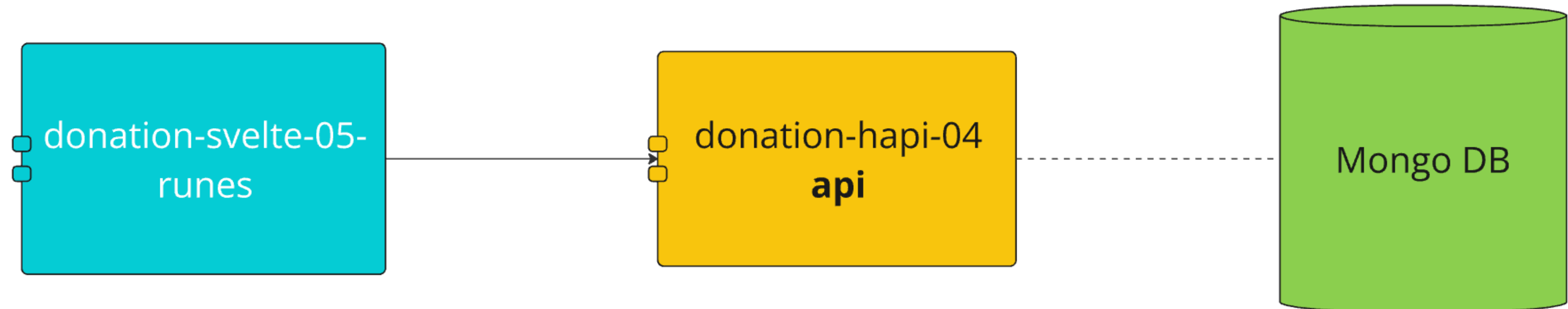


## SvelteKit Sessions



Preserve the SvelteKit  
sessions between page  
reloads

# Svelte Authentication



- Enhance session/jwt token management
- Synchronise donations & candidate lists

# 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) {
    goto("/donate");
  }
}
```

# 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;
    }
  }
}
```

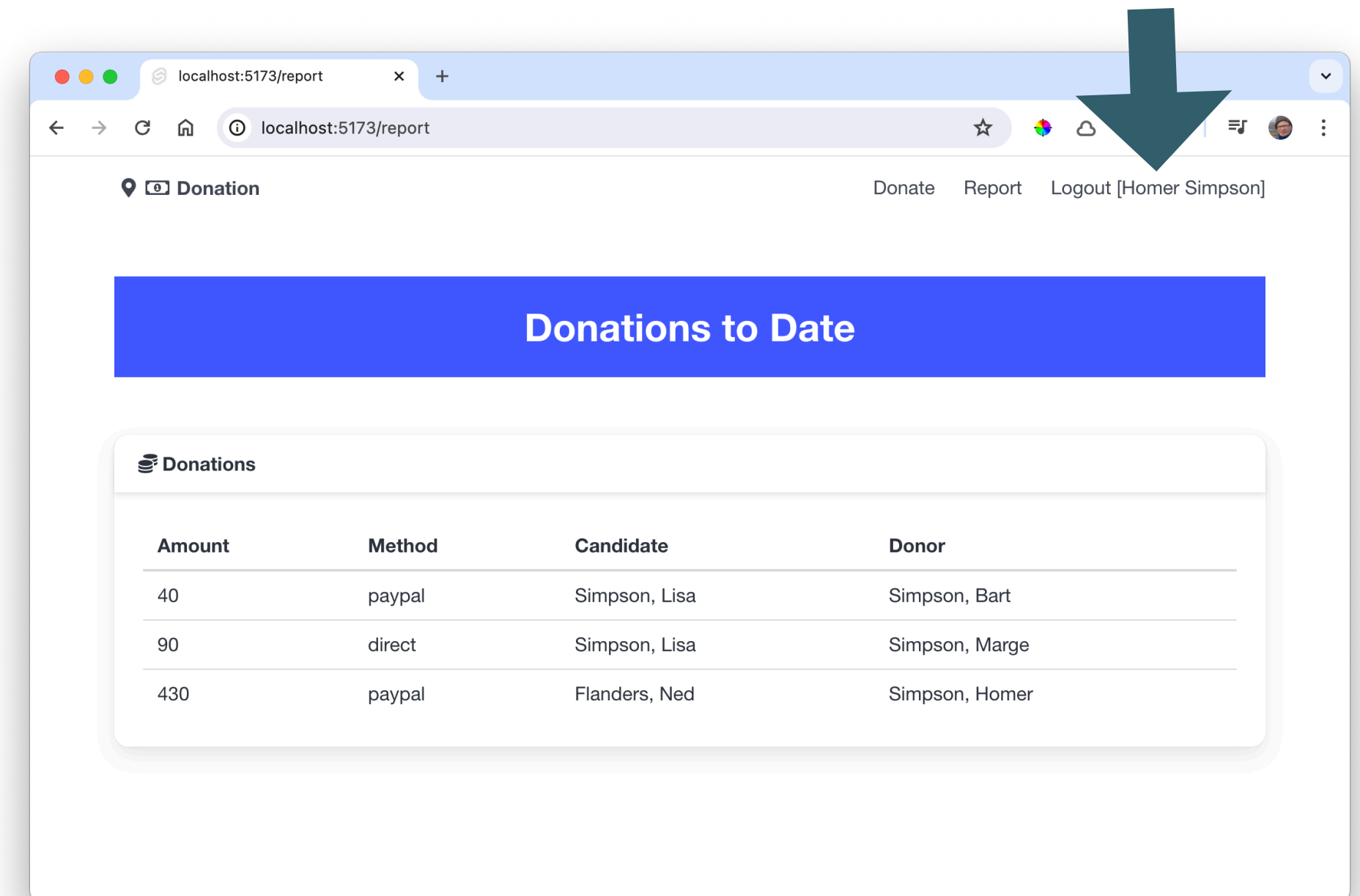
# +layout.svelte

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

# +report.svelte

- Logged in user name displayed in menu

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



```
<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";

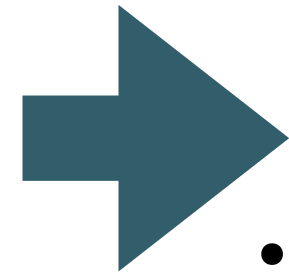
  subTitle.text = "Donation to Date";

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

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

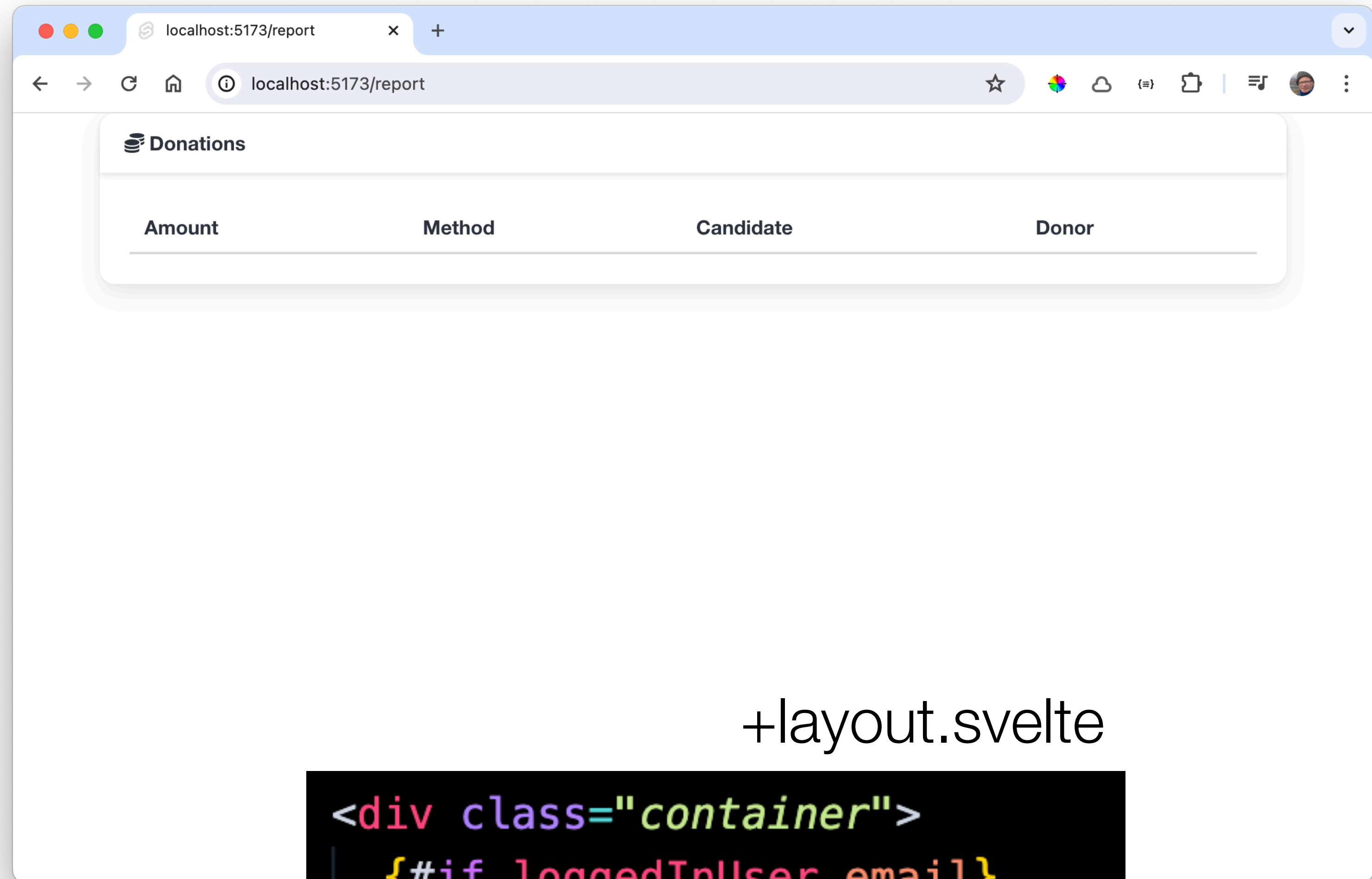
# Browser Reload Button

- If browser reload button pressed:

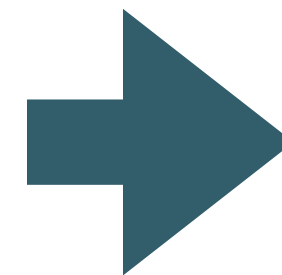


- Menu + Header disappears
- List of donations no longer displayed

- 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');
```

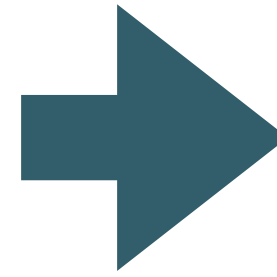


The syntax for removing all the `localStorage` items is as follows:

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



- 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

```
export const donationService = {  
  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

The screenshot shows a web application running on localhost:5173. The page title is 'Donation' and it has navigation links for 'Donate', 'Report', and 'Logout [Homer Simpson]'. A large blue banner at the top reads 'Donations to Date'. Below this is a table titled 'Donations' with columns 'Amount', 'Method', 'Candidate', and 'Donor'. The table contains two rows of data:

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	Simpson, Bart
90	direct	Simpson, Lisa	Simpson, Marge

Below the table, the browser's Application tab is open, showing the 'Storage' section. Under 'Local storage', the entry for 'http://localhost:5173' is selected. The 'donation' key is highlighted, and its value is shown as a JSON object: `{name: "Homer Simpson", token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MWEzMWQyMGUyNTliMTM1NzI3MGZlZSIsImVtYWlsIjoiaG9tZXJAc2..."}`. A yellow arrow points from the text 'This will persist in the browser until explicitly cleared' to the 'donation' key in the Application tab.



## Clear Session

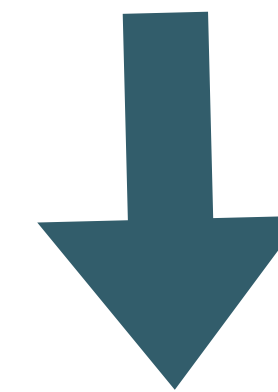
- Clears the loggedInUser rune
- Removes the contents from local storage

```
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

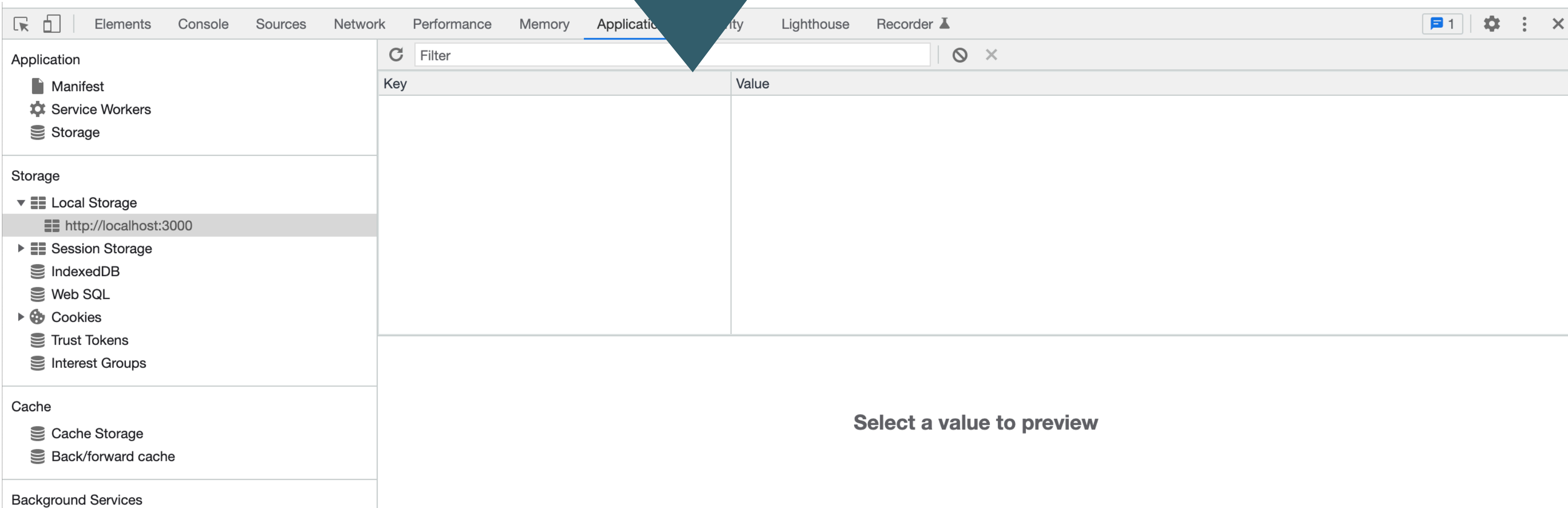
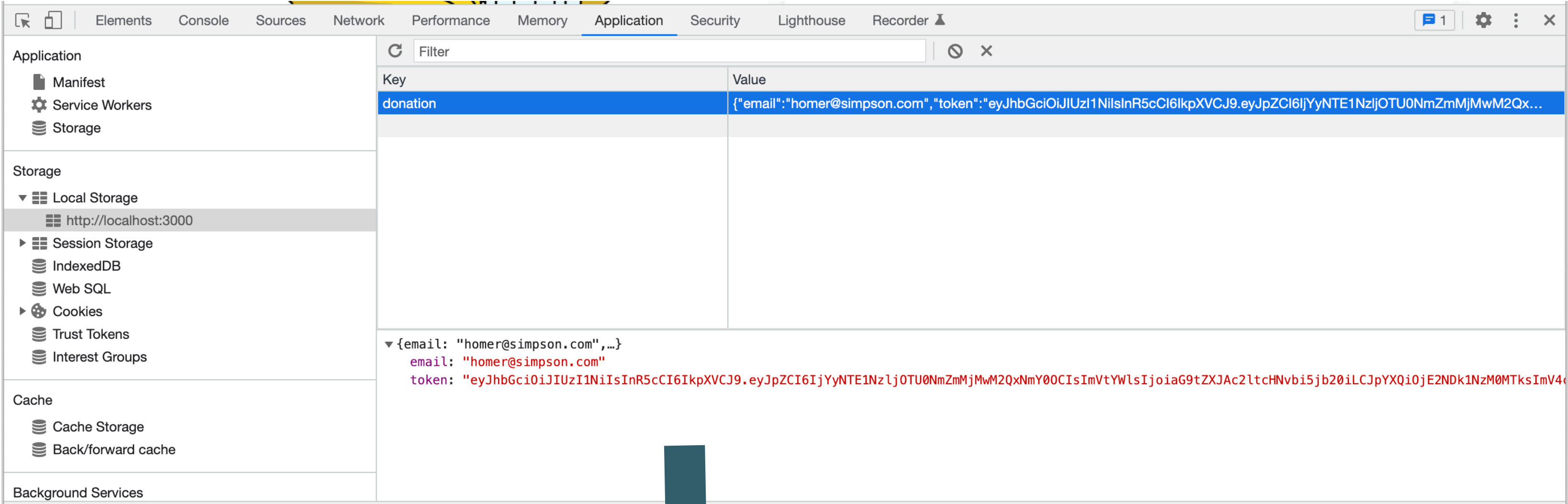
- Logout route:
- Clears the session
- Navigate to the start page

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

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

# Logout

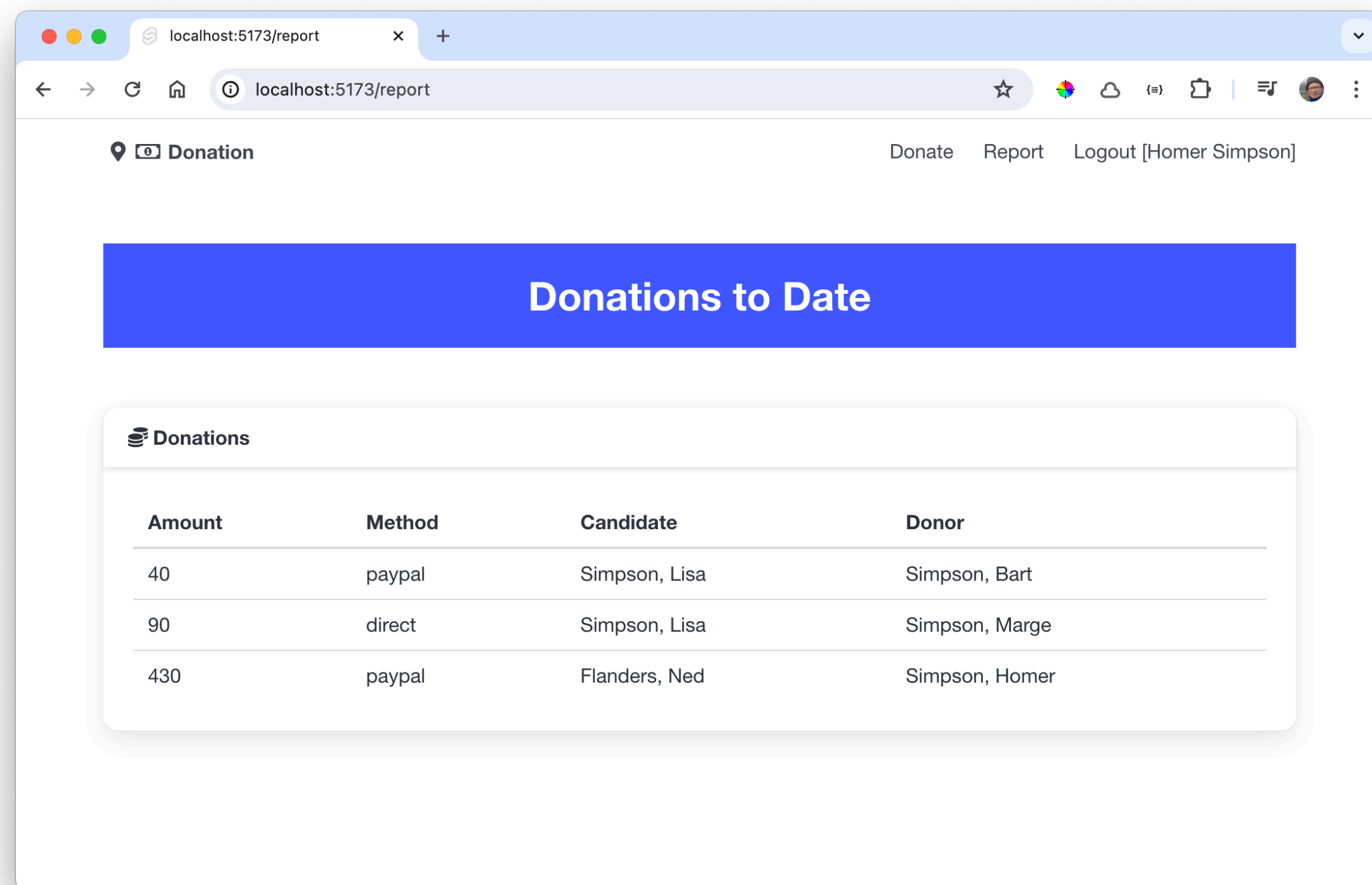
- When Logout button pressed, local storage will be cleared.



# Browser Reload Button

+layout.svelte

- There should be no change in application status



```
<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";

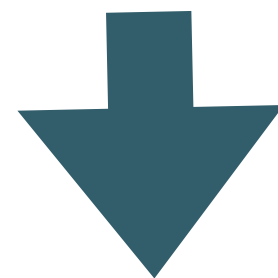
  onMount(async () => {
    await donationService.restoreSession();
  });
</script>

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

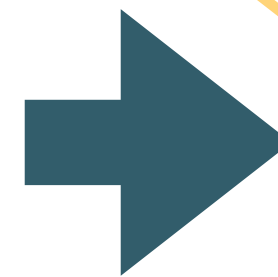
# Browser Reload Button

+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;  
  }  
},
```



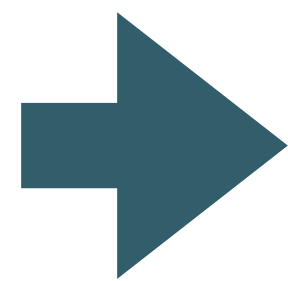
```
<script lang="ts">  
  import { loggedInUser } from "$lib/runes.svelte";  
  import { donationService } from "$lib/services/donation-service";  
  import Heading from "$lib/ui/Heading.svelte";  
  import Menu from "$lib/ui/Menu.svelte";  
  import { onMount } from "svelte";  
  
  onMount(async ()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

## SvelteKit Sessions



Preserve the SvelteKit  
sessions between page  
reloads