Improving the Diamond Card Sorting Web Application

Philipp Brandl
Tamara David
Bernhard Kargl

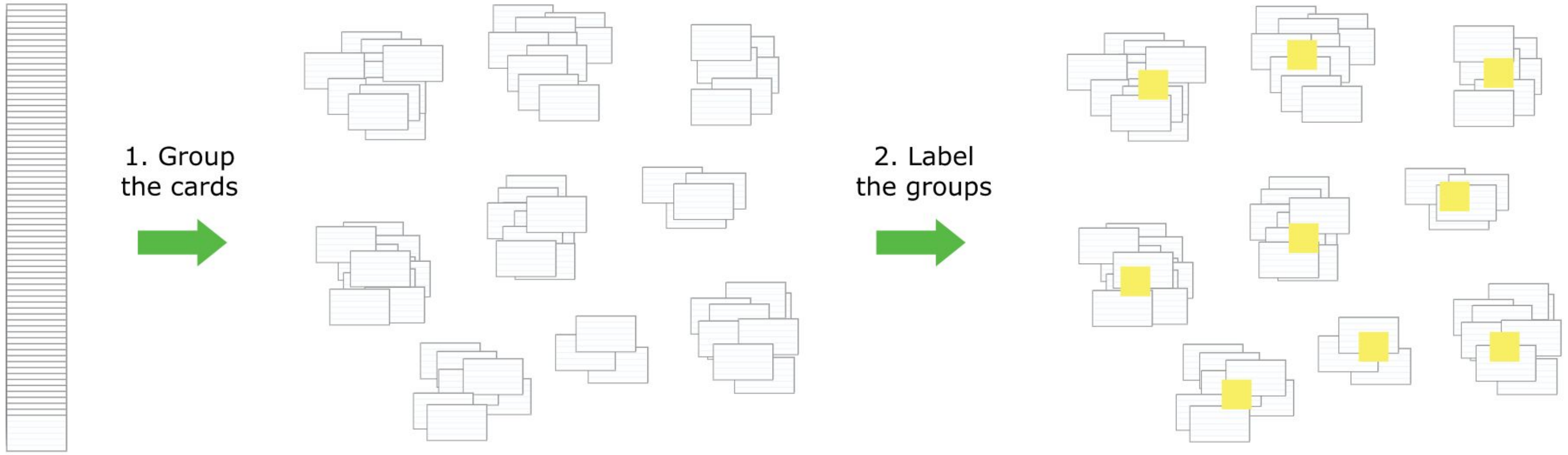
Information Architecture and Web Usability WS 2021 Group 1 - 26.01.2022

What is CardSorting?

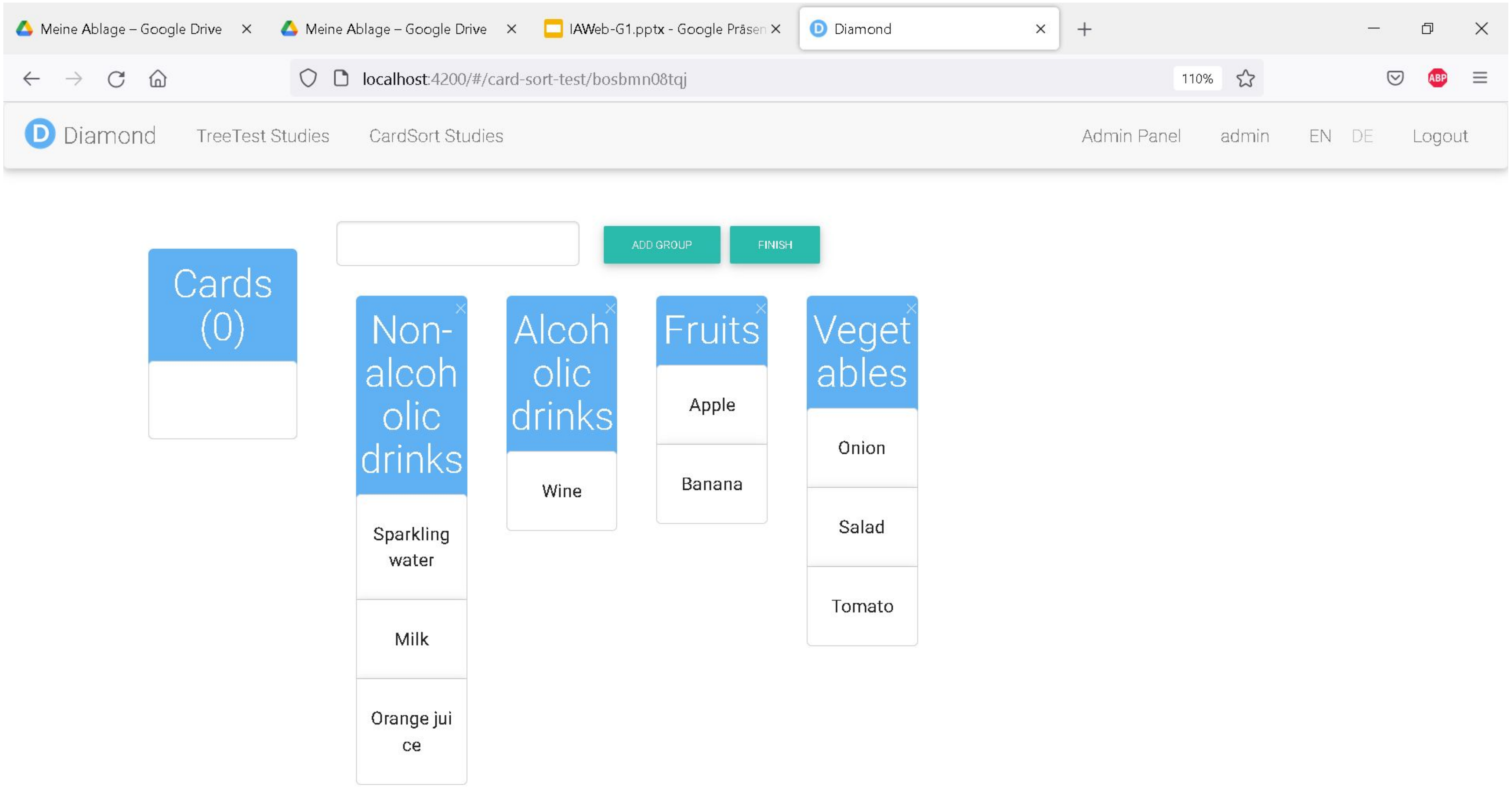
- UX research technique.
- Users organize topics into groups.
- They are asked for their train of thought after/while labelling.
- Covers the target audience domain knowledge.

Source: https://www.nngroup.com/articles/card-sorting-definition/

What is CardSorting?



Diamond CardSorting v 1.0


Meine Ablage – Google Drive
Meine Ablage – Google Drive
IAWeb-G1.pptx - Google Präsen
Diamond
localhost:4200/#/card-sort-test/bosbmn08tqj
110%
Diamond
TreeTest Studies
CardSort Studies
Admin Panel
admin
EN
DE
Logout
ADD GROUP
FINISH

Cards (0)

Non-alcoholic drinks	Alcoholic drinks	Fruits	Vegetables
Sparkling water	Wine	Apple	Onion
Milk		Banana	Salad
Orange juice			Tomato

Basic Project Tasks

Diamond

- Updating of integrated tools
- Improvement of card sorting UI
- Addition of accessible Drag’n’Drop

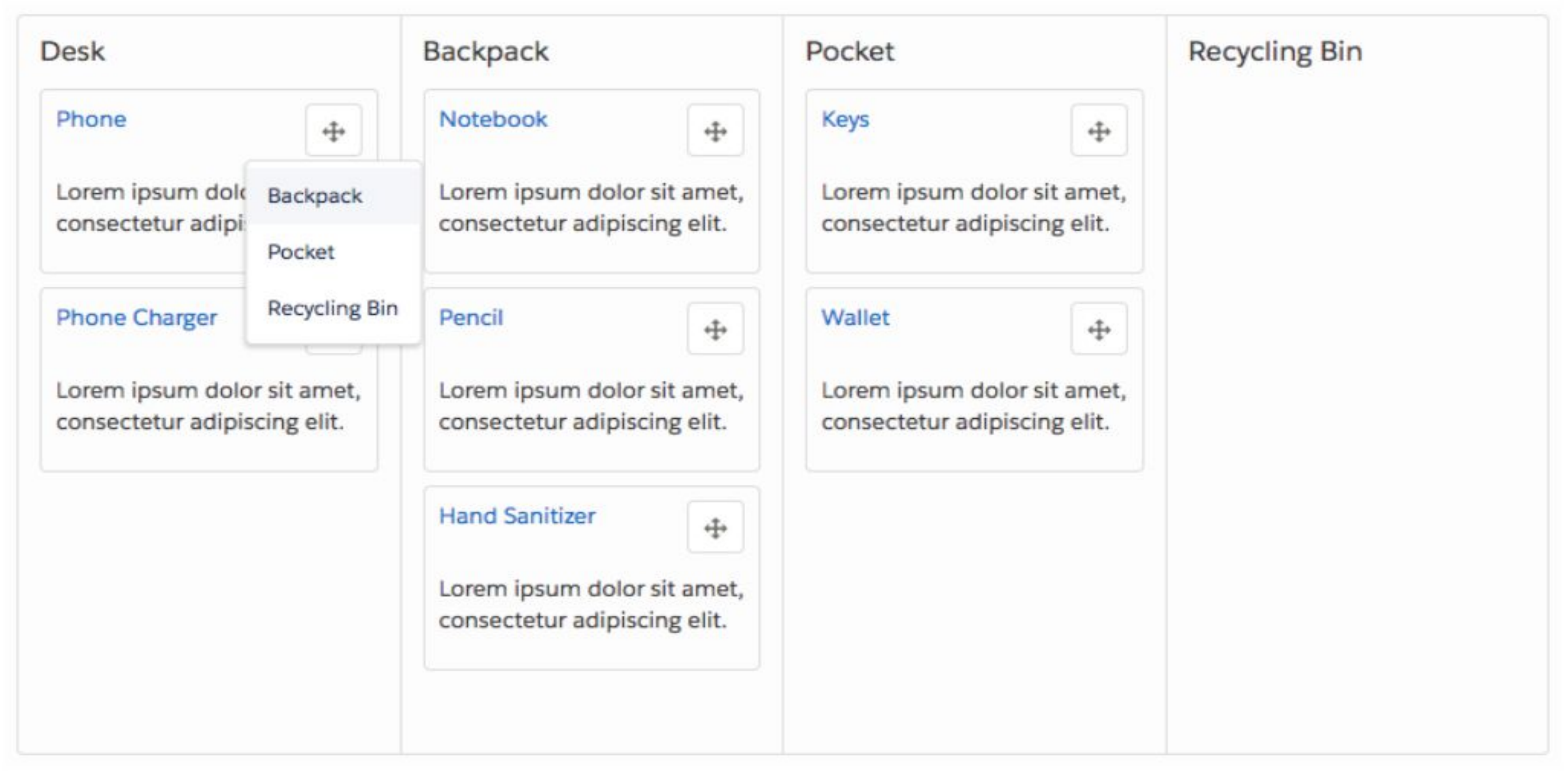
Initial Steps

- Setup project:
  - Creating Repo
  - Installing MongoDB
  - Installing NodeJS
- Angular 11 → updated to Angular 12 LTS.
- Typescript 4.0.5 → updated to Typescript 4.3.5.
- CDK version 11.0.3 → updated to CDK version 11.2.13
- Get to know the programming languages & framework.
- Get to know the initial implementation.
- Gather further improvement ideas.

Accessible Drag and Drop (basic idea)

- A button for the "drag" functionality (onclick-event), which opens a dropdown menu for group selection.
- Move between groups with tab key.
- Last element of the dropdown opens a dialog to add a new group with a given name.
- After creating a new group it can be renamed and deleted if necessary.

Accessible Drag and Drop - salesforce pattern


Desk

Phone

Lorem ipsum dol
consectetur adipi

Phone Charger

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Backpack

Notebook

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Pencil

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Hand Sanitizer

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Pocket

Keys

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Wallet

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Recycling Bin

Image Source: https://medium.com/salesforce-ux/4-major-patterns-for-accessible-drag-and-drop-1d43f64ebf09

Drag'n'Drop Salesforce Pattern - Demo

https://youtu.be/ABI8PxYh0nE

Accessible Dialog (Modal)

- Functionality:
  - Centralized interface.
  - Fully keyboard controlled.
- Tasks in Diamond:
  - Add groups.
  - Create cards.

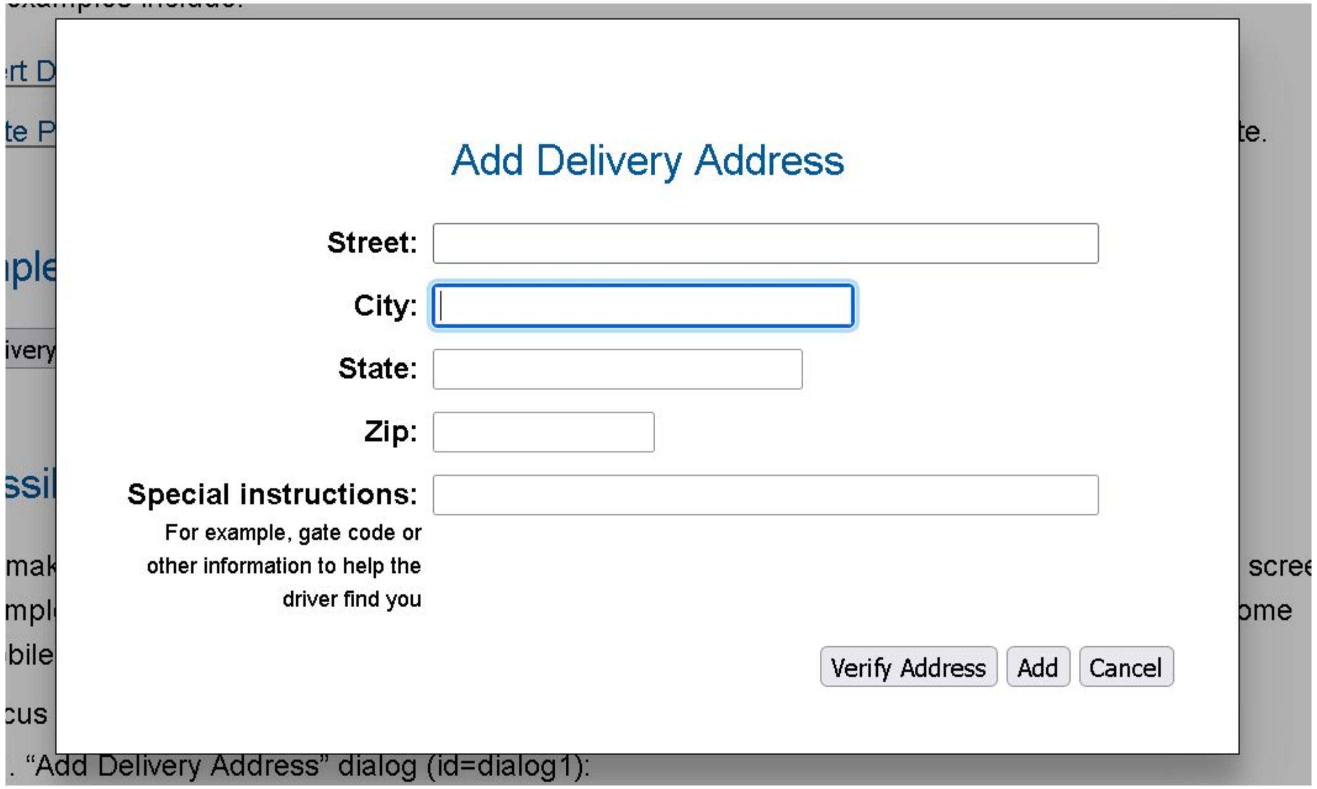
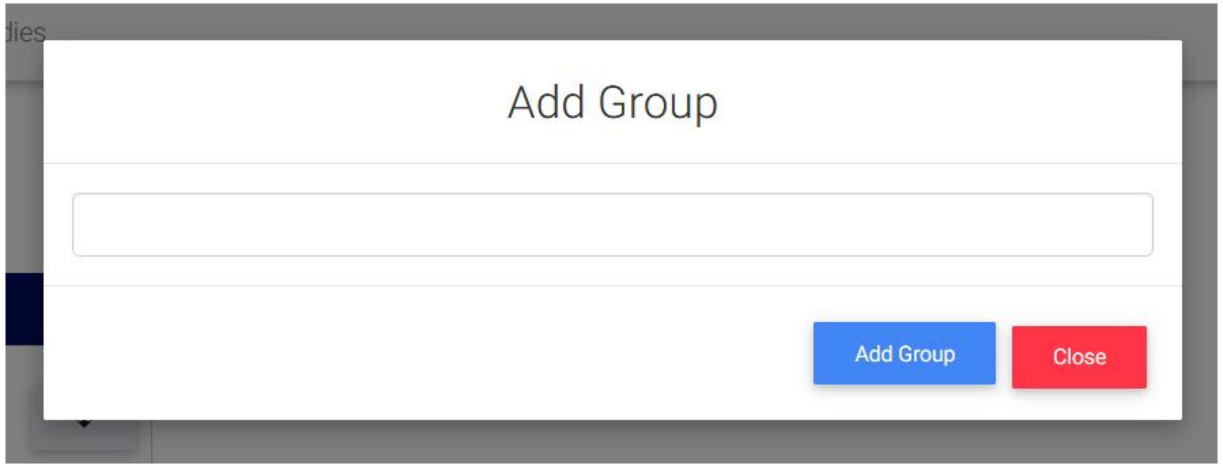

Add Delivery Address
Street:
City:
State:
Zip:
Special instructions:
For example, gate code or other information to help the driver find you
Verify Address
Add
Cancel
. "Add Delivery Address" dialog (id=dialog1):

Image Source: https://www.w3.org/TR/wai-aria-practices-1.1/examples/dialog-modal/dialog.html

Add Group - Modal HTML

```
<div id="create_group" class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header text-center">
        <h3 class="modal-title">Add Group</h3>
      </div>
      <div class="modal-body">
        <input class="form-control" type="text" [(ngModel)]="groupName"
                 type="text" (keyup.enter)="addGroup()">
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-primary btn-control" (click)="addGroup()"
                 data-dismiss="modal">{{ "Add Group" | translate }}</button>
        <button type="button" class="btn btn-danger btn-control" data-dismiss="modal" aria-label="Close">
          {{"Close" | translate}}
        </button>
      </div>
    </div>
  </div>
</div>
```



The highlighted parts add a new group to the study. Either on click or on button press.

Image Source: Diamond Implementation

Card Group HTML

```
<div class="card-group" id="{{ group_i }}">
   <button (click)="deleteGroup(group_i)" (keypress)="deleteGroup(group_i)"
               aria-label="close" class="delete glyphicon glyphicon-remove">
   </button>
   <h2 contenteditable="true" id="group_name_{{group_i}}"
         (blur)="changeGroupName($event.target, group_i)">
      {{ res.group_name }}
   </h2>
   <div cdkDropList class="group" [cdkDropListData]="res.group_list"
         (cdkDropListDropped)="drop($event)">
      <div class="card" id="{{i}}" *ngFor="let card of res.group_list; index as i" cdkDrag>
         <h5 tabindex="0">{{card}}</h5>
         <div class="dropdown">
            ....
         </div>
      </div>
   </div>
</div>
```

The red part adds an HTML object for every saved card of the group.

```
<button class="btn btn-move dropdown-toggle" type="button"
            id="dropdownMenuButton_{{i}}" data-toggle="dropdown"
            aria-haspopup="true" aria-label="move card" aria-expanded="false">
   <i class="glyphicon glyphicon-move"></i>
</button>
<ul class="dropdown-menu" id="dropdow-content">
   <span tabindex="0" role="link" id="{{resgroup_i}}"
            *ngFor="let group of resultGroups; index as resgroup_i"
            (click)="simulateDrop($event.target, card, group_i)"
            (keypress)="simulateDrop($event.target, card, group_i)">
      {{group.group_name}}
   </span>
   <span tabindex="0" role="link" class="add-group-link"
            (click)="addGroup()"  (keypress)="addGroup()">
      {{ "Add Group" | translate }}
   </span>
```

Here, a list of all groups is shown to the user, from which he can choose (simulate) in which group to drop the chosen card.

Drop() vs simulateDrop()

- Drop - function
  - needs mouse event
  - places elements into container
  - create new container if not present

- simulate drop - function
  - simulates drop function
  - move element from one group to another

Drop( ) vs. simulateDrop( ) - TypeScript

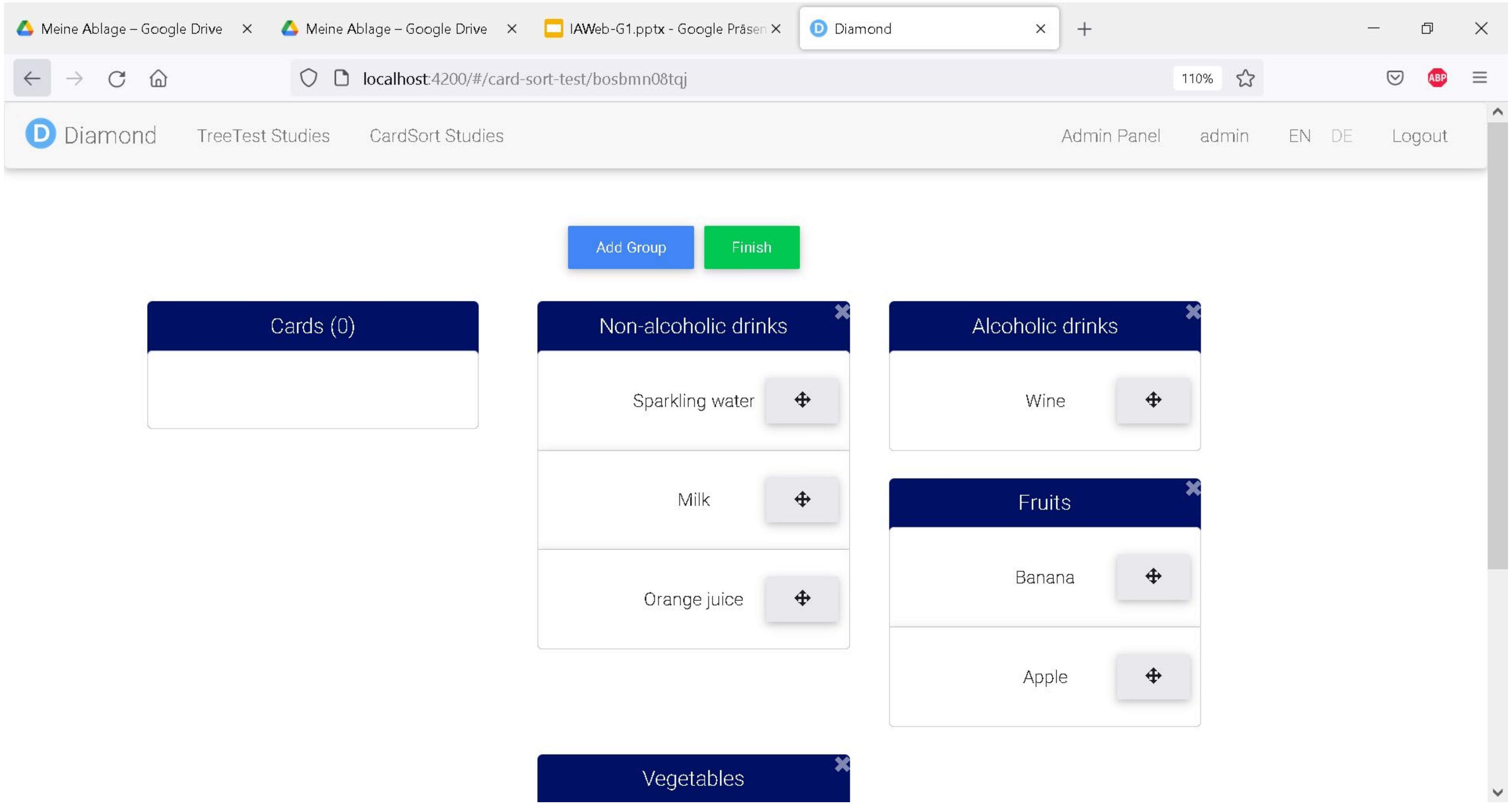
```
drop(event: CdkDragDrop<string[]>) {
   if (!event.isPointerOverContainer) {
      this.addGroup();
      transferArrayItem(event.previousContainer.data,
         this.resultGroups[this.resultGroups.length - 1].group_list,
         event.previousIndex,
         0);
   }
   else if (event.previousContainer === event.container) {
      moveItemInArray(event.container.data, event.previousIndex,
                              event.currentIndex);
      console.log(event.currentIndex);
   } else {
      transferArrayItem(event.previousContainer.data, event.container.data,
                              event.previousIndex, event.currentIndex);
   }
}
```

```
simulateDrop(target, card: any, oldgroupID){
   const cardID = this.resultGroups[oldgroupID].group_list.find(x => x === card);
   this.resultGroups[oldgroupID].group_list.splice(cardID, 1);
   this.resultGroups[target.id].group_list.push(card);
}
```

Code on the left checks, if the mouse is above the container for all card groups, when manual drag and dropping to create a new group.

On the right is the functionality to simulate a drop into a chosen group.

Diamond CardSorting v 2.0



Live Demo - Card Sorting in Diamond

https://www.youtube.com/watch?v=EcD1Ym4_hdU

Achievements

- Update basic functionality.
- Several bugfixes.
- UI Improvements:
  - Larger, accessible fonts.
  - Resizing of cards.
  - Accessibility through keyboard.
  - Creation dialog for new groups.
  - Dropdown for card movement.

Sources

- https://medium.com/salesforce-ux/4-major-patterns-for-accessible-drag-and-drop-1d43f64ebf09
- https://medium.com/salesforce-ux/4-major-patterns-for-accessible-drag-and-drop-1d43f64ebf09
- https://www.w3.org/TR/wai-aria-practices-1.1/examples/dialog-modal/dialog.html