDonationInfo() {
  if (loggedInUser.token) {
    currentDonations.donations = await this.getDonations(loggedInUser.token);
    currentCandidates.candidates = await this.getCandidates(loggedInUser.token);
  }
},
```

Amount	Method	Candidate
40	paypal	Simpson, Lisa
90	direct	Simpson, Lisa
430	paypal	Flanders, Ned
55	direct	Simpson, Maggie

Runes for Donations & Candidates

- Store the latest donations + candidates as application state
- This will ensure any view relying on the latest values of this state will be immediately updated in the DOM

```
export const currentDonations = $state({ donations: [] as Donation[] });
export const currentCandidates = $state({ candidates: [] as Candidate[] });
```

SvelteKit Sessions

Preserve the SvelteKit sessions between page reloads