Charting Donations

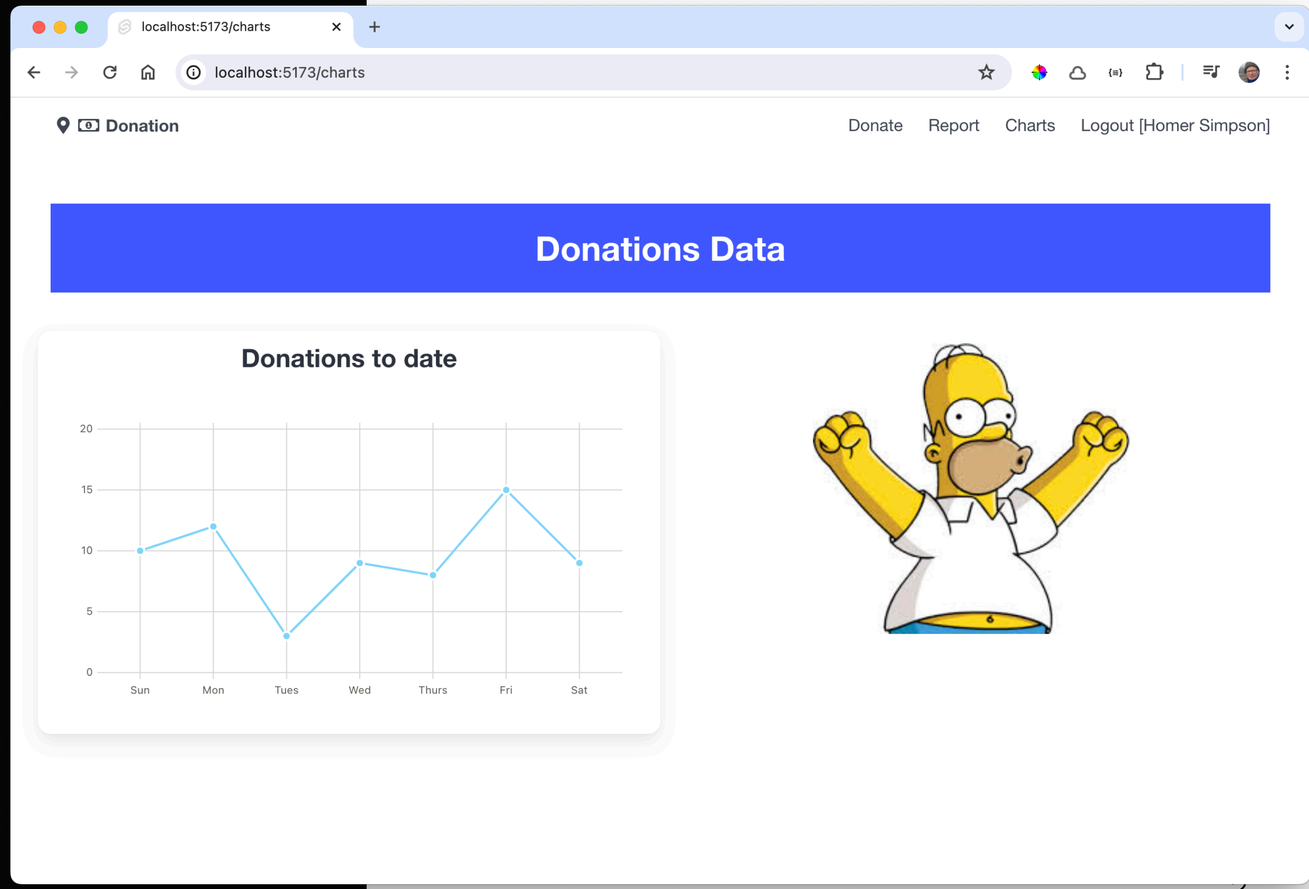
Plotting donations by candidate & payment method

```svelte
<script lang="ts">
  // @ts-ignore
  import Chart from "svelte-frappe-charts";

  const chartData = {
    labels: ["Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat"],
    datasets: [
      {
        values: [10, 12, 3, 9, 8, 15, 9]
      }
    ]
  };
</script>

<div class="columns">
  <div class="column box has-text-centered">
    <h1 class="title is-4">Donations to date</h1>
    <Chart data={chartData} type="line" />
  </div>
  <div class="column has-text-centered">
    <img alt="Homer" src="/homer4.jpeg" width="300"
  </div>
</div>
```

routes/charts/+page.svelte

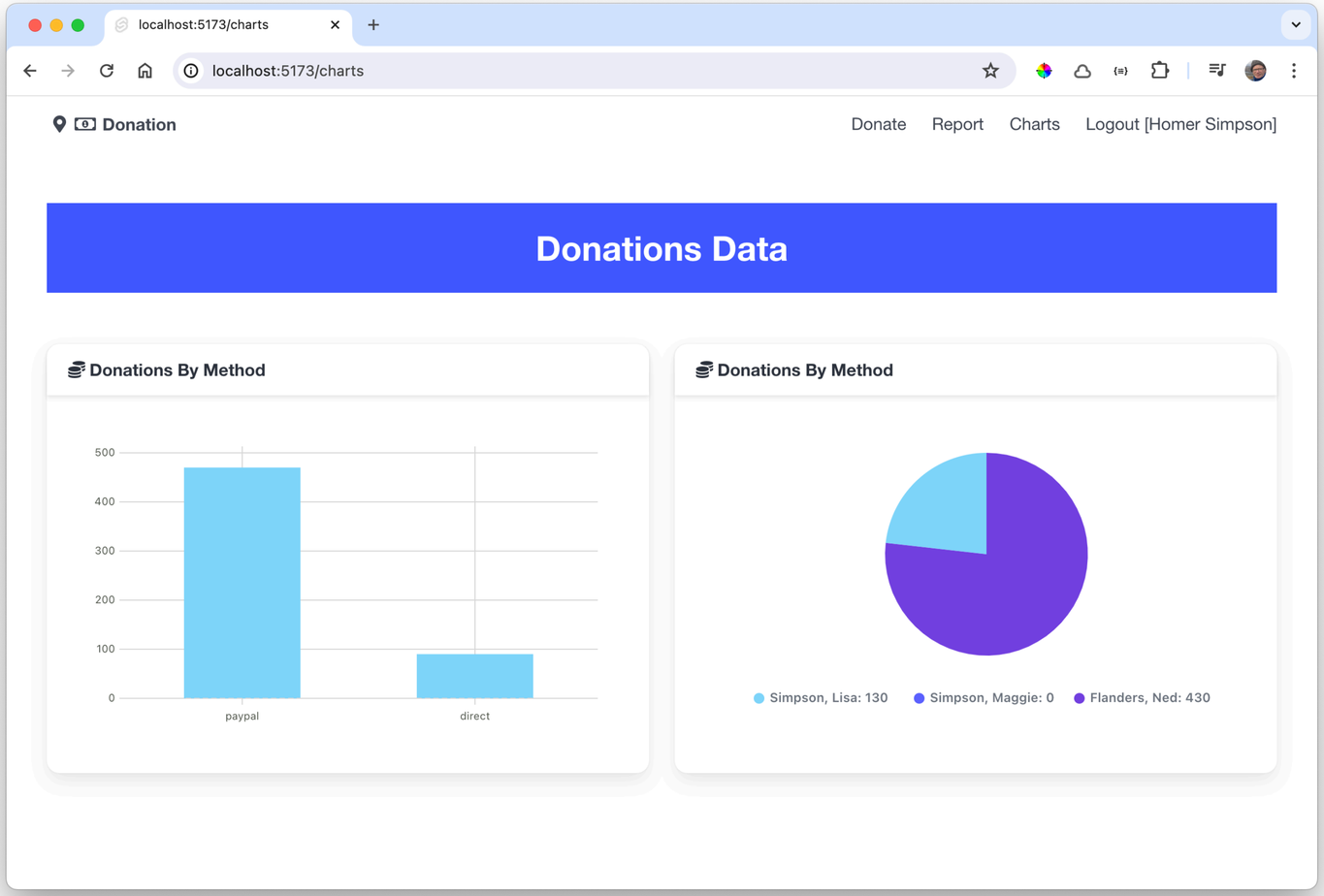


- When the page loads
- Retrieve all donations
- Compute totals of ‘paypal’ & ‘direct’ payments

```
const totalByMethod = {
  labels: ["paypal", "direct"],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};

subTitle.set("Donations Data");

onMount(async () => {
  const donationList = await donationService.getDonations(get(currentSession));
  donationList.forEach((donation) => {
    if (donation.method == "paypal") {
      totalByMethod.datasets[0].values[0] += donation.amount;
    } else if (donation.method == "direct") {
      totalByMethod.datasets[0].values[1] += donation.amount;
    }
  });
});
```

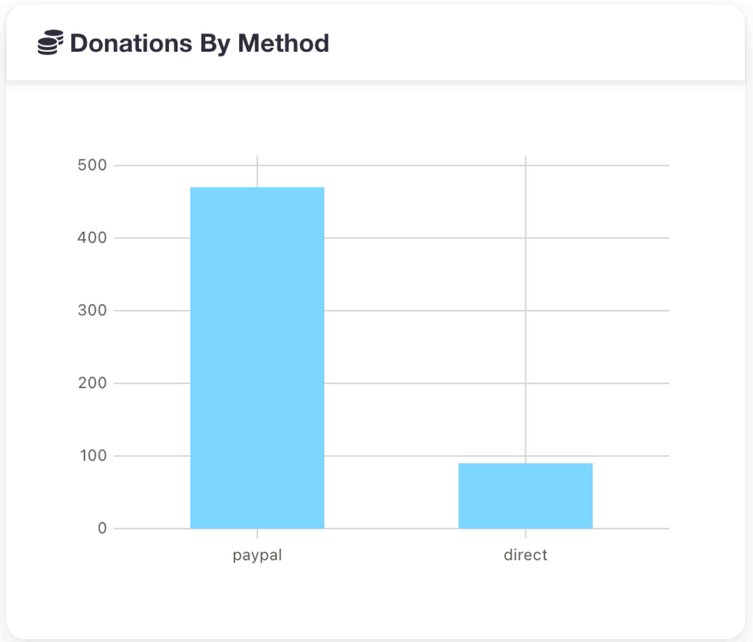


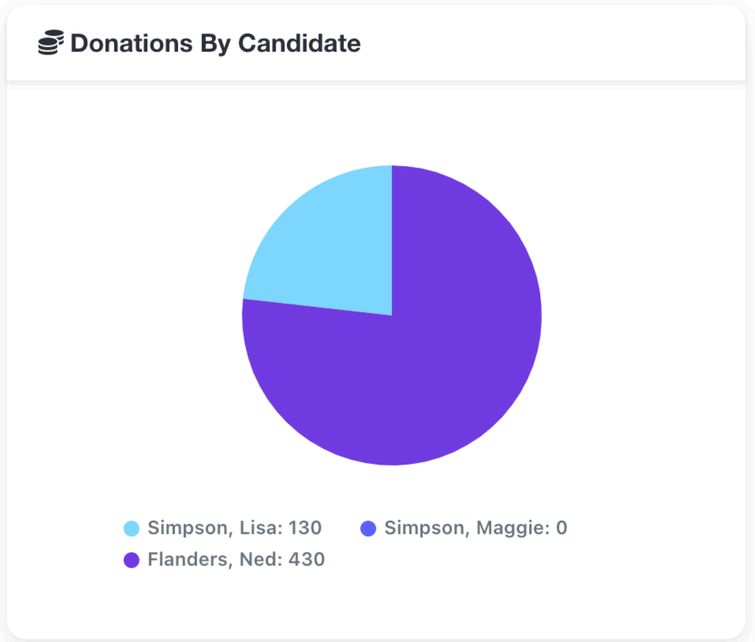

```
const totalByMethod = {
  labels: ["paypal", "direct"],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

```
<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={totalByMethod} type="bar" />
    </Card>
  </div>
  <div class="column has-text-centered">
    <Card title="Donations By Method">
      <Chart data={totalByMethod} type="pie" />
    </Card>
  </div>
</div>
```

- Donations by payment methods: Pie + Bar
- 2 Chart components bound to the same data

- Donations by may payment method





- Donations by Candidate

```
const totalByMethod = {
  labels: ["paypal", "direct"],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

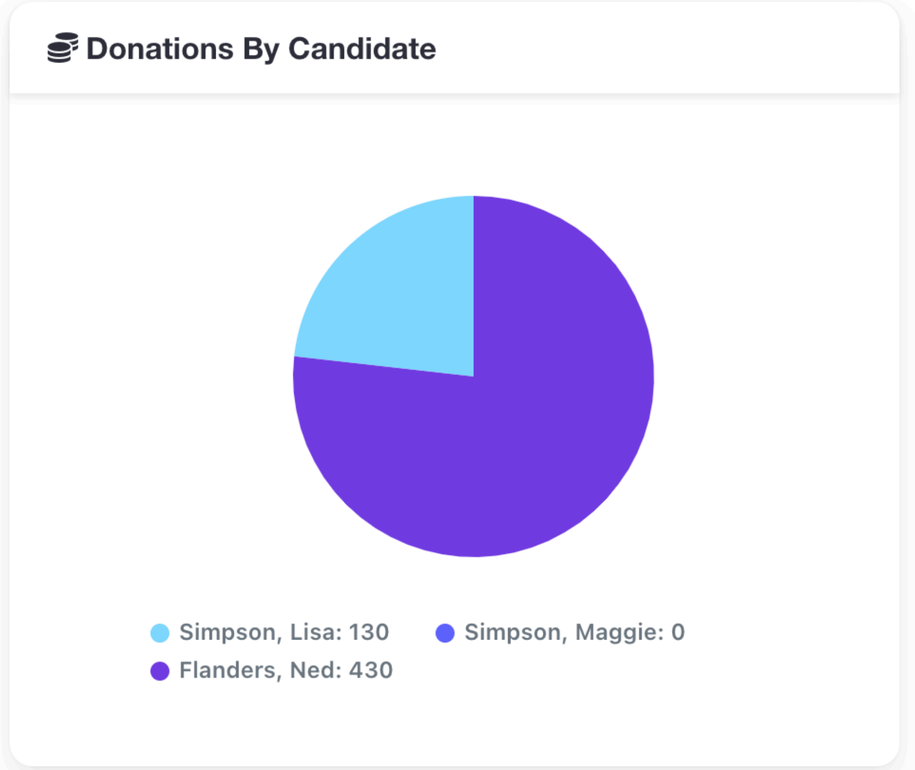
```
const donationsByCandidate = {
  labels: [],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

```
const donationsByCandidate = {
  labels: [],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```


```
const candidates = await donationService.getCandidates(get(currentSession));
donationsByCandidate.labels = [];
candidates.forEach((candidate) => {
  // @ts-ignore
  donationsByCandidate.labels.push(`${candidate.lastName}, ${candidate.firstName}`);
  donationsByCandidate.datasets[0].values.push(0);
});
```

- Retrieve the candidates and populate candidate labels

```
const donationsByCandidate = {
  labels: [],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

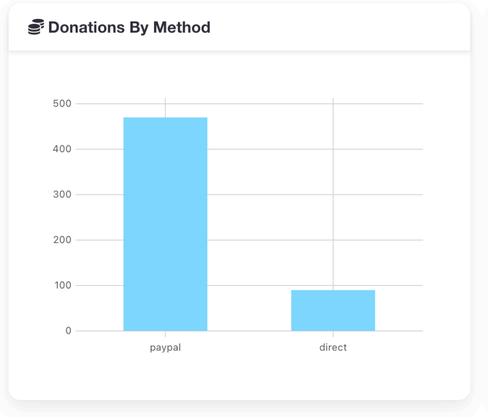


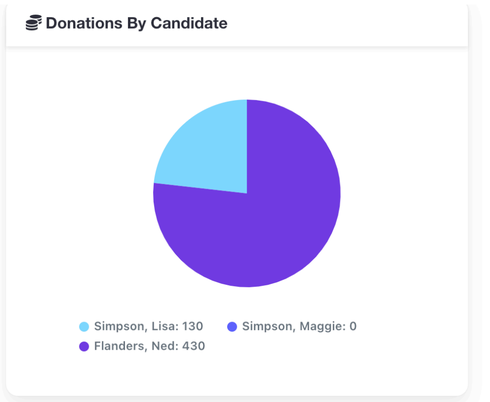

```
  candidates.forEach((candidate, i) => {
    donationList.forEach((donation) => {
      // @ts-ignore
      if (donation.candidate._id == candidate._id) {
        donationsByCandidate.datasets[0].values[i] += donation.amount;
      }
    });
  });
});
```

```
const totalByMethod = {
  labels: ["paypal", "direct"],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

```
const donationsByCandidate = {
  labels: [],
  datasets: [
    {
      values: [0, 0]
    }
  ]
};
```

```
<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={totalByMethod} type="bar" />
    </Card>
  </div>
  <div class="column has-text-centered">
    <Card title="Donations By Method">
      <Chart data={donationsByCandidate} type="pie" />
    </Card>
  </div>
</div>
```






```
const donationList = await donationService.getDonations(get(currentSession));
donationList.forEach((donation) => {
  if (donation.method == "paypal") {
    totalByMethod.datasets[0].values[0] += donation.amount;
  } else if (donation.method == "direct") {
    totalByMethod.datasets[0].values[1] += donation.amount;
  }
});
```

```
const candidates = await donationService.getCandidates(get(currentSession));
donationsByCandidate.labels = [];
candidates.forEach((candidate) => {
  // @ts-ignore
  donationsByCandidate.labels.push(`${candidate.lastName}, ${candidate.firstName}`);
  donationsByCandidate.datasets[0].values.push(0);
});
candidates.forEach((candidate, i) => {
  donationList.forEach((donation) => {
    // @ts-ignore
    if (donation.candidate._id == candidate._id) {
      donationsByCandidate.datasets[0].values[i] += donation.amount;
    }
  });
});
```

donation-utils.ts

```ts
export interface DataSet {
  labels: string[];
  datasets: [{ values: number[] }];
}
```

- DataSet captures structure of frappe data

```ts
import type { Candidate, DataSet, Donation } from "$lib/types/donation-types";

export function generateByMethod(donationList: Donation[]): DataSet {…
}

export function generateByCandidate(donationList: Donation[], candidates: Candidate[]): DataSet {…
}
```

- Refactor calculations in separate utility module

donation-utils.ts

```ts
export interface DataSet {
  labels: string[];
  datasets: [{ values: number[] }];
}
```

- DataSet captures structure of frappe data

```ts
export function generateByMethod(donationList: Donation[]): DataSet {
  const totalByMethod: DataSet = {
    labels: ["paypal", "direct"],
    datasets: [
      {
        values: [0, 0]
      }
    ]
  };

  donationList.forEach((donation) => {
    if (donation.method == "paypal") {
      totalByMethod.datasets[0].values[0] += donation.amount;
    } else if (donation.method == "direct") {
      totalByMethod.datasets[0].values[1] += donation.amount;
    }
  });

  return totalByMethod;
}
```

- Refactor calculations in separate utility module

donation-utils.ts

```ts
export interface DataSet {
  labels: string[];
  datasets: [{ values: number[] }];
}
```

- DataSet captures structure of frappe data

```ts
export function generateByCandidate(donationList: Donation[], candidates: Candidate[]): DataSet {
  const donationsByCandidate: DataSet = {
    labels: [],
    datasets: [
      {
        values: [0, 0]
      }
    ]
  };

  donationsByCandidate.labels = [];
  candidates.forEach((candidate) => {
    donationsByCandidate.labels.push(`${candidate.lastName}, ${candidate.firstName}`);
    donationsByCandidate.datasets[0].values.push(0);
  });

  candidates.forEach((candidate, i) => {
    donationList.forEach((donation) => {
      if (typeof donation.candidate !== "string") {
        if (donation.candidate._id == candidate._id) {
          donationsByCandidate.datasets[0].values[i] += donation.amount;
        }
      }
    });
  });

  return donationsByCandidate;
}
```

- Refactor calculations in separate utility module

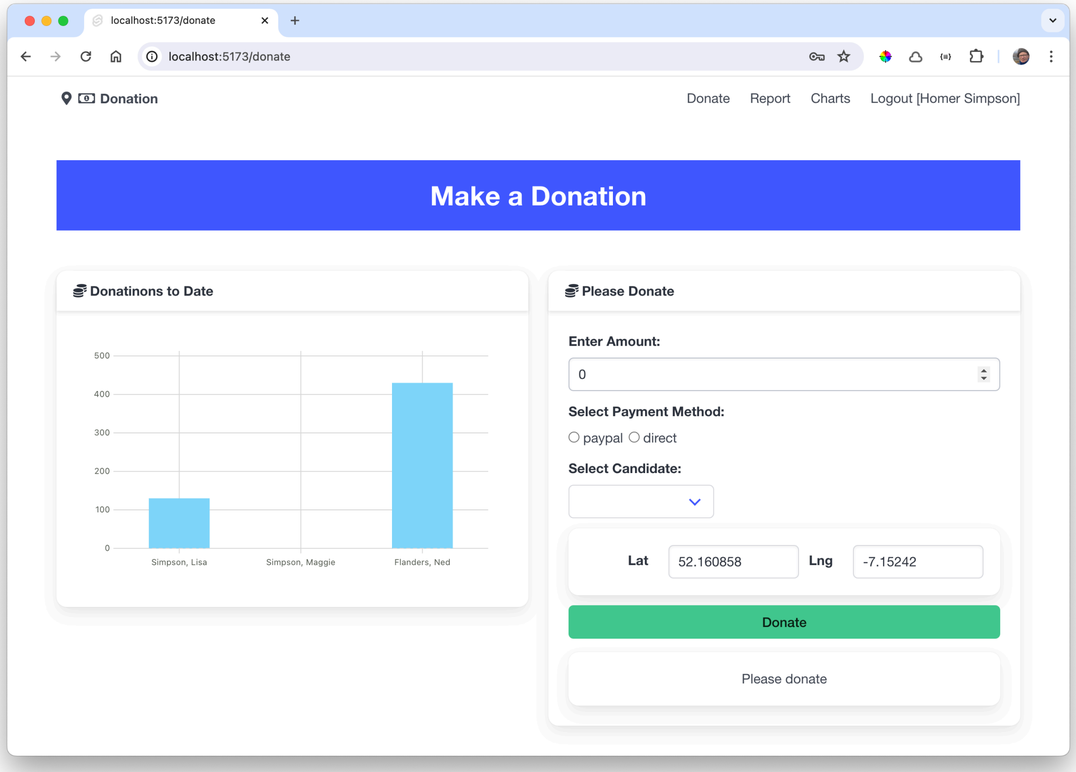
Display Chart on Donate Page

```
let candidateList: Candidate[] = [];
let donations: Donation[] = [];
let donationsByCandidate: DataSet;
subTitle.set("Make a Donation");

onMount(async () => {
  candidateList = await donationService.getCandidates(get(currentSession));
  donations = await donationService.getDonations(get(currentSession));
  const candidates = await donationService.getCandidates(get(currentSession));
  donationsByCandidate = generateByCandidate(donations, candidates);
});
```

- donation-utils facilitates simple inclusion of charts on diverse pages

```
<div class="columns">
  <div class="column">
    <Card title="Donatinons to Date">
      <Chart data={donationsByCandidate} type="bar" />
    </Card>
  </div>
  <div class="column">
    <Card title="Please Donate">
      <DonateForm {candidateList} />
    </Card>
  </div>
</div>
```



Charting Donations

Plotting donations by candidate & payment method