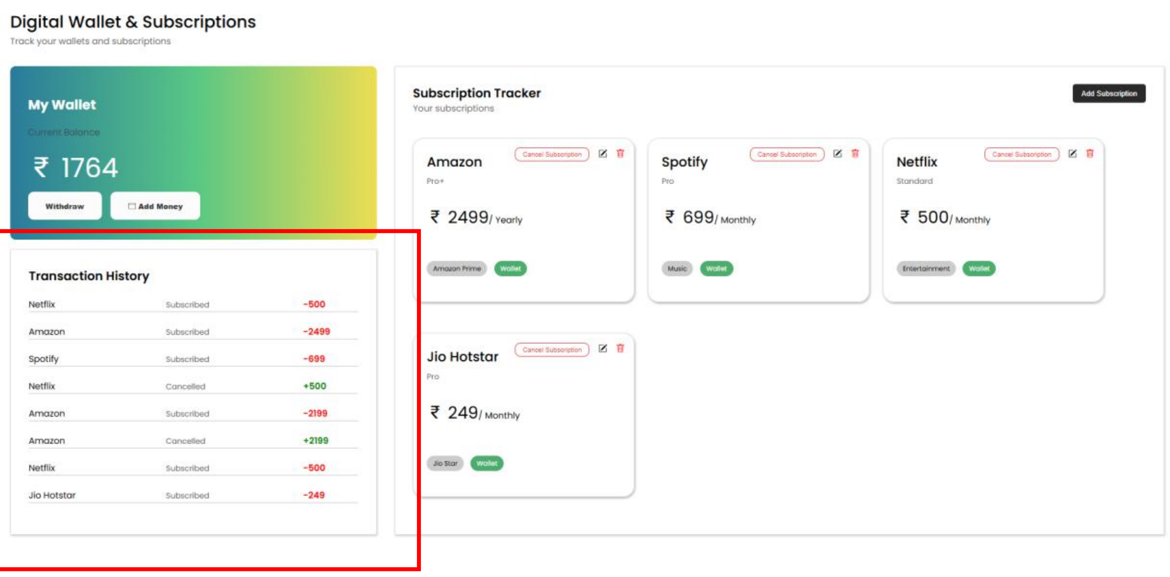
OUTPUT:

Task 6: Transaction history for wallet.


Digital Wallet & Subscriptions

Track your wallets and subscriptions

My Wallet

Current Balance

₹ 1764

Withdraw

Add Money

Transaction History

Netflix	Subscribed	-500
Amazon	Subscribed	-2499
Spotify	Subscribed	-699
Netflix	Cancelled	+500
Amazon	Subscribed	-2199
Amazon	Cancelled	+2199
Netflix	Subscribed	-500
Jio Hotstar	Subscribed	-249

Subscription Tracker

Your subscriptions

Add Subscription

Amazon — Pro+ — ₹ 2499/ Yearly — Amazon Prime — Wallet — Cancel Subscription

Spotify — Pro — ₹ 699/ Monthly — Music — Wallet — Cancel Subscription

Netflix — Standard — ₹ 500/ Monthly — Entertainment — Wallet — Cancel Subscription

Jio Hotstar — Pro — ₹ 249/ Monthly — Jio Star — Wallet — Cancel Subscription

CODE:

Index.html

```
<!DOCTYPE html>

<html>


<head>

  <meta charset="utf-8">

  <title>AssessmentApp</title>

  <meta name="description" content="">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  {{content-for "head"}}


  <link integrity="" rel="stylesheet" href="{{rootURL}}assets/vendor.css">
  <link integrity="" rel="stylesheet" href="{{rootURL}}assets/assessment-app.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
    href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,4
00;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&
display=swap"
    rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">


  {{content-for "head-footer"}}
</head>


<body>
  {{content-for "body"}}


  <script src="{{rootURL}}assets/vendor.js"></script>
  <script src="{{rootURL}}assets/assessment-app.js"></script>


  {{content-for "body-footer"}}
</body>


</html>
```

Templates/application.hbs

```
{{page-title "AssessmentApp"}}


{{outlet}}


<span class="head-text">Digital Wallet &amp; Subscriptions</span><br>

<span class="head-desc-text">Track your wallets and subscriptions</span>


<div class="container">
  <div class="left">
    <div class="my-wallet">
      <h2>My Wallet</h2>
      <p class="amount-title">Current Balance</p>
      <span class="amount"><i class="bi bi-currency-rupee"></i> {{this.amount}}</span>
      <div class="wallet-btn-container">
        <button class="wallet-btn withdraw" type="button">Withdraw</button>
        <button class="wallet-btn add-money" type="button" {{on "click"
this.toggleModal}}><i
            class="bi bi-wallet"></i> Add Money</button>
      </div>
    </div>


    <div class="transaction-history">
      <span class="heading">Transaction History</span>
      {{#each this.history as |hist|}}
      <div class="transaction">
        <span class="hist-name">{{hist.name}}</span>
        <span class="hist-desc">{{hist.transaction}}</span>
```

```
            <span class={{if (eq hist.transaction "Subscribed") "del" "plus"}}>{{if (eq
hist.transaction "Subscribed") "-" "+"}}{{hist.subamount}}</span>

          </div>

          {{/each}}

        </div>


        {{#if this.isModelOpen}}

        <div class="overlay"></div>

        <div class="add-modal">

          <i class="bi bi-x-lg close-icon" {{on "click" this.toggleModal}}></i>

          <label for="amount-value">Amount</label>

          <input type="number" placeholder="Enter Amount" id="amount-value">

          <button class="add-btn" type="button" {{on "click" this.addToWallet}}>Add to
Wallet</button>

        </div>


        {{/if}}

    </div>


    <div class="subscription-tracker">

        <div class="subscription-tracker-header">

          <div class="subscription-tracker-head">

            <span class="heading">Subscription Tracker</span>

            <span class="description">Your subscriptions</span>

          </div>

          <button class="add-sub-btn" type="button" {{on "click" this.toggleSubModal}}>Add
Subscription</button>

        </div>
        <div class="subscriptions">
          {{#each this.subscriptions as |subscription index|}}
          <div class="subscription-card">
            <i class="bi bi-pencil-square edit-icon" {{on "click" (fn this.editSubscription
index)}}></i>
            <i class="bi bi-trash edit-icon del" {{on "click" (fn this.dltSubscription index)}}></i>
            <button class="edit-icon cancel" {{on "click" (fn this.cancelSubscription
index)}}>Cancel
          Subscription</button>
            <span class="name">{{subscription.name}}</span>
            <span class="sub-plan">{{subscription.subplan}}</span>
            <div>
          <span class="sub-amount"><i class="bi bi-currency-rupee"></i>
{{subscription.subamount}}</span>/
          <span class="billing-cycle">{{subscription.billcycle}}</span>
            </div>
            <div>
          <span class="category">{{subscription.category}}</span>
          <span class="payment-method">{{subscription.paymethod}}</span>
            </div>
          </div>

          {{else}}
          <div class="empty-subscription">
            <img src="/nosub.png" alt="img" style="width: 500px; height:500px;
margin:1rem">
            <p style="font-size: 18px; font-weight:600">No Subscriptions Found</p>
          </div>
          {{/each}}
```

```
        </div>


        {{#if this.isSubModelOpen}}
        <div class="overlay"></div>
        <div class="add-sub-modal">
            <i class="bi bi-x-lg close-icon" {{on "click" this.toggleSubModal}}></i>
            <label for="amount-value">Name</label>
            <input type="text" value={{this.name}} placeholder="Enter Name" id="name" {{on
    "input" this.updateName}}>
            <PowerSelect @selected={{this.subplan}} @options={{this.plans}}
    @onChange={{this.updatePlan}}
                @labelText="Subscription Plan" as |plan|>
                {{plan}}
            </PowerSelect>
            <PowerSelect @selected={{this.billcycle}} @options={{this.billings}}
    @onChange={{this.updateBilling}}
                @labelText="Billing Cycle" as |billing|>
                {{billing}}
            </PowerSelect>
            <label for="sub-amount">Amount</label>
            <input type="number" value={{this.subamount}} placeholder="Enter Amount"
    id="sub-amount" {{on "input"
                this.updateAmount}}>
            <PowerSelect @selected={{this.category}} @options={{this.categories}}
    @onChange={{this.updateCategory}}
                @labelText="Category" as |category|>
                {{category}}
            </PowerSelect>
```

```
          <PowerSelect @selected={{this.paymethod}} @options={{this.paymentMethod}}
@onChange={{this.updatePayment}}
              @labelText="Payment Method" as |pay|>
              {{pay}}
          </PowerSelect>
          <button class="add-sub-btn add" type="button" {{on "click"
this.addSubscription}}>{{if this.isEditing
              "Update Subscription" "Add Subscription"}} </button>
      </div>

      {{/if}}

   </div>
</div>

<BasicDropdownWormhole />
```

Controllers/application.js

```
import Controller from '@ember/controller';
import { tracked } from '@glimmer/tracking';
import { action } from '@ember/object'

export default class ApplicationController extends Controller {

   constructor() {
      super(...arguments)
      this.loadAmount();
      this.loadSubscription();
```

```
    this.loadHistory();
  }

  @tracked amount = 0;
  @tracked isModelOpen = false;
  @tracked isSubModelOpen = false;
  @tracked isEditing = false;
  @tracked editIndex = null;

  @tracked subscriptions = [];

  @tracked name = '';
  @tracked subplan = '';
  @tracked billcycle = '';
  @tracked subamount = '';
  @tracked category = '';
  @tracked paymethod = '';

  plans = ['Standard', 'Pro', 'Pro+'];
  billings = ['Weekly', 'Monthly', 'Yearly'];
  categories = ['Entertainment', 'Music', 'Amazon Prime', 'Jio Star'];
  paymentMethod = ['UPI', 'Debit Card', 'Net Banking', 'Wallet'];

  @tracked history = [];

  @action toggleModal() {
    this.isModelOpen = !this.isModelOpen;
  }
```

```
@action toggleSubModal() {
   this.isSubModelOpen = !this.isSubModelOpen;
   this.resetInputs();
}

@action addToWallet() {
   const value = document.getElementById('amount-value').value;
   this.amount = parseInt(this.amount) + parseInt(value);
   localStorage.setItem("amount", this.amount);
   this.toggleModal();
}

@action loadAmount() {
   const resAmount = localStorage.getItem("amount") || 0;
   this.amount = resAmount;
}

@action updateName(event) {
   this.name = event.target.value;
}

@action updatePlan(plan) {
   this.subplan = plan;
}

@action updateBilling(bill) {
   this.billcycle = bill;
```

```
}

@action updateAmount(event) {
   this.subamount = event.target.value;
}

@action updateCategory(cat) {
   this.category = cat;
}

@action updatePayment(pay) {
   this.paymethod = pay;
}

@action addSubscription() {
   let subscription = {
      name: this.name,
      subplan: this.subplan,
      billcycle: this.billcycle,
      subamount: this.subamount,
      category: this.category,
      paymethod: this.paymethod
   }

   if (!this.isEditing) {
      if (subscription.paymethod == "Wallet") {
         if (parseInt(this.subamount) <= parseInt(this.amount)) {
            this.amount = this.amount - this.subamount;
```

```
                    localStorage.setItem("amount", this.amount);
                    this.subscriptions = this.subscriptions ? [...this.subscriptions, subscription] :
[subscription];
                    localStorage.setItem("subscriptions", JSON.stringify(this.subscriptions));
                    this.loadSubscription();
                    this.toggleSubModal();
                    this.resetInputs();
                    subscription.transaction = "Subscribed";
                    this.history = this.history ? [...this.history, subscription] : [subscription];
                    localStorage.setItem("history", JSON.stringify(this.history));
                    this.loadHistory();
                } else {
                    alert("Insufficient Wallet Balance");
                }
            } else {
                this.subscriptions = this.subscriptions ? [...this.subscriptions, subscription] :
[subscription];;
                localStorage.setItem("subscriptions", JSON.stringify(this.subscriptions));
                this.loadSubscription();
                this.toggleSubModal();
                this.resetInputs();
            }


        } else {
            this.subscriptions[this.editIndex] = subscription;
            this.isEditing = !this.isEditing;
            localStorage.setItem("subscriptions", JSON.stringify(this.subscriptions))
            this.toggleSubModal();
            this.loadSubscription();
        this.resetInputs();
    }
}

@action editSubscription(index) {
    this.editIndex = index;
    this.isEditing = !this.isEditing;
    this.toggleSubModal();
    this.name = this.subscriptions[index].name;
    this.subplan = this.subscriptions[index].subplan;
    this.billcycle = this.subscriptions[index].billcycle;
    this.subamount = this.subscriptions[index].subamount;
    this.category = this.subscriptions[index].category;
    this.paymethod = this.subscriptions[index].paymethod;
}

@action dltSubscription(index) {
    this.subscriptions.splice(index, 1);
    this.subscriptions = [...this.subscriptions];
    localStorage.setItem("subscriptions", JSON.stringify(this.subscriptions));
    this.loadSubscription();

}

@action cancelSubscription(index) {

    let cancelled = {
        name: this.subscriptions[index].name,
        subplan: this.subscriptions[index].subplan,
        billcycle: this.subscriptions[index].billcycle,
        subamount: this.subscriptions[index].subamount,
        category: this.subscriptions[index].category,
        paymethod: this.subscriptions[index].paymethod,
        transaction:"Cancelled"
    }

    if (this.subscriptions[index].paymethod == "Wallet") {
        this.amount = parseInt(this.amount) + parseInt(this.subscriptions[index].subamount);
        localStorage.setItem("amount", this.amount);
        this.history = this.history ? [...this.history, cancelled] : [cancelled];
        localStorage.setItem("history", JSON.stringify(this.history));
        this.loadHistory();
        alert("Amount refunded to wallet");
    }
    this.subscriptions.splice(index, 1);
    this.subscriptions = [...this.subscriptions];
    localStorage.setItem("subscriptions", JSON.stringify(this.subscriptions));
    this.loadSubscription();
}

@action resetInputs() {
    this.name = '';
    this.subplan = '';
    this.billcycle = '';
    this.subamount = '';
    this.category = '';
```

```
        this.paymethod = '';
    }

    @action loadSubscription() {
        const resSubscription = JSON.parse(localStorage.getItem("subscriptions"));
        this.subscriptions = resSubscription;
        console.log(this.subscriptions);
    }

    @action loadHistory() {
        const resHistory = JSON.parse(localStorage.getItem("history"));
        this.history = resHistory;
        console.log(this.history);
    }
}
```

Styles/app.css

```
/* Ember supports plain CSS out of the box. More info:
https://cli.emberjs.com/release/advanced-use/stylesheets/ */

body {
    font-family: Poppins;
    margin: 1.5rem 3rem;
}

.overlay {
    width: 100%;
    height: 100%;
    background-color: #2a2a2a;
    opacity: 0.7;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 1;
}

.head-text {
    font-size: 30px;
    font-weight: 500;
}

.head-desc-text, .description {
    font-size: 16px;
    color: #676767;
}

.container {
    display: flex;
    gap: 2rem;
    margin: 2rem 0;


}
.left {
    display: flex;
    flex-direction: column;
    width: 30%;
}
```

```
.my-wallet {
   background:linear-gradient(90deg,rgb(42 123 155 / 100%) 0%, rgb(87 199 133 / 100%)
50%, rgb(237 221 83 / 100%) 100%);
   color: #fff;
   border-radius: 10px;
   padding: 2rem;
   height: 250px;
}

.amount-title {
   color: #2b3e47;
}

.amount {
   font-size: 48px;
}

.wallet-btn-container {
   display: flex;
   gap: 1rem;
   margin: 0.5rem 0;
}

.wallet-btn {
   border: none;
   padding: 1rem 2rem;
   border-radius: 10px;
```

```
    background-color: #fcfcfc;
    color: #353535;
    cursor: pointer;
    font-weight: 800;
}

.wallet-btn i {
    font-weight: 800;
}

.add-modal {
    background-color: #fff;
    color: black;
    border-radius: 10px;
    padding: 2rem 4rem;
    z-index: 5;
    position: absolute;
    top: 30%;
    left: 40%;
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

input {
    padding: 0.5rem 1rem;
    border-radius: 5px;
    border: 1px solid #ccc;
```

```css
    width: auto;

}

.close-icon {
    position: absolute;
    top: 1rem;
    right: 1rem;
    cursor: pointer;
}

.transaction-history {
    margin-top: 1rem;
    padding: 2rem;
    background-color: #fff;
    box-shadow: 1px 2px 4px rgb(0 0 0 / 20%);
    border: 1px solid #ddd;
}

.transaction {
    border-bottom: 1px solid #ccc;
    margin: 1.5rem auto;
    display: flex;
    justify-content: space-between;
    align-items: center;
    gap: 3rem;
}
```

```
.transaction span {
   width: 100px;
}

.hist-name {
   font-size: 16px;
}

.hist-desc {
   font-size: 14px;
   color: #575757;
}

.subscription-tracker {
   background-color: #fff;
   box-shadow: 1px 2px 4px rgb(0 0 0 / 20%);
   border: 1px solid #ddd;
   padding: 2rem;
   display: flex;
   flex-direction: column;
   width: 60%;
}

.add-btn, .add-sub-btn {
   border: none;
   padding: 0.6rem 1rem;
   background-color: #2a2a2a;
   color: white;
```

```css
    border-radius: 5px;
    cursor: pointer;
}

.add {
    margin: 1rem 0;
}

.subscription-tracker-header {
    display: flex;
    align-items: start;
    justify-content: space-between;
    gap: 2rem;
    margin-bottom: 1.5rem;
}

.subscription-tracker-head {
    display: flex;
    flex-direction: column;
}

.heading {
    font-size: 22px;
    font-weight: 600;
}

.subscriptions {
    display: flex;
```

```
  gap: 1.5rem;
  flex-wrap: wrap;

}

.subscription-card {
  margin: 1rem 0;
  padding: 1.5rem;
  width: 350px;
  border-radius: 20px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: start;
  gap: 0.5rem;
  box-shadow: 1px 2px 4px rgba(0,0,0,0.3);
  position: relative;
  border: 1px solid #ddd;
}

.subscription-card div {
  margin: 1.5rem 0;
}

.edit-icon {
 position: absolute;
 top: 1rem;
 right: 3rem;
```

```
  cursor: pointer;
}

.del {
   right: 1rem;
   color: red;
   font-weight: 600;
}

.plus {
   color: green;
   font-weight: 600;
}

.cancel {
   right: 5rem;
   border: 1px solid #e64646;
   border-radius: 10px;
   padding: 0.3rem 0.8rem;
   background: transparent;
   color: #e64646;
   font-size: 12px;
}

/* .edit-del-modal {
   position: absolute;
   top: 1rem;
   right: 1rem;
```

```css
    background-color: white;
    border: 1px solid #ccc;
    padding: 0.3rem 2rem;
    font-size: 12px;
    cursor: pointer;
} */

.add-sub-modal {
    background-color: #fff;
    color: black;
    border-radius: 10px;
    padding: 2rem;
    z-index: 5;
    position: fixed;
    top: 8%;
    display: flex;
    flex-direction: column;
    gap: 1rem;
    width: 400px;
}

.name {
    font-size: 24px;
    font-weight: 500;
}

.sub-plan {
    font-size: 14px;
    color: #666666;
}

.sub-amount {
    font-size: 30px;
}

.category, .payment-method {
    border-radius: 20px;
    padding: 0.3rem 0.7rem;
    font-size: 12px;
}

.category {
    background-color: #ccc;
    margin-right: 0.5rem;
}

.payment-method {
    background-color: #4cad6e;
    color: white;
}

.empty-subscription {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    width: 100%;

    height: 50vh;

}
```