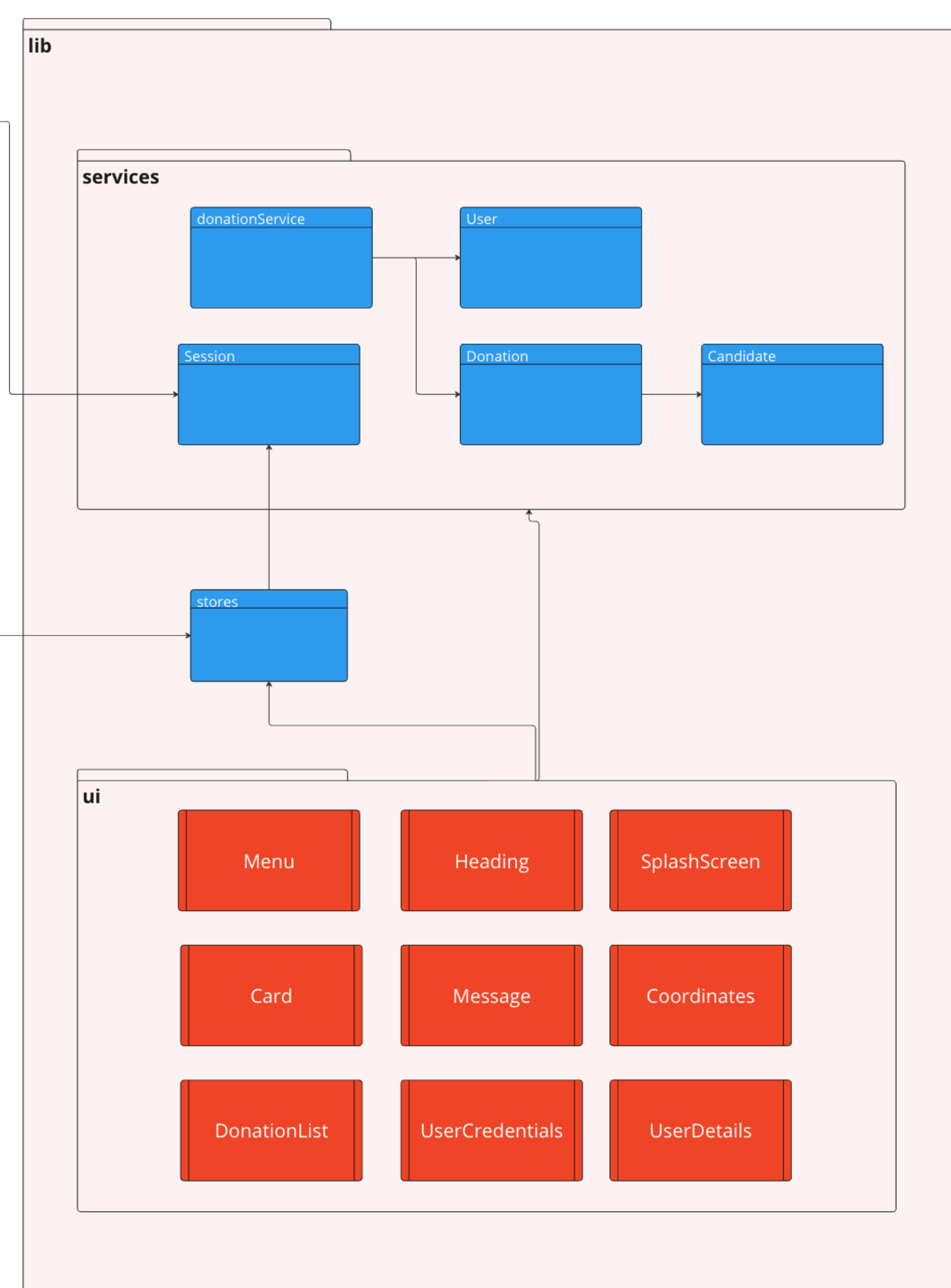
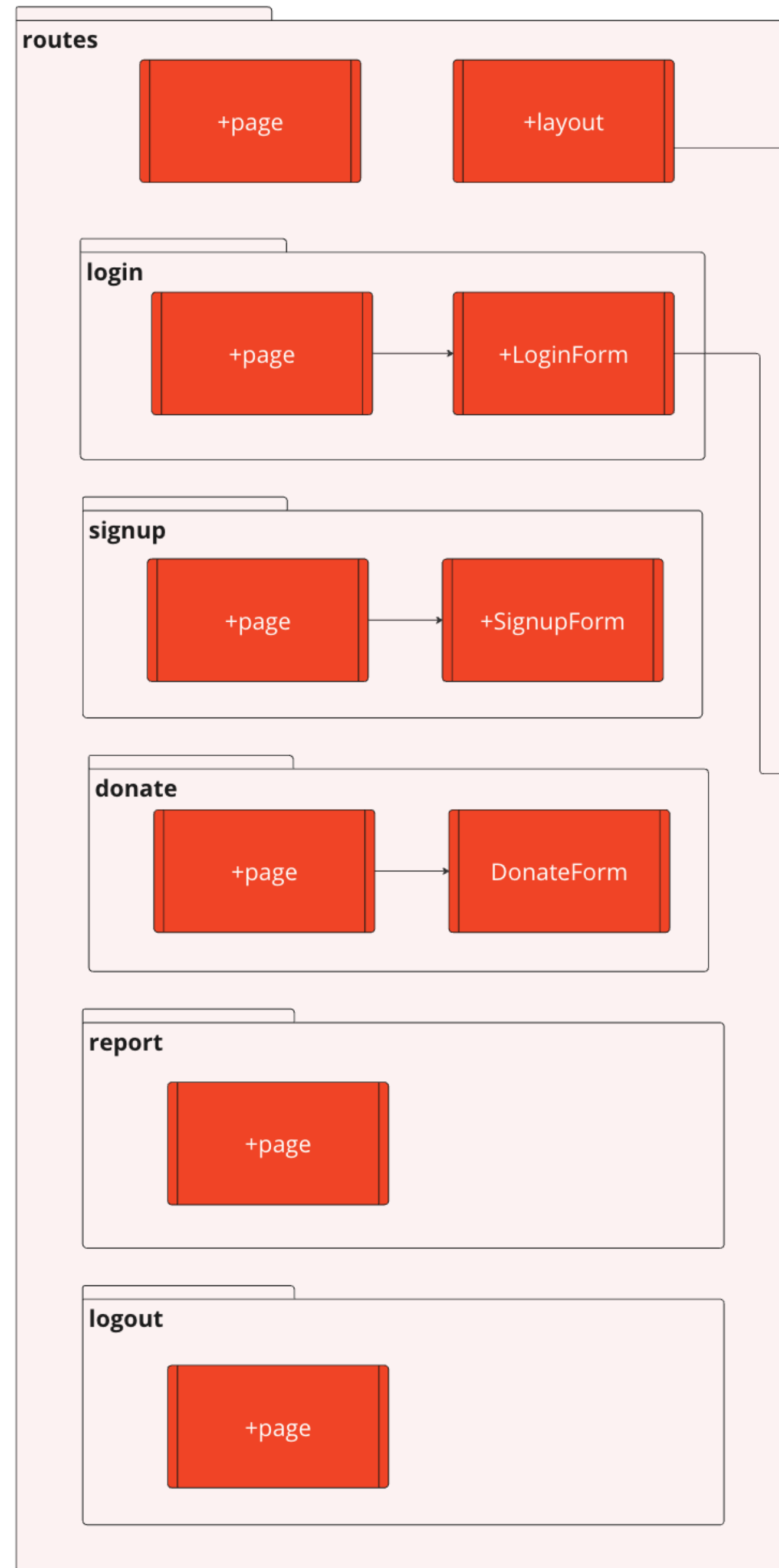
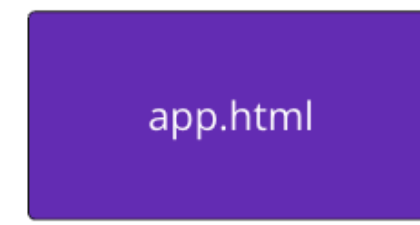
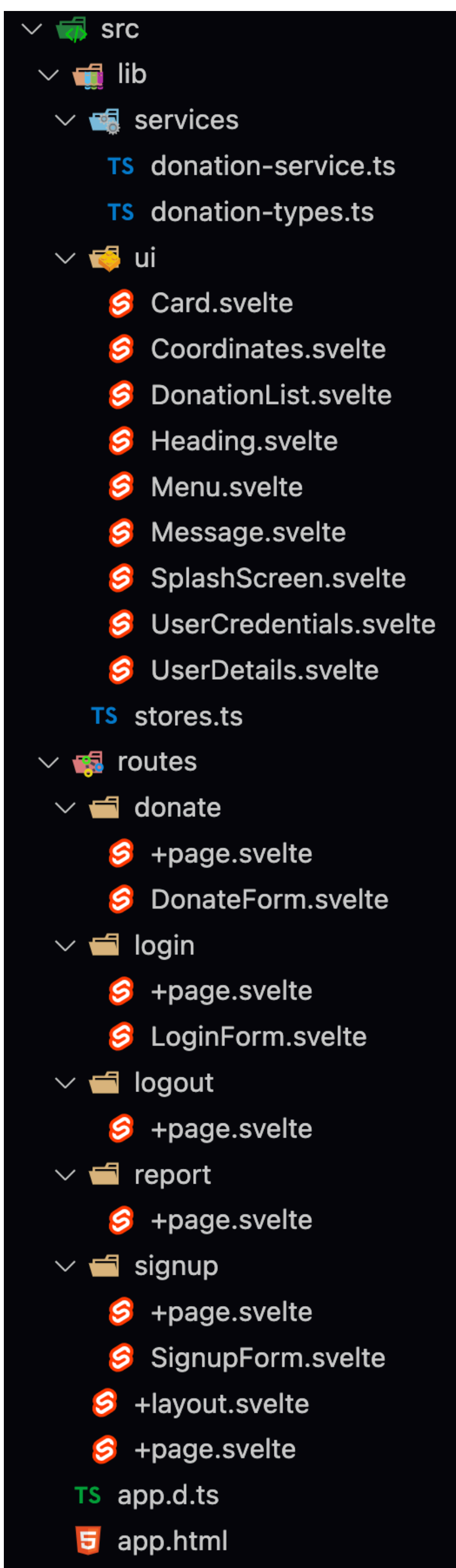


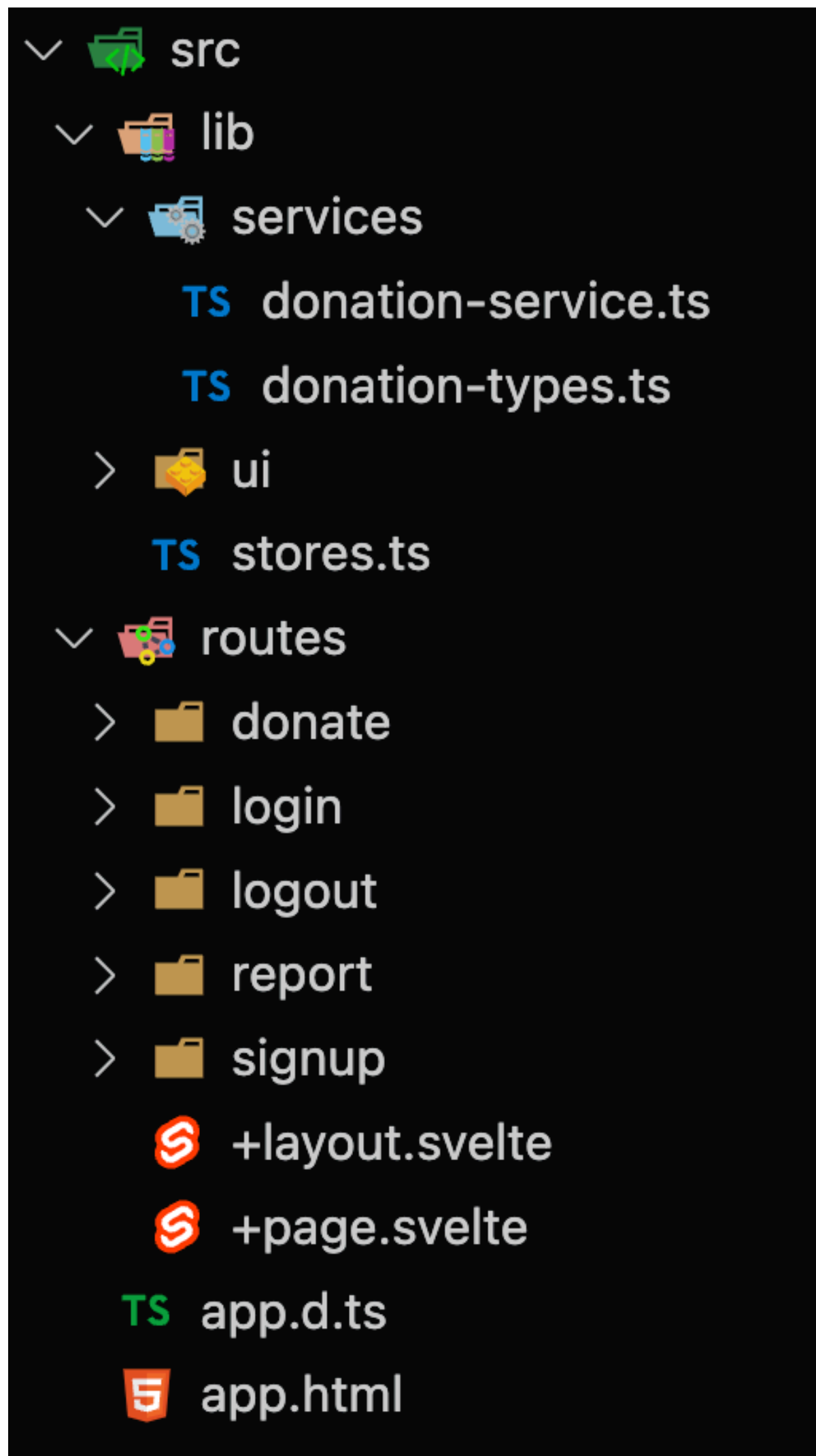
Donate in Svelte



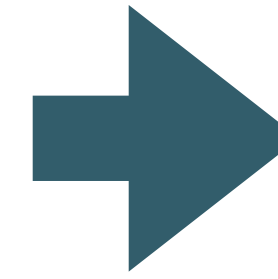
Implement the Donate &
Report front end features



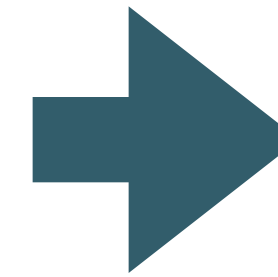
Donation-types



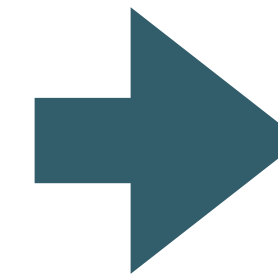
Session: logged in user
name, id + JWT token



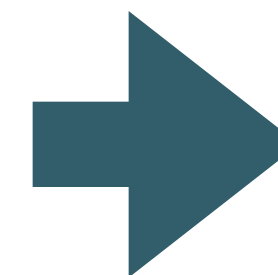
User: user details



Candidate: Candidate
details



Donation: the structure
of a donation object



```
export interface Session {  
  name: string;  
  _id: string;  
  token: string;  
}  
  
export interface User {  
  firstName: string;  
  lastName: string;  
  email: string;  
  password: string;  
  _id: string;  
}  
  
export interface Candidate {  
  firstName: string;  
  lastName: string;  
  office: string;  
  _id: string;  
}  
  
export interface Donation {  
  amount: number;  
  method: string;  
  candidate: Candidate | string;  
  donor: User | string;  
  lat: number;  
  lng: number;  
}
```

Report view

Donations to Date

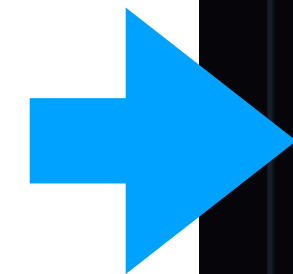
📊 Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	Bart Simpson
90	direct	Simpson, Lisa	Marge Simpson
430	paypal	Flanders, Ned	Homer Simpson

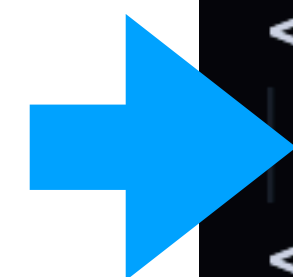
For authenticated user, fetch all donations from Donation API

routes/report/+page.svelte

Retrieve donations
from DonationList



Pass donations to
DonationList
component



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

  subTitle.set("Donations to Date");
  let donations: Donation[] = [];
  onMount(async () => {
    donations = await donationService.getDonations(get(currentSession));
  });
</script>

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


donationService

```
async getDonations(session: Session): Promise<Donation[]> {  
  try {  
    axios.defaults.headers.common["Authorization"] = "Bearer " + session.token;  
    const response = await axios.get(this.baseUrl + "/api/donations");  
    return response.data;  
  } catch (error) {  
    return [];  
  }  
}
```

- Retrieve all donations

DonationList.svelte

- Donations array passed into the DonationList component
- Render donations in a table


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

```
<script lang="ts">
  import type { Donation } from "$lib/services/donation-types";

  export let donations: Donation[];
</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>
          {#if typeof donation.candidate !== "string"}
            {donation.candidate.lastName}, {donation.candidate.firstName}
          {/if}
        </td>
        <td>
          {#if typeof donation.donor !== "string"}
            {donation.donor.lastName}, {donation.donor.firstName}
          {/if}
        </td>
      </tr>
    {/each}
  </tbody>
</table>
```

Donate

 Please Donate

Enter Amount:

0

Select Payment Method:

☐ paypal

☐ direct

Select Candidate:

Simpson,Maggie

▼

Lat

52.160858

Lng

-7.15242

Donate

Please donate

DonateForm.svelte

Please Donate

Enter Amount:

0

Select Payment Method:

☐ paypal

☐ direct

Select Candidate:

Simpson,Maggie

Lat

52.160858

Lng

-7.15242

Donate

Please donate

```
<form on:submit|preventDefault={donate}>
  <div class="field">
    <label class="label" for="amount">Enter Amount:</label>
    <input bind:value={amount} class="input" id="amount" name="amount" type="number" />
  </div>
  <div class="field">
    <div class="control">
      <label class="label" for="amount">Select Payment Method:</label>
      {#each paymentMethods as method}
        <input bind:group={selectedMethod} class="radio" type="radio" value={method} /> {method}
      /each
    </div>
  </div>
  <div class="field">
    <label class="label" for="amount">Select Candidate:</label>
    <div class="select">
      <select bind:value={selectedCandidate}>
        {#each candidateList as candidate}
          <option value={candidate._id}>{candidate.lastName},{candidate.firstName}</option>
        /each
      </select>
    </div>
  </div>
  <Coordinates bind:lat bind:lng />
  <div class="field">
    <div class="control">
      <button class="button is-success is-fullwidth">Donate</button>
    </div>
  </div>
</form>
<div class="box mt-4">
  <div class="content has-text-centered">
    {message}
  </div>
</div>
```

Bindings

➔ **Bindings:** To create a two-way binding between data and the UI.


A variable from the component state (name),
can be bound to a form field:

```
<script>  
let name = ''  
</script>  
  
<input bind:value={name}>
```

isChecked (boolean) is bound to a check box

```
<script>  
let isChecked  
</script>  
  
<input type=checkbox bind:checked={isChecked}>
```

Input

 Please Donate

Enter Amount:

0

Select Payment Method:

☐ paypal ☐ direct

Select Candidate:

Simpson, Maggie ▼

Lat 52.160858 Lng -7.15242

Donate

Please donate

```
let amount = 0;
```

```
<div class="field">  
  <label class="label" for="amount">Enter Amount:</label>  
  <input bind:value={amount} class="input" id="amount" name="amount" type="number" />  
</div>
```

Logic

- In Svelte components you can create a loop using the `{#each}{/each}` syntax
- Conditional logic is via the `{#if}{/if}`

→ **Logic:** To specify way of expressing logic, like conditionals and loops.

```
<script>
let goodDogs = ['Roger', 'Syd']
</script>

{#each goodDogs as goodDog}
  <li>{goodDog}</li>
{/each}
```

```
<script>
let isRed = true
</script>
```

```
{#if isRed}
  <p>Red</p>
{:else}
  <p>Not red</p>
{/if}
```

Radio

Please Donate

Enter Amount:

Select Payment Method:
☐ paypal ☐ direct

Select Candidate:

Lat Lng

Donate

Please donate

```
let paymentMethods = ["paypal", "direct"];  
let selectedMethod = "";
```

```
<div class="field">  
  <div class="control">  
    <label class="label" for="amount">Select Payment Method:</label>  
    {#each paymentMethods as method}  
      <input bind:group={selectedMethod} class="radio" type="radio" value={method} /> {method}  
    {/each}  
  </div>  
</div>
```


Select / Dropdown

Please Donate

Enter Amount:

Select Payment Method:
☐ paypal ☐ direct

Select Candidate:

Simpson,Maggie ▾

Lat Lng

Donate


Please donate

```
export let candidateList: Candidate[] = [];
```

```
let selectedCandidate = "";
```

```
<div class="field">
  <label class="label" for="amount">Select Candidate:</label>
  <div class="select">
    <select bind:value={selectedCandidate}>
      {#each candidateList as candidate}
        <option value={candidate._id}>{candidate.lastName},{candidate.firstName}</option>
      {/each}
    </select>
  </div>
</div>
```

Text

 Please Donate

Enter Amount:

Select Payment Method:

☐ paypal ☐ direct

Select Candidate:

Simpson,Maggie

Lat

Lng

Donate

Please donate

```
let message = "Please donate";
```

```
<div class="box mt-4">
  <div class="content has-text-centered">
    {message}
  </div>
</div>
```

Events

➔ **Events:** Define a listener for a DOM event directly


To listen to the click event, pass a function to the on:click attribute

Svelte passes the event handler as the argument of the function

```
<script>
const doSomething = () => {
  alert('clicked')
}
</script>

<button on:click={doSomething}>Click me</button>
```

Button

 Please Donate

Enter Amount:

0

Select Payment Method:

☐ paypal

☐ direct

Select Candidate:

Simpson,Maggie

Lat

52.160858

Lng

-7.15242

Donate

Please donate

```
async function donate() {
  console.log(`Just donated: ${amount} to ${selectedCandidate} via ${selectedMethod} payment`);
  console.log(`lat: ${lat}, lng: ${lng}`);
}
```

```
<form on:submit|preventDefault={donate}>
```

```
<div class="field">
  <div class="control">
    <button class="button is-success is-fullwidth">Donate</button>
  </div>
</div>
</form>
```

```

async function donate() {
  if (selectedCandidate && amount && selectedMethod) {
    const candidate = candidateList.find((candidate) => candidate._id === selectedCandidate);
    if (candidate) {
      const donation: Donation = {
        amount: amount,
        method: selectedMethod,
        candidate: selectedCandidate,
        lat: lat,
        lng: lng,
        donor: $currentSession._id
      };
      const success = await donationService.donate(donation, get(currentSession));
      if (!success) {
        message = "Donation not completed – some error occurred";
        return;
      }
      donation.candidate = candidate;
      donation.donor = $currentSession.name;
      message = `Thanks! You donated ${amount} to ${candidate.firstName} ${candidate.lastName}`;
    }
  } else {
    message = "Please select amount, method and candidate";
  }
}

```



```

async function donate() {
  if (selectedCandidate && amount && selectedMethod) {
    const candidate = candidateList.find((candidate) => candidate._id === selectedCandidate);
    if (candidate) {
      const donation: Donation = {
        amount: amount,
        method: selectedMethod,
        candidate: selectedCandidate,
        lat: lat,
        lng: lng,
        donor: $currentSession._id
      };
      const success = await donationService.donate(donation, get(currentSession));
      if (!success) {
        message = "Donation not completed – some error occurred";
        return;
      }
      donation.candidate = candidate;
      donation.donor = $currentSession.name;
      message = `Thanks! You donated ${amount} to ${candidate.firstName} ${candidate.lastName}`;
    }
  } else {
    message = "Please select amount, method and candidate";
  }
}

```

Find selected candidate

Create donation object

Invoke Donation API

Update donation success msg

Donate in Svelte



Implement the Donate &
Report front end features