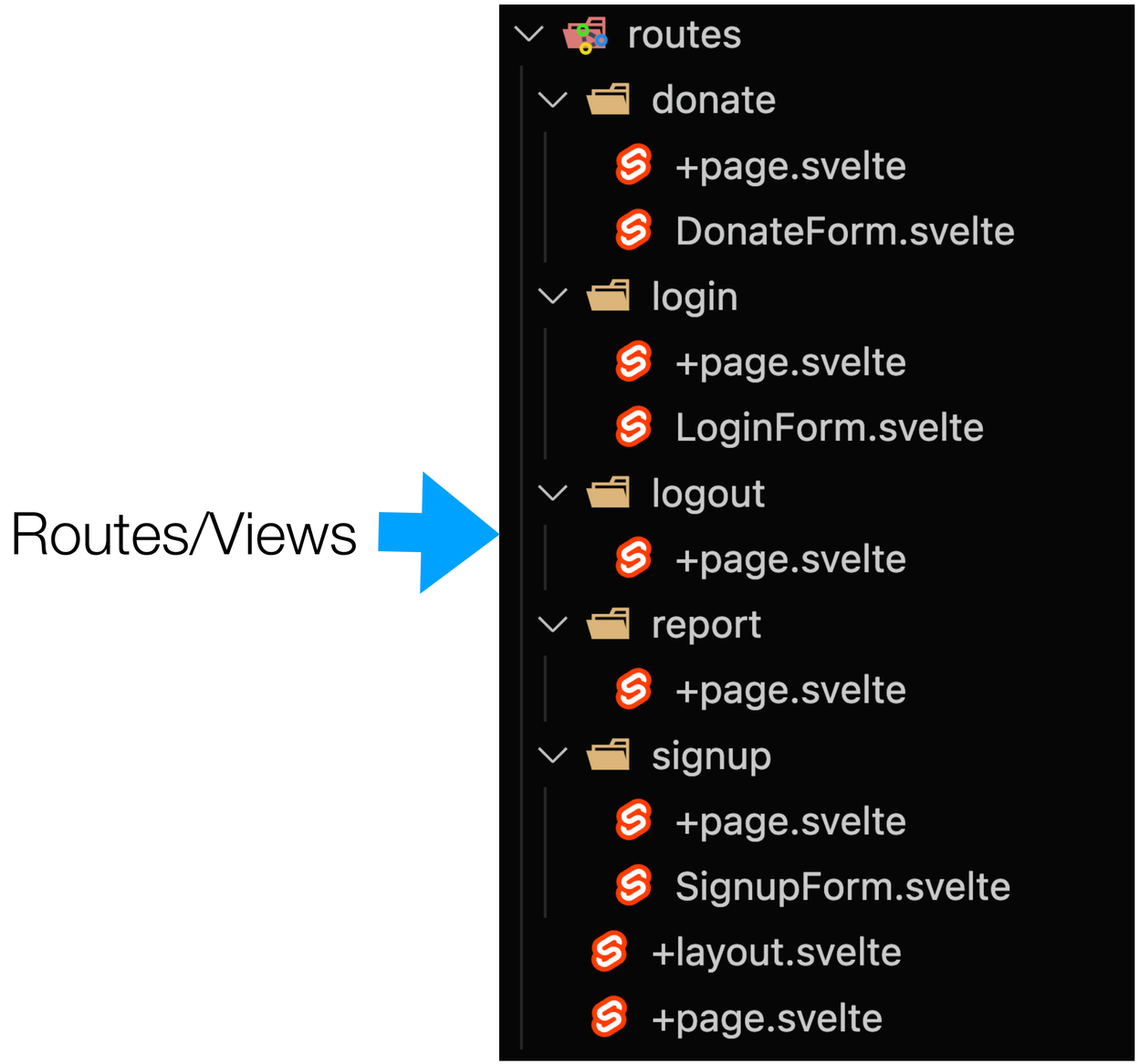
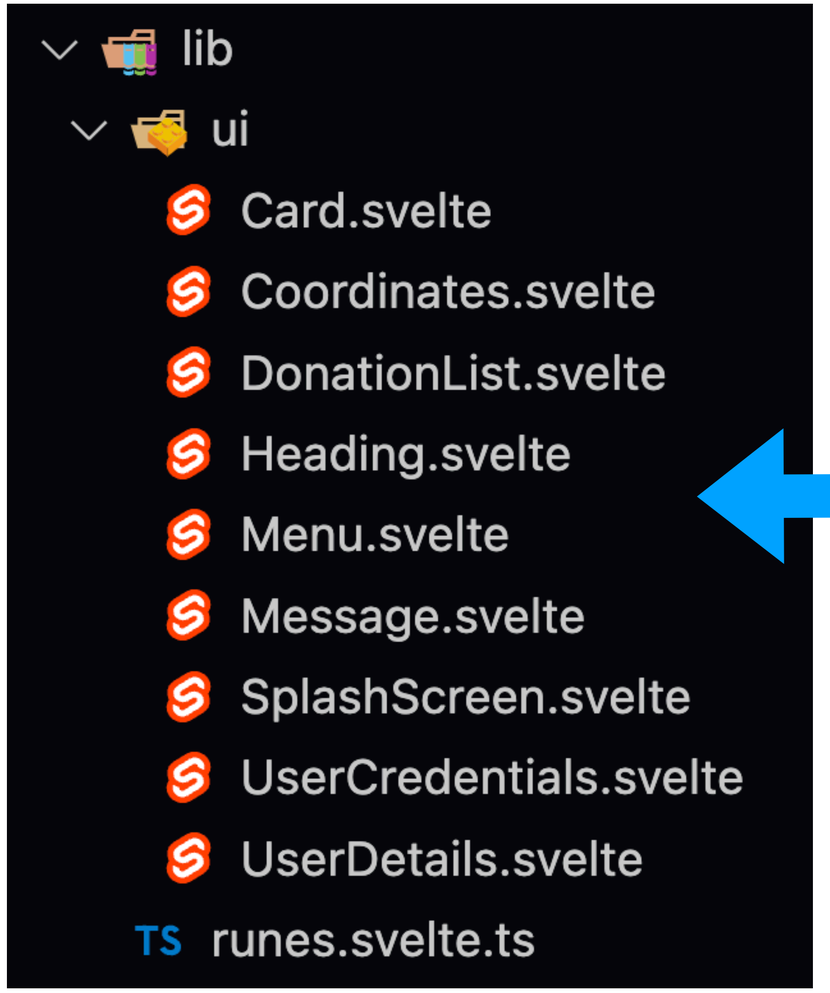
Pages & Components

Creating and assembling Svelte Components

SvelteKit Web Application



Routes/Views



Reusable UI Components

Runes/Shared state

Report Route

Donation

Donate Report Logout [homer@simpson.com]

Donations to Date

Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

routes/report/+page.svelte

Menu.svelte

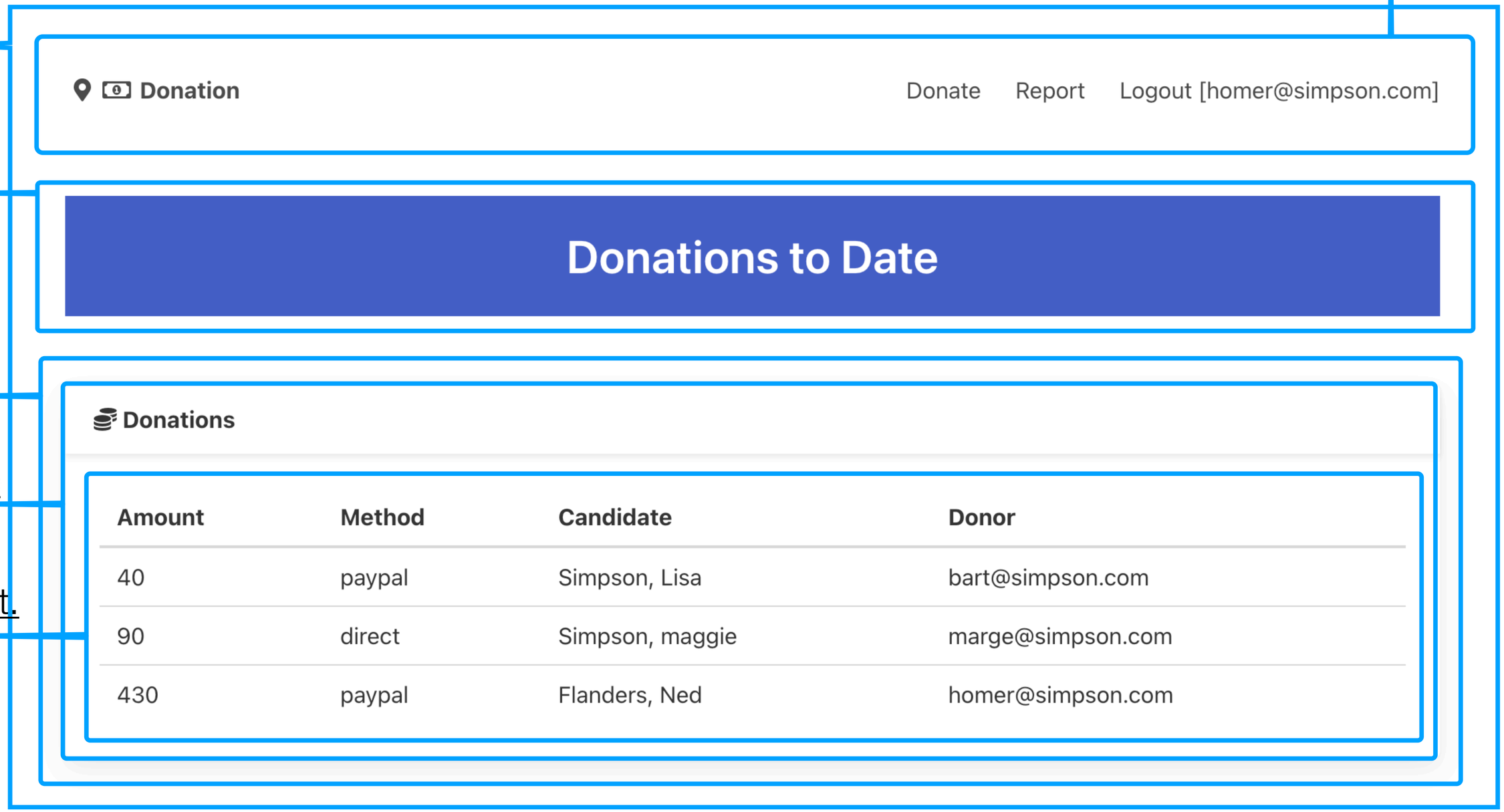
layout.svelte

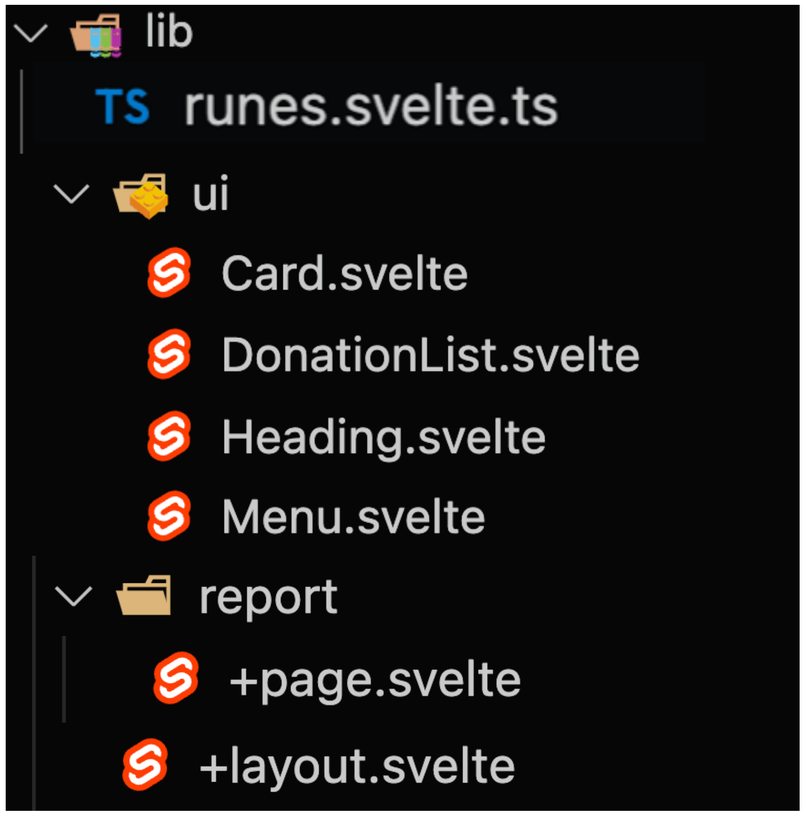
Heading.svelte

+page.svelte

Card.svelte

DonationList.svelte





routes/report/+page.svelte

Menu.svelte

layout.svelte

Heading.svelte

+page.svelte

Card.svelte

DonationList.svelte

Donation Donate Report Logout [homer@simpson.com]

Donations to Date

Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

routes/report/+page.svelte

Menu.svelte

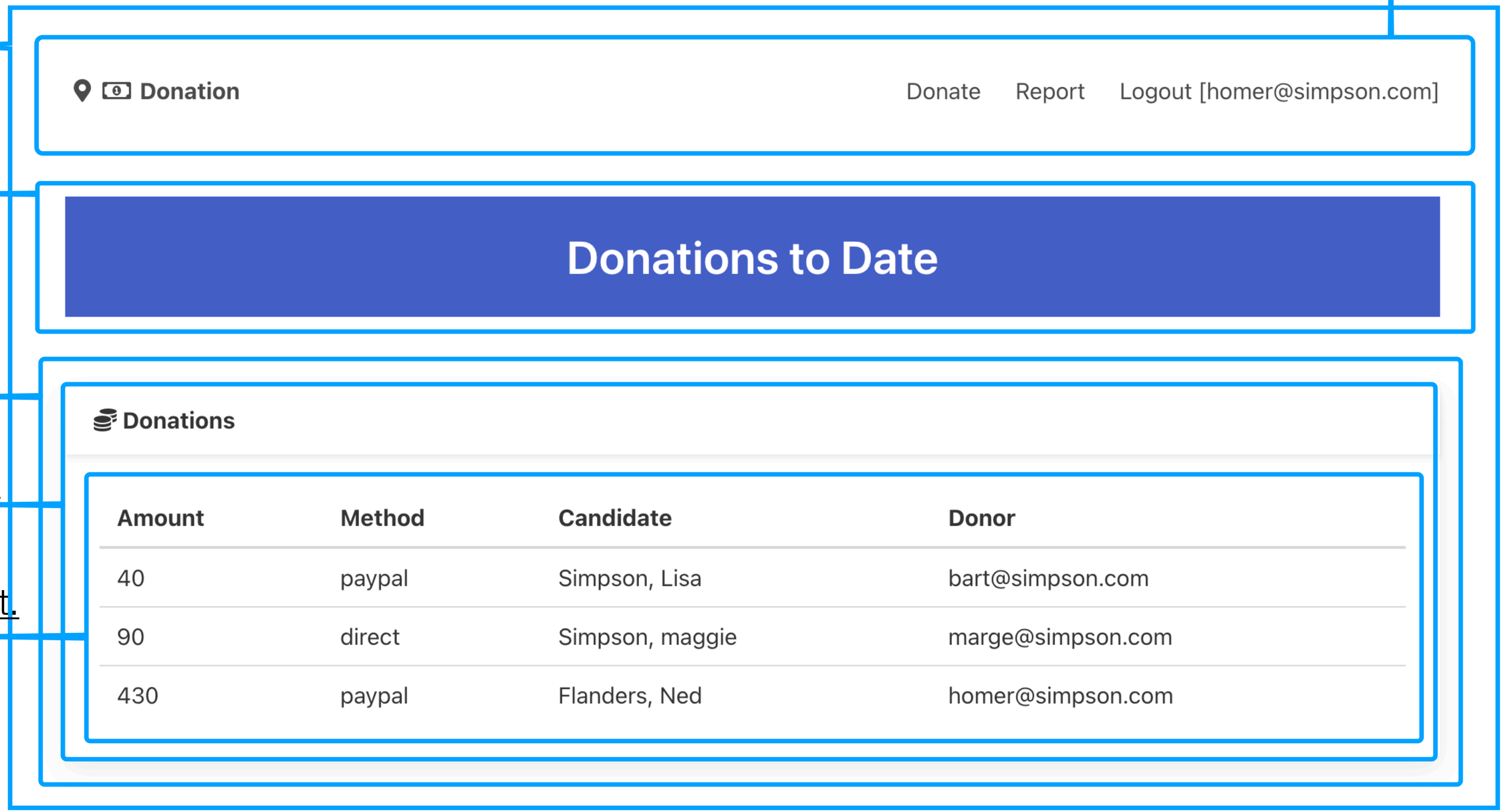
layout.svelte

Heading.svelte

+page.svelte

Card.svelte

DonationList.svelte



Menu.svelte

Menu.svelte

Import “currentSession” store

```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.email}]</a>
      </div>
      <div></div>
    </div>
  </div>
</nav>
```

Place current value of rune

runes.svelte.ts

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

routes/report/+page.svelte

Menu.svelte

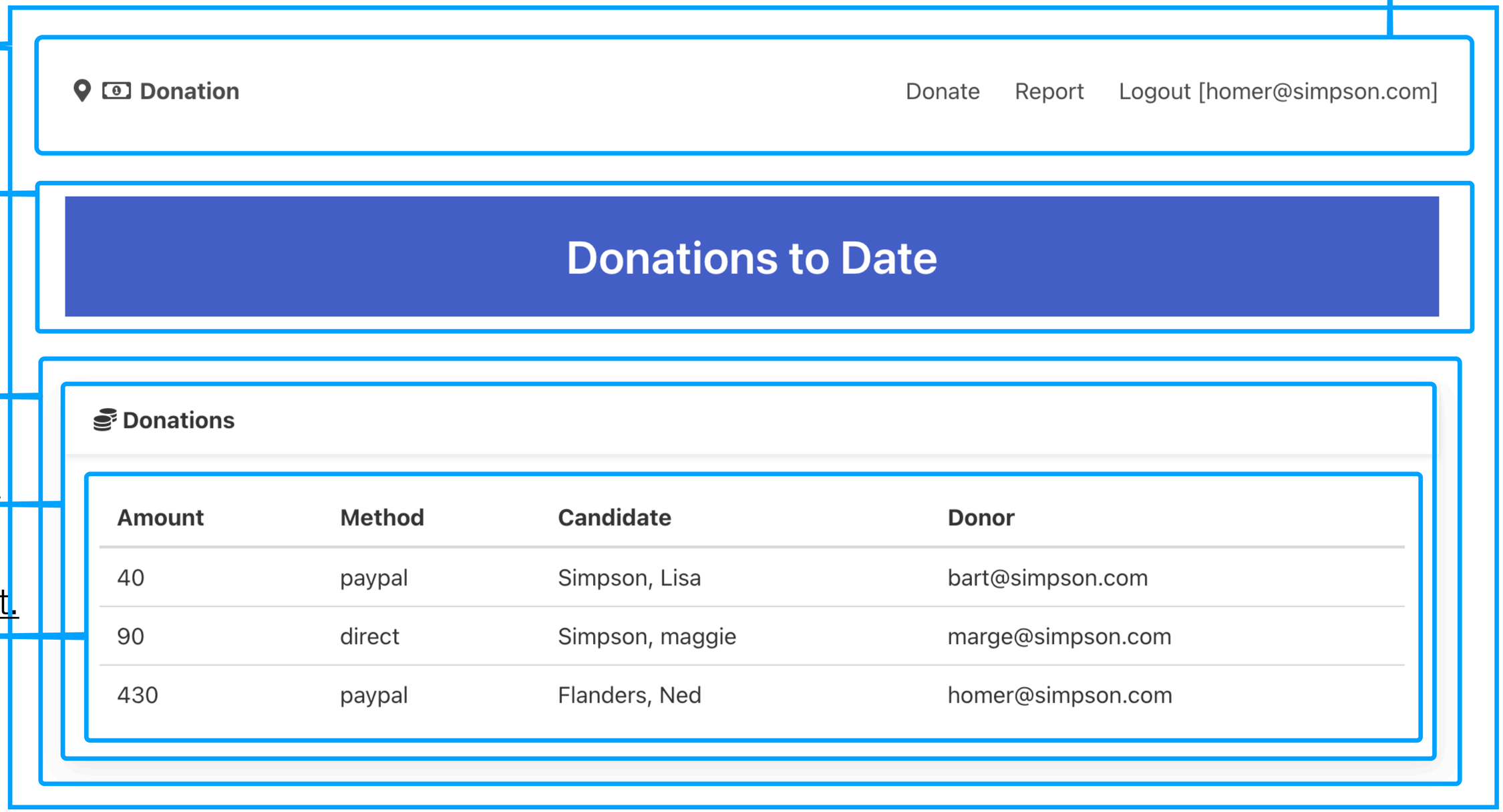
layout.svelte

Heading.svelte

+page.svelte

Card.svelte

DonationList.svelte



Heading.
svelte

Donations to Date

Heading.svelte

Import “subTitle” rune

Place current value of rune

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

<section class="hero is-link is-small mt-6 mb-6">
  <div class="hero-body">
    <div class="container">
      <h1 class="title has-text-centered">
        {subTitle.text}
      </h1>
    </div>
  </div>
</section>
```

runes.svelte.ts

```ts
export const subTitle = $state({ text: "" });
```

routes/report/+page.svelte

Menu.svelte

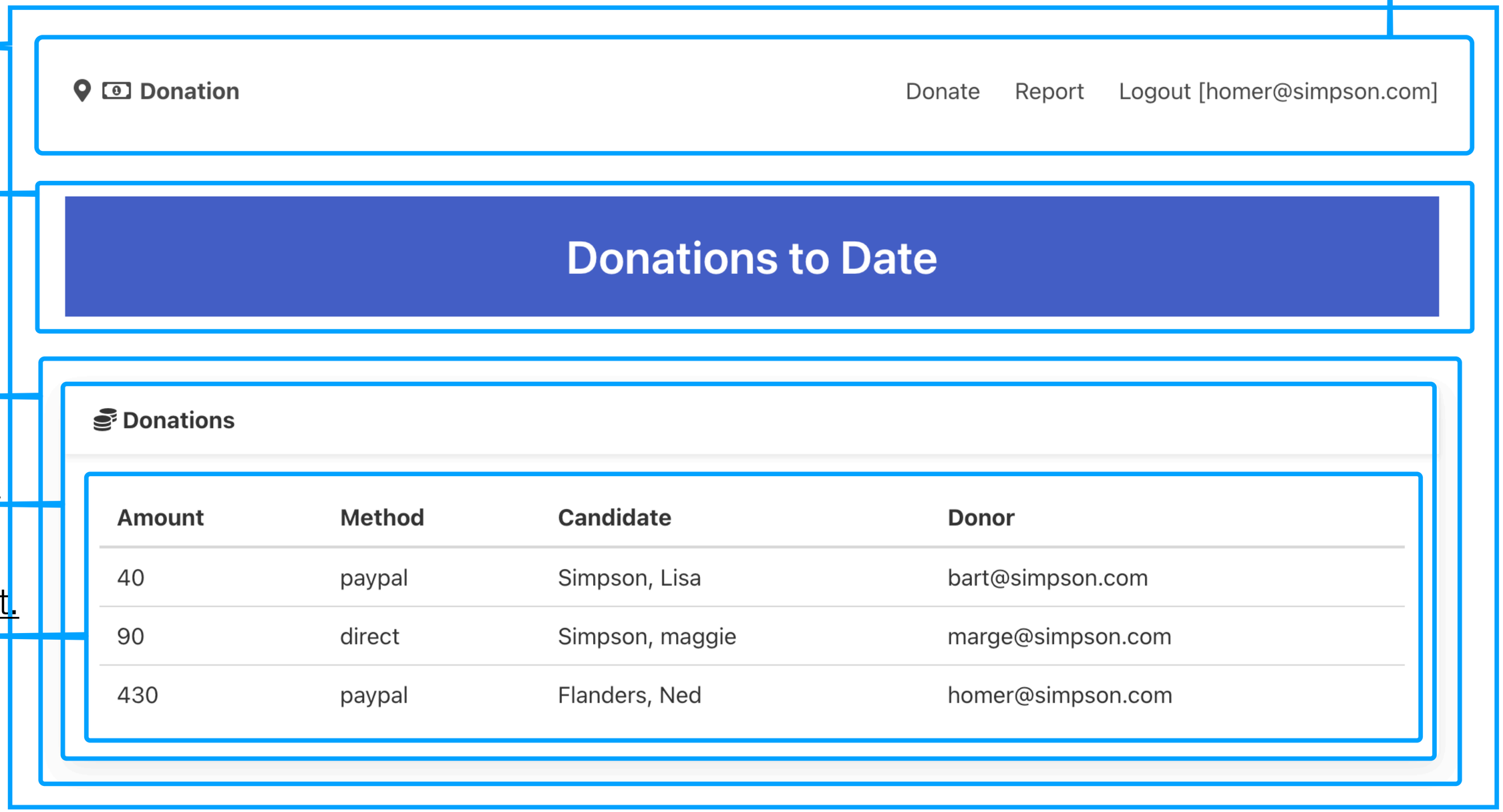
layout.svelte

Heading.svelte

+page.svelte

Card.svelte

DonationList.svelte



DonationList.svelte

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

DonationList.svelte

Expect an array of donations to be passed to the component

Iterate through each donation in the array

Display each row of the table

```
<script lang="ts">
  export let donations:any = [];
</script>

<table class="table is-fullwidth">
  <thead>
    <th>Amount</th>
    <th>Method</th>
    <th>Candidate</th>
    <th>Donor</th>
  </thead>
  <tbody>
    {#each donations as donation}
      <tr>
        <td>
          {donation.amount}
        </td>
        <td>
          {donation.method}
        </td><td>
          {donation.candidate.lastName}, {donation.candidate.firstName}
        </td>
        <td>
          {donation.donor}
        </td>
      </tr>
    {/each}
  </tbody>
</table>
```

routes/report/+page.svelte

Menu.svelte

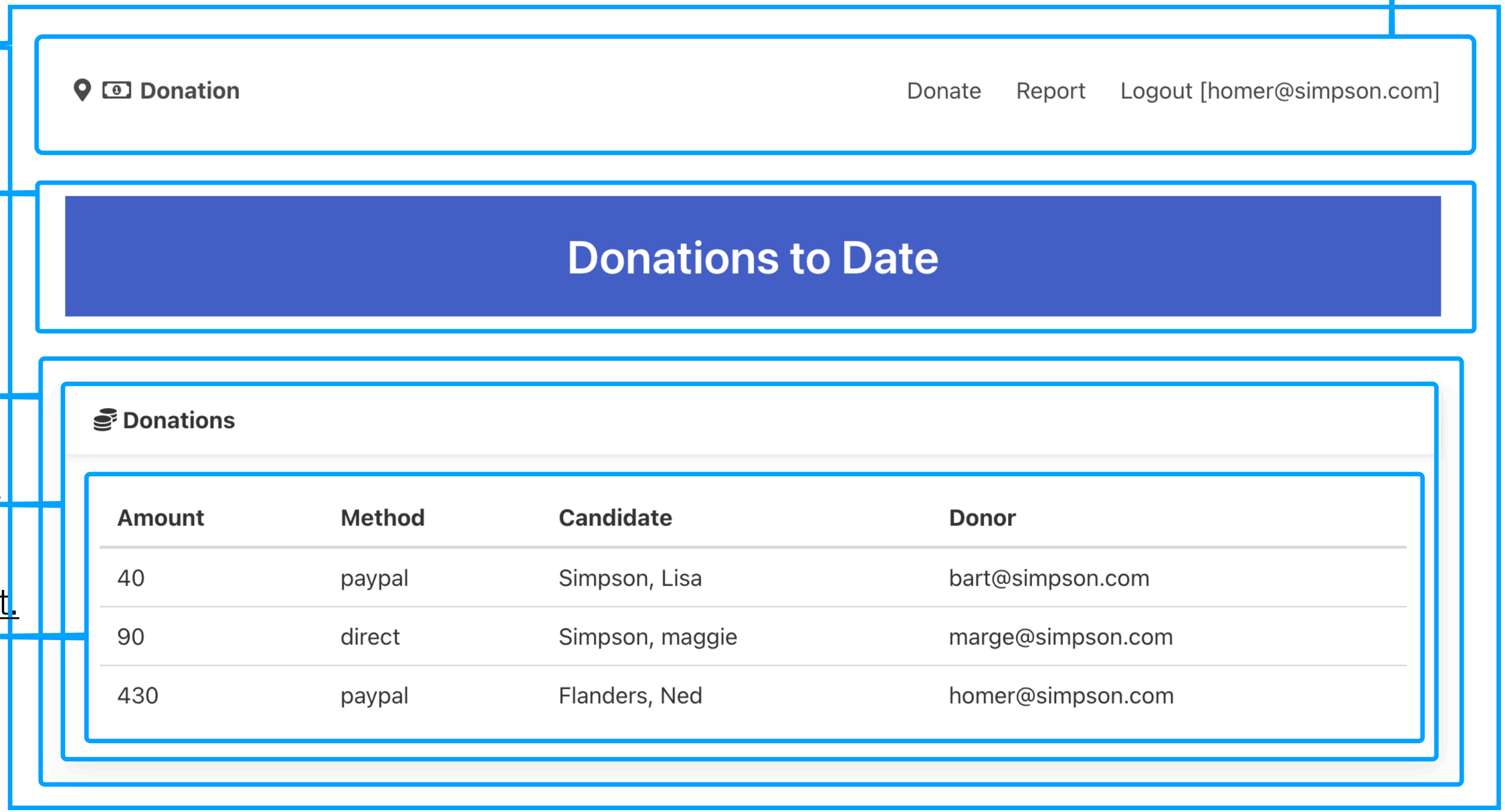
layout.svelte

Heading.svelte

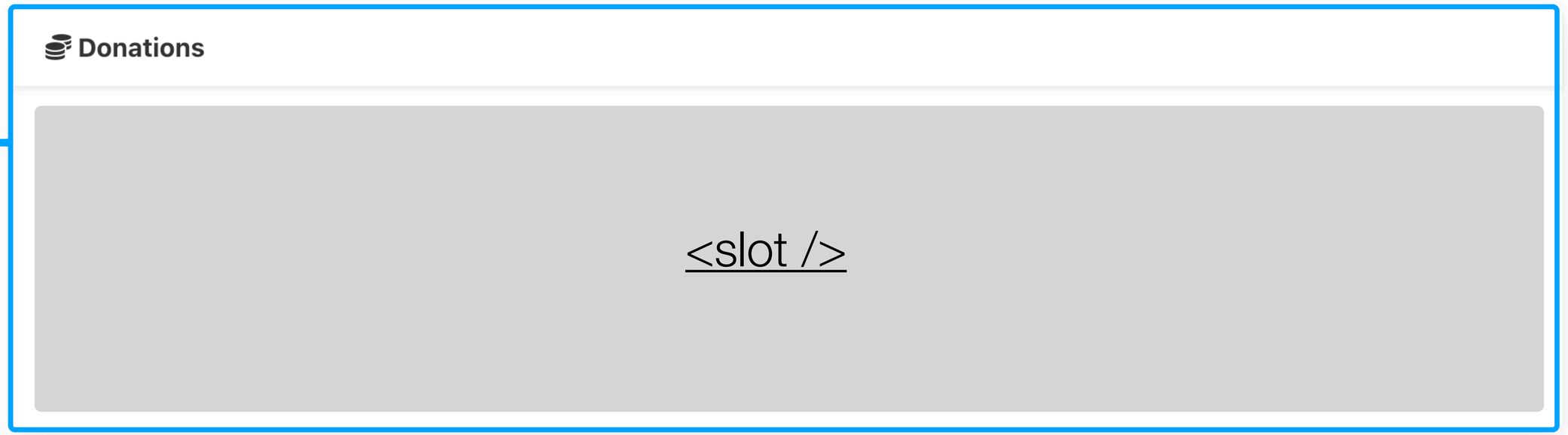
+page.svelte

Card.svelte

DonationList.svelte



Card.svelte



Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

Card.svelte

Expect ‘title’ property to be based to component

Children provided by the component that creates this Card component

```
<script lang="ts">
  let {title = "", children} = $props();
</script>

<div class="card mb-5">
  <header class="card-header">
    <p class="card-header-title">
      <span class="icon"><i class="fas fa-coins"></i></span><span>{title}</span>
    </p>
  </header>
  <div class="card-content">
    <div class="content">
      {@render children()}
    </div>
  </div>
</div>
```

Place ‘title’ in element

routes/report/+page.svelte

Menu.svelte

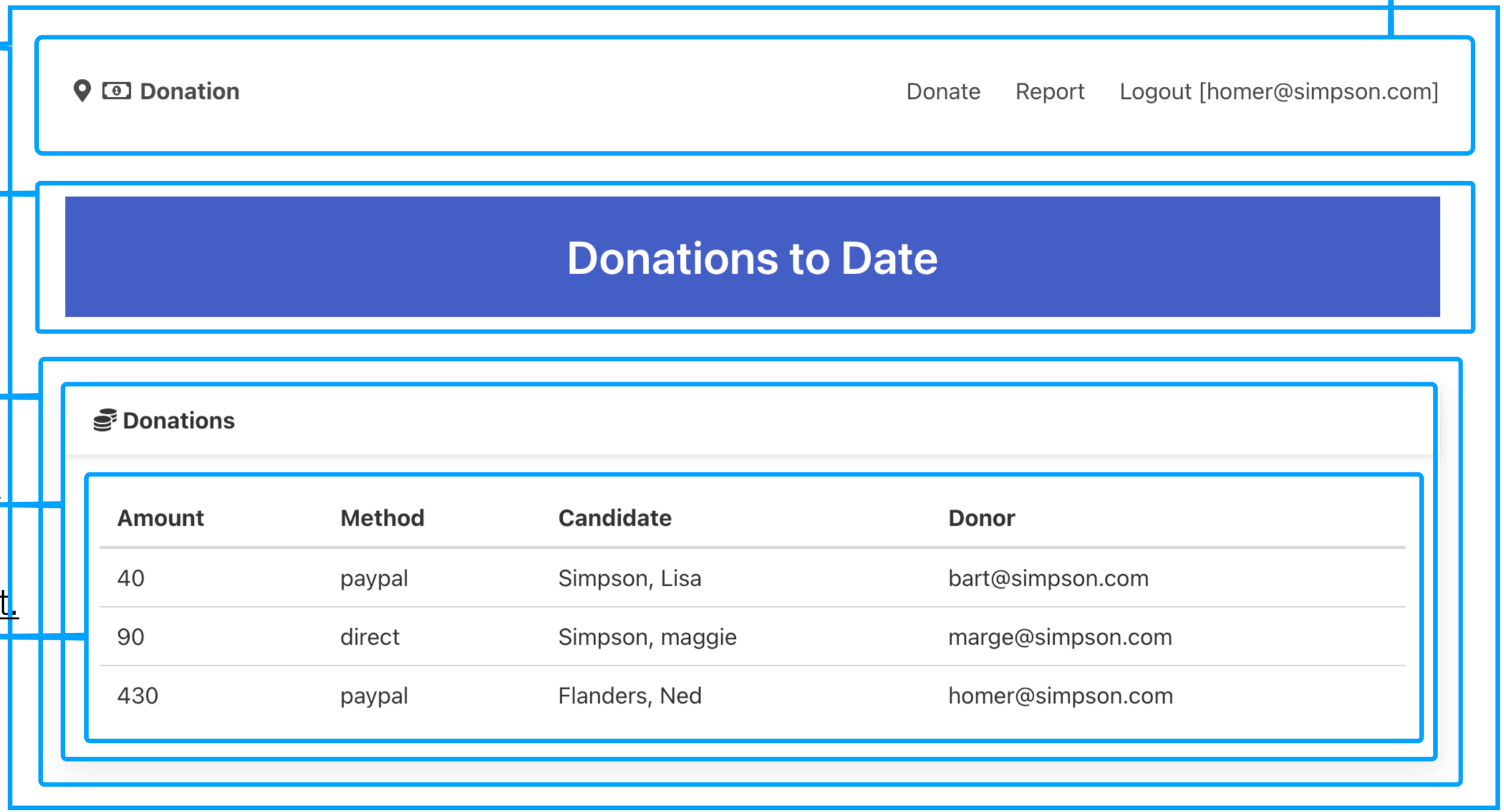
layout.svelte

Heading.svelte

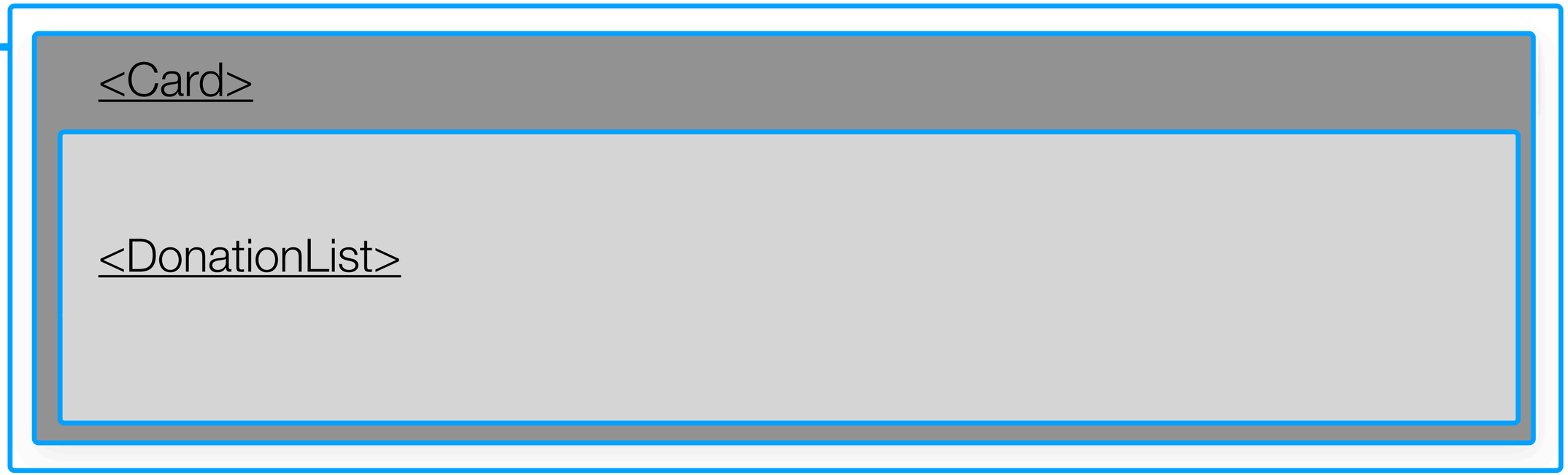
+page.svelte

Card.svelte

DonationList.svelte



+page.
svelte



Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	bart@simpson.com
90	direct	Simpson, maggie	marge@simpson.com
430	paypal	Flanders, Ned	homer@simpson.com

+page.svelte

Import subTitle rune - so we can set its value

Import Card and DonationList components

*Children :*provide the component to be ‘slotted’ into the card body

```svelte
<script lang="ts">
  import { subTitle } from "$lib/runes.svelte";
  import Card from "$lib/ui/Card.svelte";
  import DonationList from "$lib/ui/DonationList.svelte";

  subTitle.text = "Donations to date";
</script>

<Card title="Donations">
  <DonationList />
</Card>
```

routes/report/+page.svelte

Menu.svelte

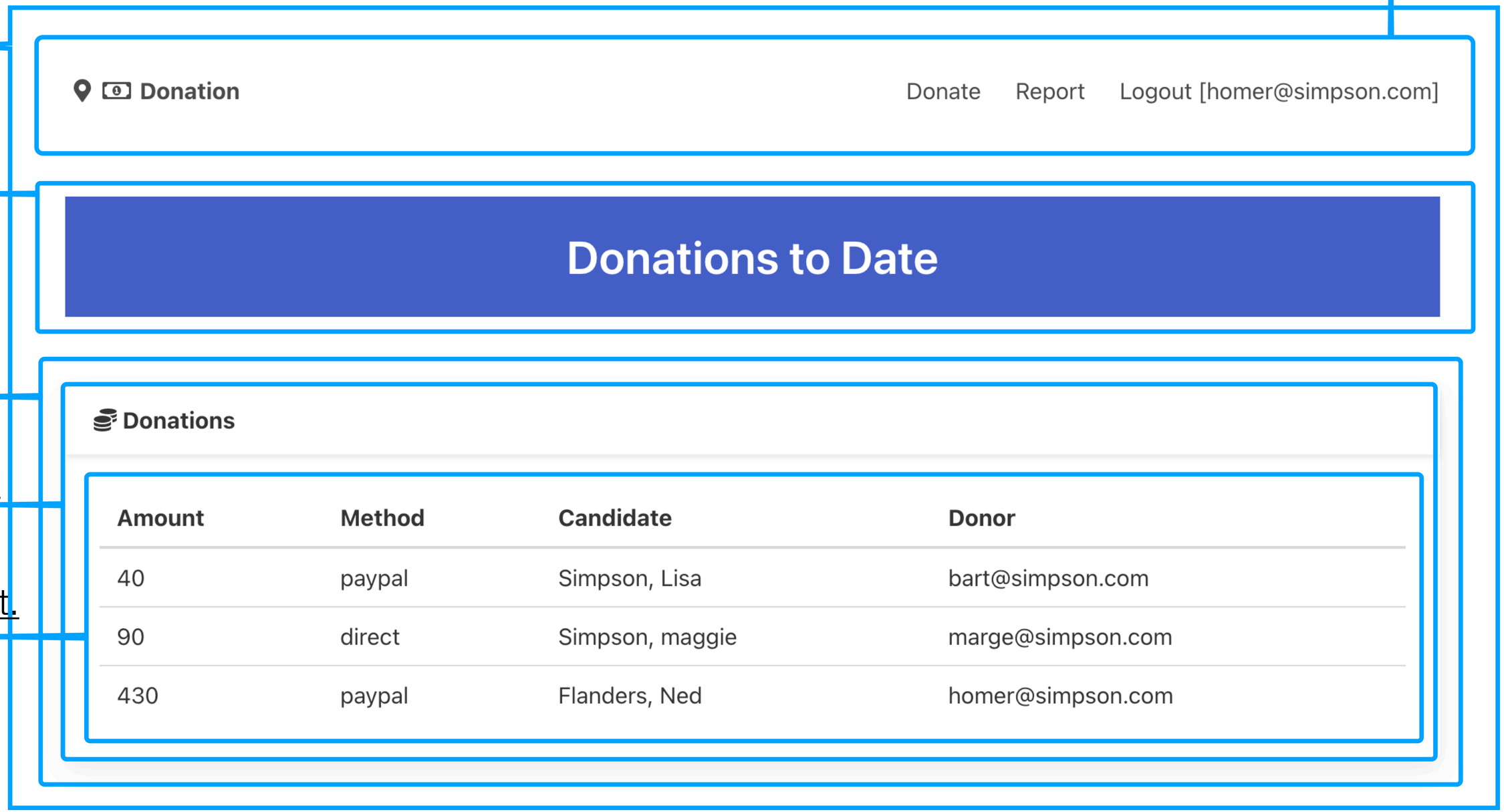
layout.svelte

Heading.svelte

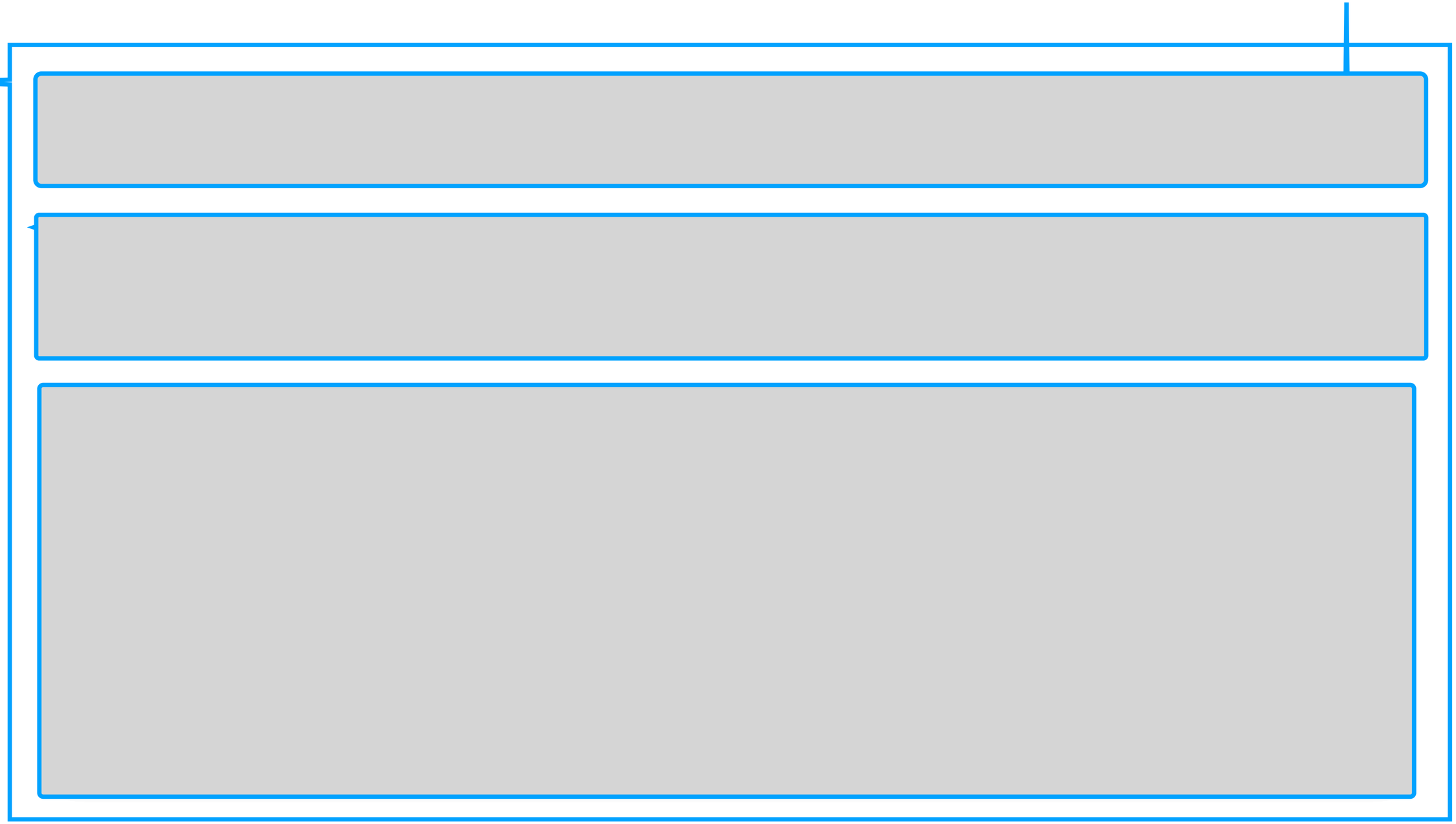
+page.svelte

Card.svelte

DonationList.svelte



layout.
svelte



+layout.svelte

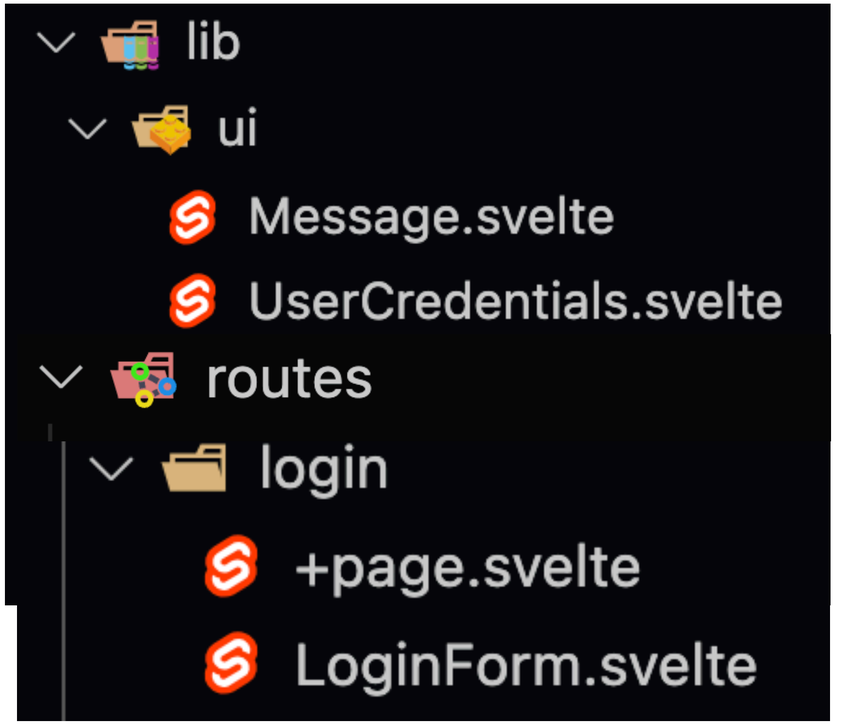
- Plays a similar role to layout.hbs in Handlebars
- Display Menu + Heading if we have a session defined
- All page content based on this layout

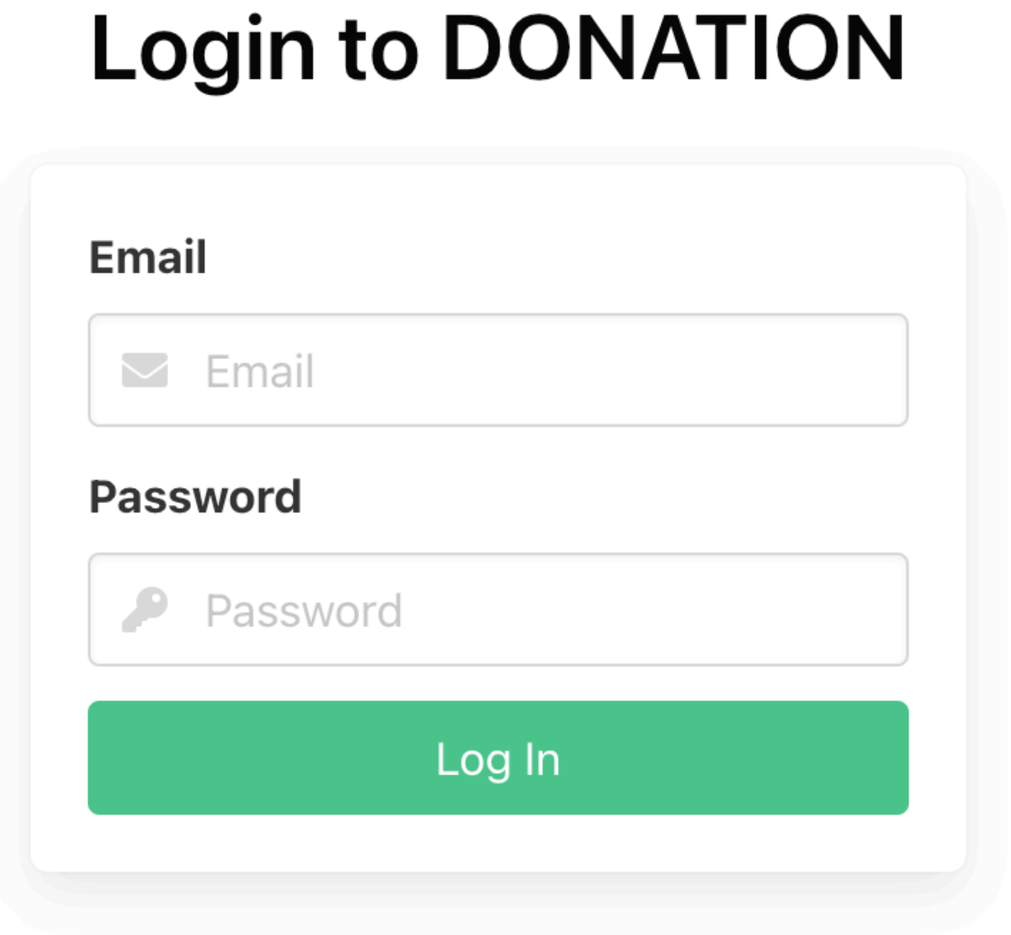
```svelte
<script lang="ts">
  import { loggedInUser } from "$lib/runes.svelte";
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";

  let {children} = $props();
</script>

<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  {@render children()}
</div>
```

Login Route


lib
ui
Message.svelte
UserCredentials.svelte
routes
login
+page.svelte
LoginForm.svelte


Login to DONATION
Email
Email
Password
Password
Log In

Message.svelte

- Simple message box tone displayed if authentication fails

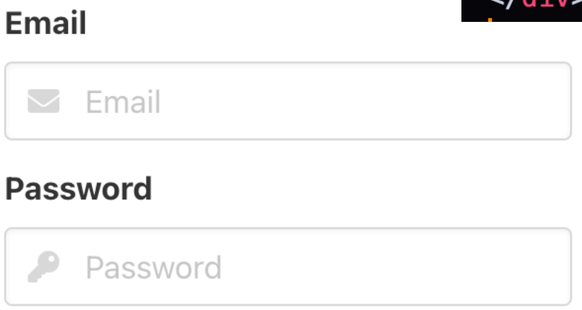
```
<script lang="ts">
  let { message } = $props();
</script>

<article class="message is-danger">
  <div class="message-body">
    {message}
  </div>
</article>
```

UserCredentials.svelte

```svelte
<script lang="ts">
  let { email = $bindable(''), password = $bindable('') } = $props();
</script>

<div class="field">
  <label class="label" for="email">Email</label>
  <div class="control has-icons-left">
    <input id="email" bind:value={email} class="input" type="text" placeholder="Email" name="email" />
    <span class="icon is-small is-left">…
    </span>
  </div>
</div>
<div class="field">
  <label class="label" for="password">Password</label>
  <div class="control has-icons-left">
    <input id="password" bind:value={password} class="input" type="password" placeholder="Password" name="password" />
    <span class="icon is-small is-left">
      <i class="fa fa-key"></i>
    </span>
  </div>
</div>
```

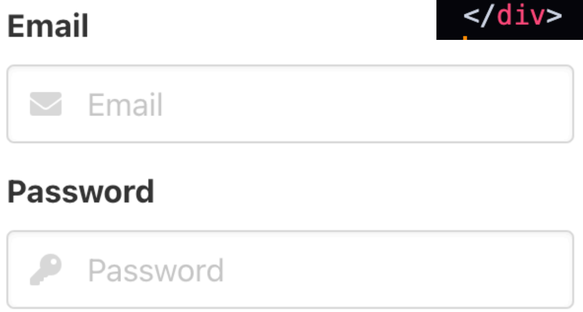


- Display user credentials text fields
- Input controls values bound to js variables
- Variables available as props to parent component

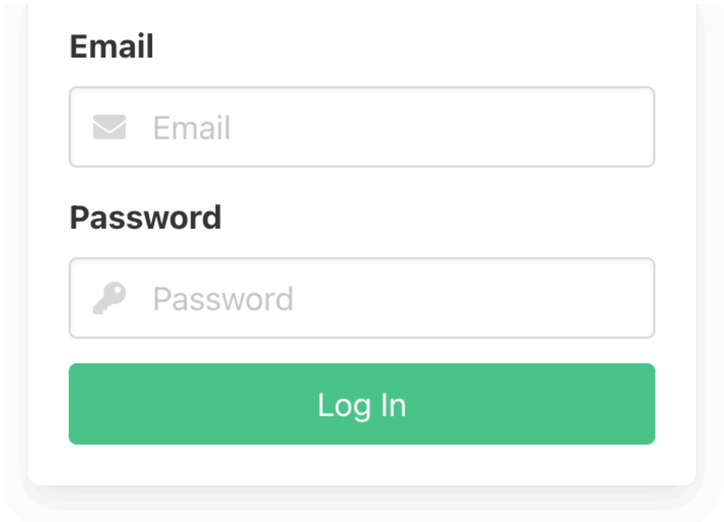
UserCredentials.svelte

```
<script lang="ts">
  let { email = $bindable(''), password = $bindable('') } = $props();
</script>

<div class="field">
  <label class="label" for="email">Email</label>
  <div class="control has-icons-left">
    <input id="email" bind:value={email} class="input" type="text" placeholder="Email" name="email" />
    <span class="icon is-small is-left">…
    </span>
  </div>
</div>
<div class="field">
  <label class="label" for="password">Password</label>
  <div class="control has-icons-left">
    <input id="password" bind:value={password} class="input" type="password" placeholder="Password" name="password" />
    <span class="icon is-small is-left">
      <i class="fa fa-key"></i>
    </span>
  </div>
</div>
```



LoginForm.svelte



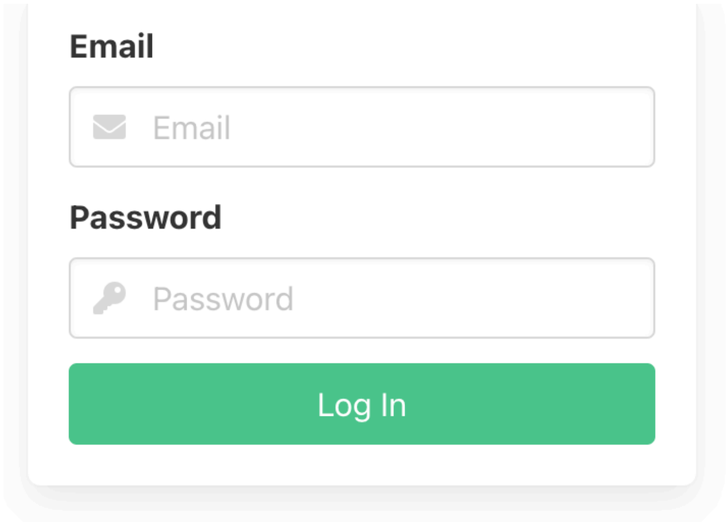

```svelte
<script lang="ts">
  import { goto } from '$app/navigation';
  import { loggedInUser } from '$lib/runes.svelte';
  import Message from '$lib/ui/Message.svelte';
  import UserCredentials from '$lib/ui/UserCredentials.svelte';

  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>

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

LoginForm.svelte



- Login Event Handler: invoked when button pressed

- Variables ‘email’ & ‘password’ bound as UserCredentials props

```svelte
<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>

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

LoginForm.svelte

- Hard code ‘success’ to true
- Goto forces SvelteKit router to load “/donate” route

```svelte
<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>

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

Pages & Components

Creating and assembling Svelte Components