The Project Management and Sorting Interfaces for Sortit

Matthias Stefan

The Project Management and Sorting Interfaces for Sortit

Matthias Stefan B.Eng.

Master's Project

Master's Degree Programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dr. Keith Andrews
Institute of Interactive Systems and Data Science (ISDS)

Graz, 2018-01-27

Abstract

Sortit is an online card sorting tool that allows a project manager to set up and manage a card sorting study. Participants receive a URL and can organize cards into groups using a web-based interface. Sortit uses the JavaScript full stack framework Meteor for its backend and the JavaScript library React for its frontend. Data is stored in the object-oriented database MongoDB. The initial implementation was developed by Haris Ljajić [2016].

Upon this work, a project management interface was created for the project manager. An overview page displays basic information about a project, including the cards and current sorts. Sorts can be imported and exported from a CSV file. The import process gives feedback about malformed files. The sorting interface supports drag and drop functionality for the cards and when finished the program prompts the participant to explain the thinking behind their sorting strategy (their mindset). The project uses the coding guideline BEM for its style sheets. Reactstrap is used to integrate Bootstrap 4 for various interface components.

Contents

List of Figures

List of Listings

Chapter 1

Introduction

Information Architecture (IA) is the "structural design of an information space to facilitate intuitive access to its contents", as described by Andrews [2018, page 25]. Within the field of IA, Card Sorting, as described by Bartel [2017], is a method used to construct an information hierarchy that is understandable to users. Terms are written onto cards and participants are asked to organize them into groups and then name the groups. Many tools are available to do the sort including locally installed software and online. [Bartel, 2017]

Tullis [2017] lists eight such tools. Five are web-based (OptimalSort, Proven by Users Online Card Sorting, UserZoom, UsabiliTest Card-Sorting, and Simple Card Sort), but none of them is available under a free open source license. Every web-based service is centralized and only usable on their server. There are three locally installable tools: uzCardSort is an open source plug-in for Firefox, xSort for Mac and UXSort for Windows are free but not open source.

Sortit is a new open source web-based card sorting tool, which can be installed on a web server or run locally. Sortit is built with the full-stack JavaScript framework Meteor [2017]. MongoDB [2017] is used as an object-orientated database. The frontend is built using React. [Stefanov, 2016] Keith Andrews and Haris Ljajic designed the system architecture for Sortit [Ljajić, 2016]. In version 1 of Sortit, the sorting interface worked in a simple way and the project management interface was a simple list of created projects.

React was developed by Facebook in order to solve problems with frontend JavaScript programming. The state of the application is linked to the DOM of the browser. If the state changes, React updates the DOM in an intelligent way, in order to handle changes together as far as possible. [Stefanov, 2016] React uses a script language called JSX to define components. JSX is a combination of JavaScript and XML / HTML. The programmer writes components in HTML and adds JavaScript functionality. More precisely, the HTML is embedded in a React JavaScript component, where functions, variables, states and prop types (that are forwarded from one component to an inner one) are used. The render function updates the DOM.

Chapter 2

CSS Styling and Reactstrap

Version 1 of Sortit used Bootstrap version 3 for styling user interface components such as the pop-up dialogs. The current version 2 of Sortit uses Bootstrap 4 [TBA, 2017], which supports the responsive design paradigm [Marcotte, 2014].

Sortit uses Reactstrap [Hernandez et al., 2017b] to integrate Bootstrap 4. All user interface components, such as Button, Form, Card, Warning Message and the card and sort lists, are made with Reactstrap. The overview page, sort list and card list are arranged with the Grid layout [Hernandez et al., 2017a]. Grid Layout allows to specify responsive placement of elements in a row. However full Flexbox layout [Coyier, 2017] is not yet implemented.

The style sheets have been restructured to use the Block Element Modular (BEM) naming convention [Strukchinsky and Starkov, 2017].

2.1 Reactstrap

Version 1 of Sortit used Bootstrap 3. Version 2 of Sortit now uses Bootstrap 4 alpha, which is included in Reactstrap. The library is not dependent on jQuery. The following features are used from Reactstrap:

- Alert: import feedback.
- Button (including Drop Down).
- Card: For the Project manager overview page and import interfaces.
- Collapse: Collapse parts of project management interface.
- Form: for HTML forms.
- Layout (grid for Flexbox): Layout of project managers tabs (lists).
- Navs and Navbar: top site menu.
- Tables: For the feedback table (upload preview of sorts and cards).
- Tabs: Navigation between sites of project management interface.

Hernandez et al. [2017b] describes the use of the components with examples. Listing 2.1 shows an example component in Sortit. The Grid is used to position the upload cards component. Reactstrap components use the react state. The Form's `onSubmit` event triggers a function `uploadSampleData` in the same file.

```
 1 uploadSampleData = (event) => {
 2         event.preventDefault();
 3         ...
 4 }
 5 ...
 6 toggle () {
 7     this.setState({ collapse: !this.state.collapse });
 8 }
 9 ...
10 render () {
11     this.reload = false;
12 
13     return (
14       <Container>
15         ...
16         <Row>
17           <Col sm={{ size: 2, push: 4, pull: 4, offset: 1 }}>
18             <Button onClick={this.toggle}>Add Cards</Button>
19           </Col>
20         </Row>
21         <Row>
22           <Col md={{ size: 6, push: 2, pull: 2, offset: 1 }}>
23             <Collapse isOpen={this.state.collapse} className="project__card-element
                   ">
24               <Card block>
25                 <CardTitle>Import Cards</CardTitle>
26                 <CardText>
27                   <Form onSubmit={this.uploadSampleData} method='POST' encType='
                       multipart/form-data'>
28                     <FormGroup>
29                       <Label for="exampleFile"></Label>
30                       <Input type="file" ref="upload" name="uploadCards" id="
                           uploadCards" />
31                       <FormText color="muted">
32                         Import a CSV File with Cards.
33                       </FormText>
34                     </FormGroup>
35                     <Button id="submit">Submit</Button>
36                   </Form>
37                 </CardText>
38               </Card>
39               ...
40             </Collapse>
41             ...
42           </Col>
43         </Row>
44         ...
45       </Container>
46     );
47 }
```

Listing 2.1: The Reactstrap implementation of Sortit's feature to upload new cards to an existing sort. The render function (line 10) renders the component. The component is defined in JSX in the return argument. The Container Row and Col are grid elements. In line 18, the `Add Cards` button expands the upload interface. The `onClick` and `onSubmit` actions call the corresponding functions in line 1 and 6.

2.2 Styling with CSS and BEM

Sortit uses SASS with the SCSS file format [Catlin et al., 2013]. SCSS extends CSS style sheets with extra features like variables, nested declarations, and functions. A compiler converts one or more SCSS files to one or more corresponding CSS files [Cederholm, 2013].

The file `global.scss` contains variables where colors, font styles, transmission paths, borders and shadow elements are defined. Additionally, there are basis classes which can not be assigned to a specific part of the page, like button or hidden. Other style information is described in the `sort.scss`, `menu.scss`, `csm.scss` and `_default.scss` files. The default file sets default directives for the used HTML tags to have the same base for every browser. The `csn.scss` file contains styling for the input group and password validator. The `menu.scss` file contains styling for the top navigation menu, `project.scss` file for the the project management interface, `sort.scss` for the sorting interface, and `admin.scss` for the administration interface.

BEM is a naming convention for modeling frontend user interface components as css classes [Strukchinsky and Starkov, 2017]. There are only classes which can be used by every HTML element. The styling is without reference to the selected element which brings simplicity and faster render times in the browser. Class names in BEM have the following structure:

```
block__element--modifier
```

for example:

```
project__row--deactivated
```

The `block` defines the larger block where this style description is used, typically larger areas like the menu, the footer, or a special page. For Sortit, the sorting interface (sort) and the raw data table (raw) are considered to be blocks.

An `element` modifier is optional. It is part of a block that has to be styled separately. This may be a button, a link, or a heading. Even if it is often used to style a certain HTML element it is not limited to it. A heading can be defined by any of `h1`, `h2`, `h3` ... If this changes, the style command remains valid.

The `modifier` styles a variant of the element, for example highlighted or disabled. The modifier name should reflect the underlying purpose rather than any specific presentational attribute. For example, `button--highlighted` rather than `button--red` since the style of highlighting may change.

To make the interface freely scalable, only relative sites in rem are used [Snook, 2011]. Examples of how BEM is realized in the sorting interface are shown in Listing 2.2 and Listing 2.3.

```
card-sort {
  margin-bottom: 0.5rem;
  padding: 0.5rem;
  ...
}

.card-sort--hoverd {
   margin-bottom: 2rem;
}
```

Listing 2.2: Part of the style sheet file `sort.scss` responsible for the sorting interface. If a card hovers over another element during dragging the second directive becomes active. The `margin-bottom` of the card enlarges and a gap appears where it is placed when dropped.

```
.project__row{
  padding: 0.5rem;
}

.project__row--deactivated
{
  font-style: italic;
  color: $color__text--background;
}
```

Listing 2.3: Part of the style sheet file `project.scss` responsible for the project management interface. The directives style the rows of the card and sort lists. Reactivated rows we give a less prominent text color `$color__text--background`.

Chapter 3

Sorting Interface

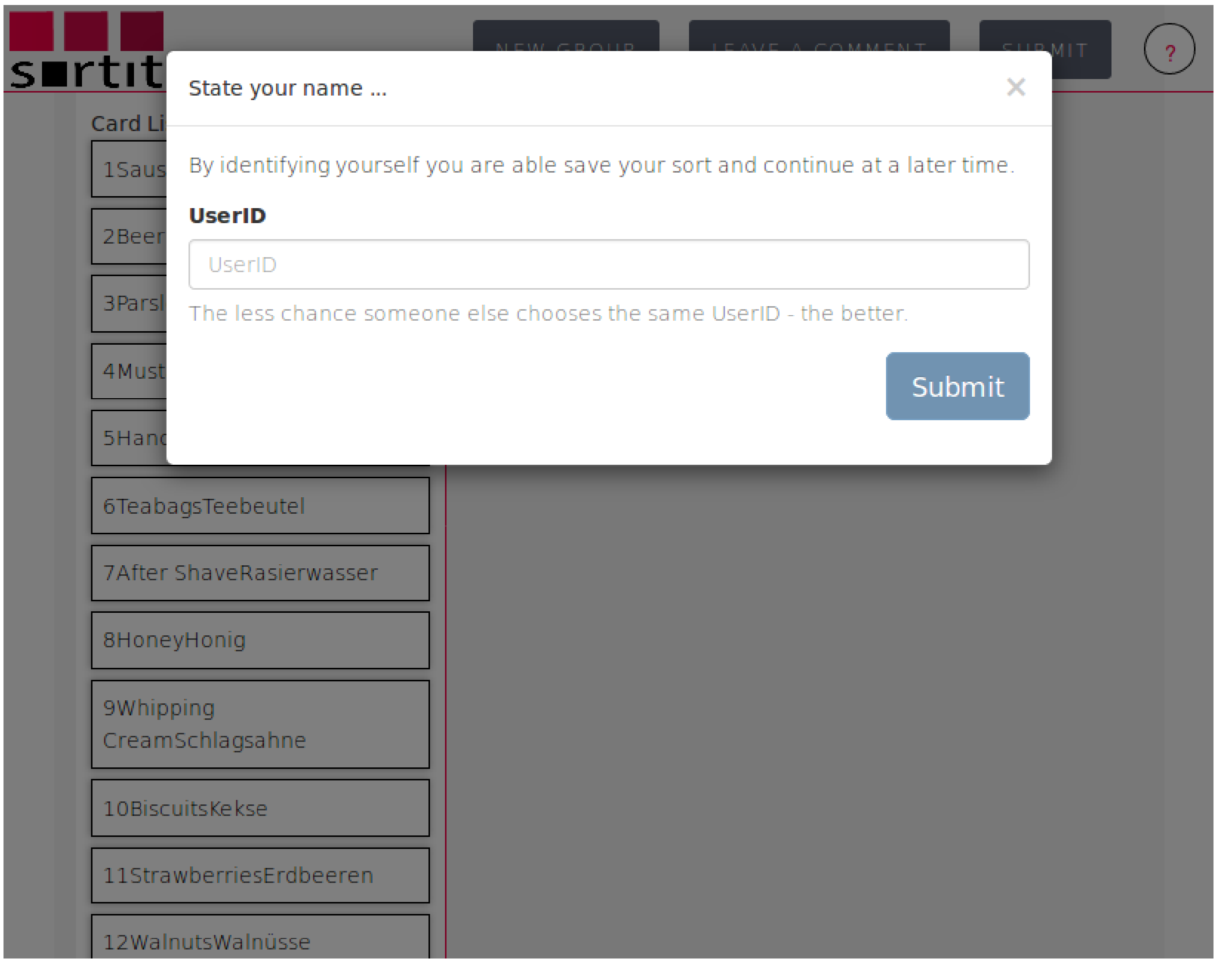
Sortit's sorting interface (the Sorting View) is designed to be used by the individual participants to perform a card sort. At the beginning, all the cards are listed on the left side and have to be sorted and placed into groups in the workspace on the right side.

3.1 Original Version (Sortit v1)

The original version 1 of Sortit, described by Ljajić [2016], used the following workflow to invite participants (sorters) to a card sorting study. The project manager sent the sort link to everybody who should participate on the sort. The sorter entered a user name in a pop-up window, shown in Figure 3.1. Individual sorters did not have to create an account or authenticate themselves, simple knowledge of the sort URL was sufficient. Afterwards, the sorter was redirected to the sorting interface to perform their sort. There were security concerns, because anyone could do the sort of anyone else, if the name was known. Furthermore,the same user perform multiple sorts by entering different user names.

Once identified, the sort view shown in Figure 3.2 was displayed. Cards could be moved with drag-and-drop. If a card was dropped onto empty space a new group was created automatically. If a card was dropped onto an existing group, the card was appended to that group. A group was also movable inside another group, creating a subgroup.

The elements of each card (ID, Title and Subtitle) were displayed in a single row with little or no separation. Subgroups were compressed into the same width as their parents, and thus became successively narrower.



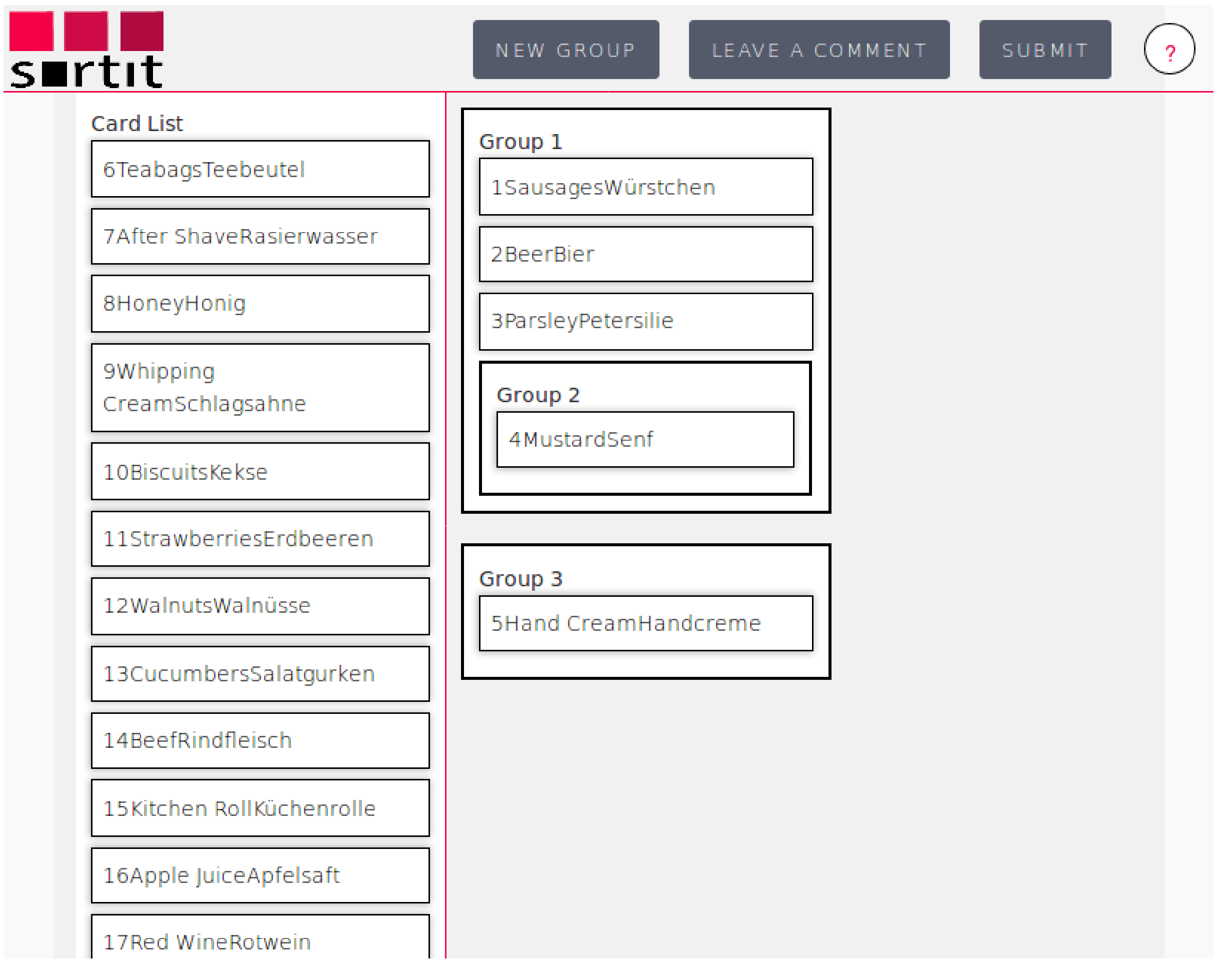
State your name ...

By identifying yourself you are able save your sort and continue at a later time.

UserID

The less chance someone else chooses the same UserID - the better.

Figure 3.1: In Sortit v1, the sorter entered a name or ID to start or resume a sort.



Card List

6TeabagsTeebeutel

7After ShaveRasierwasser

8HoneyHonig

9Whipping CreamSchlagsahne

10BiscuitsKekse

11StrawberriesErdbeeren

12WalnutsWalnüsse

13CucumbersSalatgurken

14BeefRindfleisch

15Kitchen RollKüchenrolle

16Apple JuiceApfelsaft

17Red WineRotwein

Group 1

1SausagesWürstchen

2BeerBier

3ParsleyPetersilie

Group 2

4MustardSenf

Group 3

5Hand CreamHandcreme

Figure 3.2: The sorting interface (Sorting View) from Sortit v1. Cards were placed into groups by drag-and-drop.

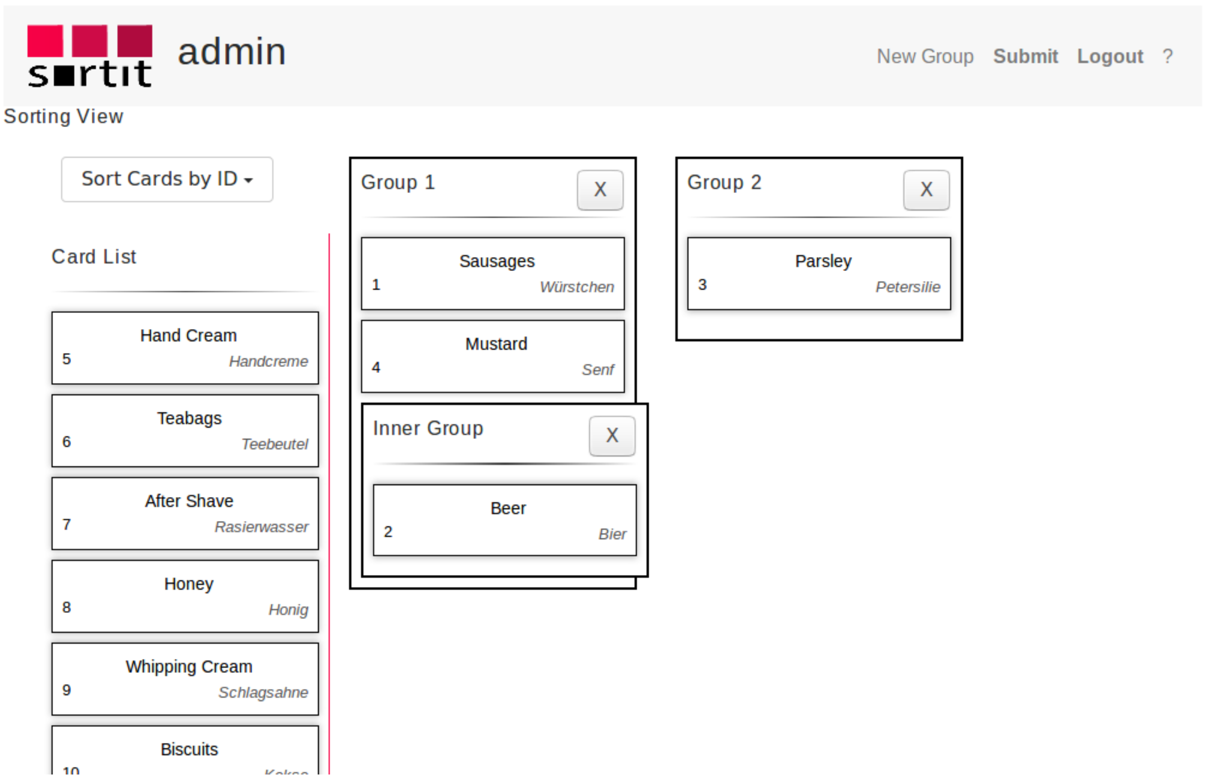


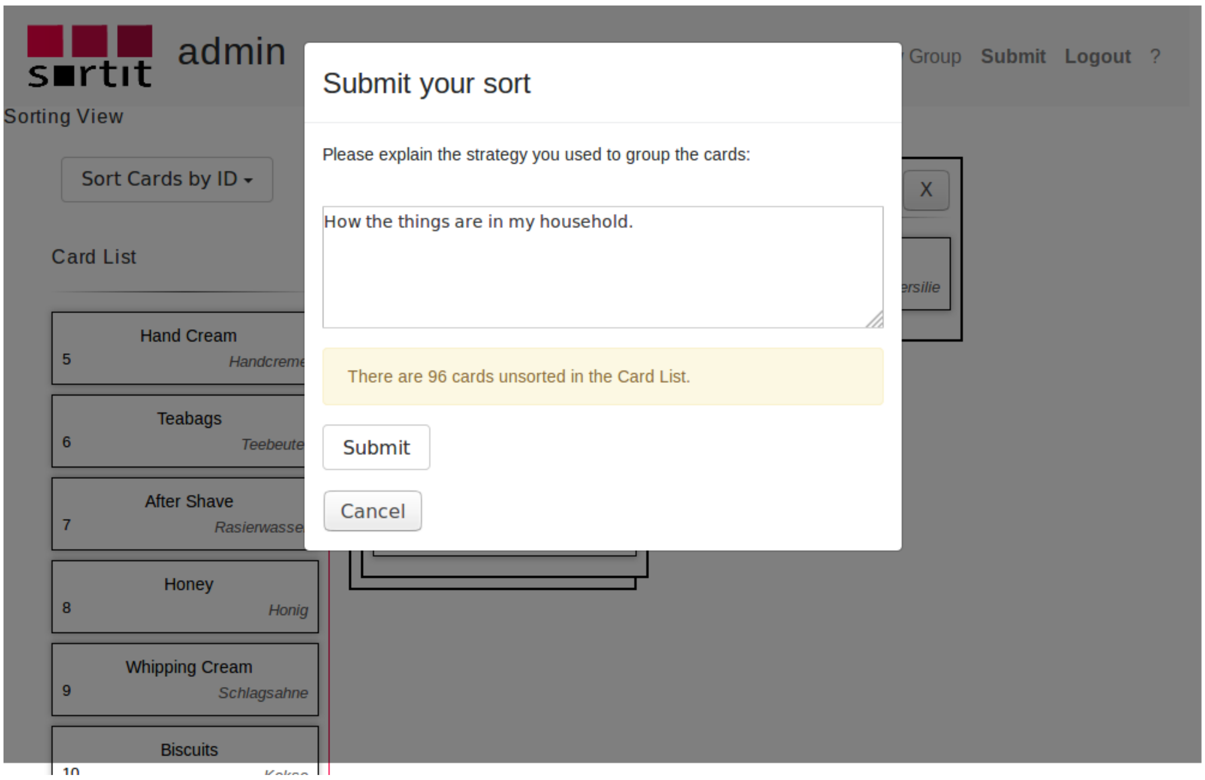
Figure 3.3: The new sort interface with one subgroup.

3.2 New Version (Sortit v2)

The start window where each sorter had to enter a name or ID has gone. Every participant now needs an account with a user name and password to log in to the system.

The new sort interface is displayed in Figure 3.3. The appearance of the cards, the groups, and the card list is defined in the `sort.scss` file. All cards have the same width. The card ID is displayed in the bottom left, the Title in the middle, and the Subtitle in the bottom right corner. The Group name is larger and there is now a delete button to delete a group. When a subgroup is deleted, any cards in the deleted group are moved into its parent. When a top-level group is deleted, any cards are put back into the card list. The cards are now sortable by ID, Title, and Subtitle. This affects the card list and all groups in the interface, but has no persistence in the database.

After the sorter has finished and submits their sort, they are asked to describe the strategy they used to group the cards (their mindset). To this end, the pop-up shown in Figure 3.4 is displayed to confirm submission of a sort. The sorter is informed about any unsorted cards remaining in the card list. After the sorter presses `Submit`, the sort is no longer editable, and the final confirmation pop-up shown in Figure 3.5 appears. Only the project manager can now reopen the sort.

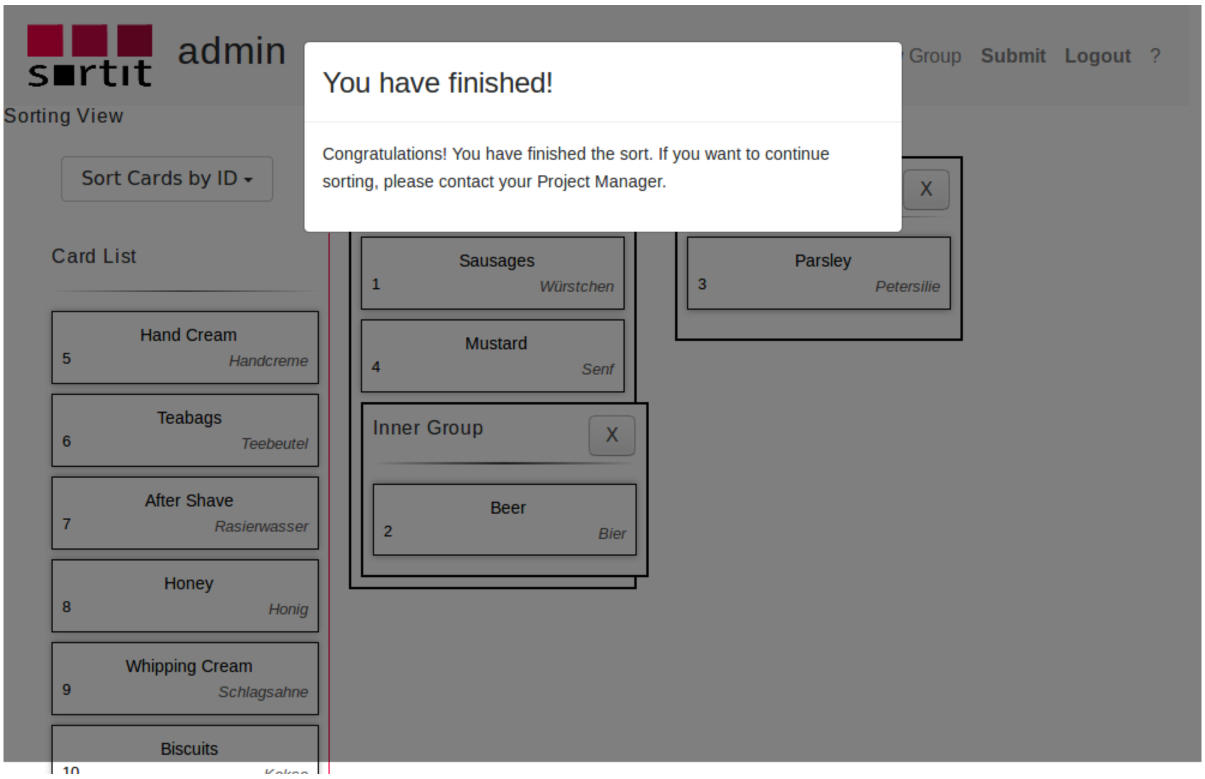

Submit your sort

Please explain the strategy you used to group the cards:

How the things are in my household.

There are 96 cards unsorted in the Card List.

Figure 3.4: The submission confirmation pop-up.


You have finished!

Congratulations! You have finished the sort. If you want to continue sorting, please contact your Project Manager.

Figure 3.5: The final confirmation pop-up.

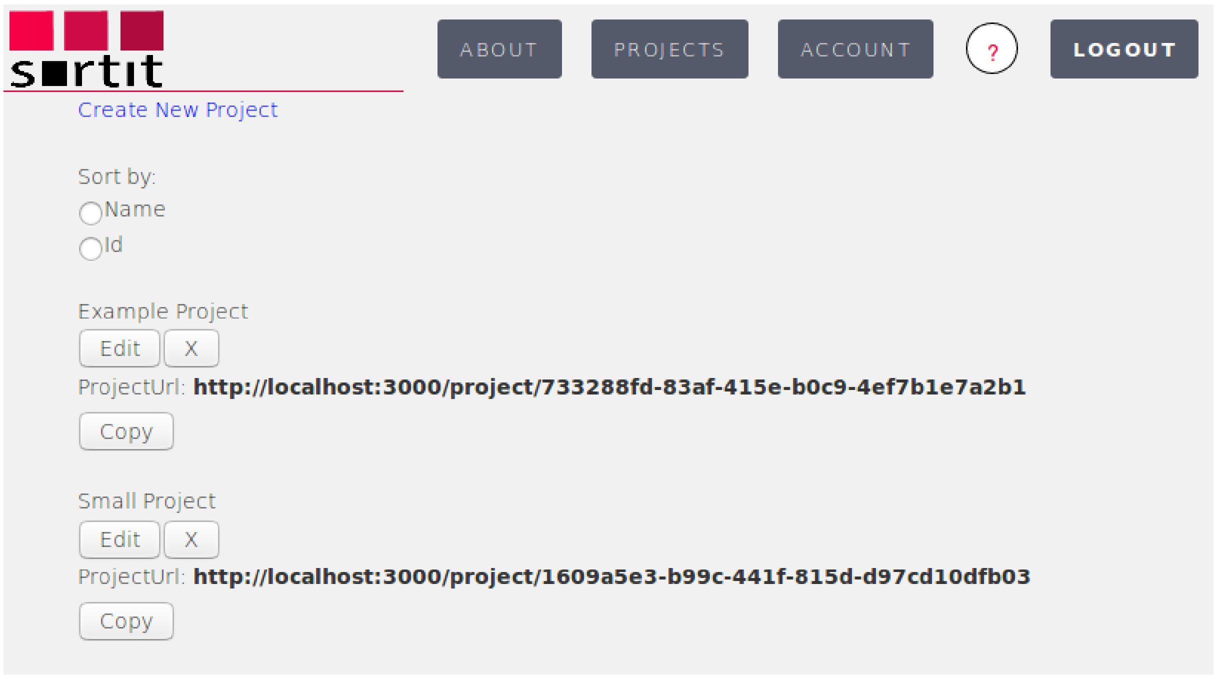
Chapter 4

Project Management Interface

Sortit's project management interface (the Project Management View) was extensively extended for Sortit v2. The project management interface is the tool for a project manager to create, adapt, monitor, and analyze card sorting projects and their results.

4.1 Original Version (Sortit v1)

The original project management tool in Sortit v1 was quite minimalistic, simply a list of projects, with their name and URL, as shown in Figure 4.1. Projects could be created, edited, or deleted. Existing projects could be listed in order of Name or ID. The project ID was a randomly generated hash string.



Example Project

ProjectUrl: **http://localhost:3000/project/733288fd-83af-415e-b0c9-4ef7b1e7a2b1**

Small Project

ProjectUrl: **http://localhost:3000/project/1609a5e3-b99c-441f-815d-d97cd10dfb03**

Figure 4.1: The original project management interface of Sortit v1. Projects could be listed in order of Name or ID. Projets could be created, edited, or deleted.

4.2 New Version (Sortit v2)

The new version of the project management interface for Sortit v2 has many improvements in both appearance and functionality. The list of projects can be ordered by name or by creation date, as shown in Figure 4.2. Each project element displays the name, creation date, and project link. Clicking the down-arrow button expands the description to show additional overview statistics for the project.

Clicking on the project name redirects the user to the Project Detail View, which consists of four tabs: Overview, Cards, Sorts, and Raw Data. The default tab is the Overview tab shown in Figure 4.3. It consists of the project metadata (also shown in the list of projects.). This metadata is divided into the Project Settings, the Current Status and the Assigned Mindsets (illustrated in a pie chart). Hovering over a segment of the Assigned Mindset pie chart displays the name of the corresponding mindset and the number of sorts associated with that mindset. This overview page was developed by Linda Kolb and Eva Ulbrich as part of the Information Architecture and Web Usability course in WS 2016/2017.

The Cards tab displays all the cards in the project as shown as individual in Figure 4.4. The ID, Title, and Subtitle are displayed and are editable. When the user clicks on an individual Title or Subtitle, the name becomes an input field and the corresponding text can be changed. Clicking on the × deletes a card. Changes are applied to existing as well as future sorts.

The Sorts tab displays the currently available sorts, as shown in Figure 4.5. The Sort ID, the User Name, and User Thinking are explained. User thinking is the comment that the user entered after completing the sort. The Mindset is assigned by the project manager and groups multiple user thinkings under one name. The number of top level groups is indicated. The completed column shows the percentage and absolute numbers of cards that have already been sorted. A sort can be closed for that user, so it can no longer be changed. A sort can be hidden, so that it is not visible in the Raw Data tab and is not exportable. This is useful if a sort has not been taken seriously or is an outlier for some reason. A sort can be deleted by clicking the button marked ×, and can be later restored by clicking the subsequent Undo Delete button.

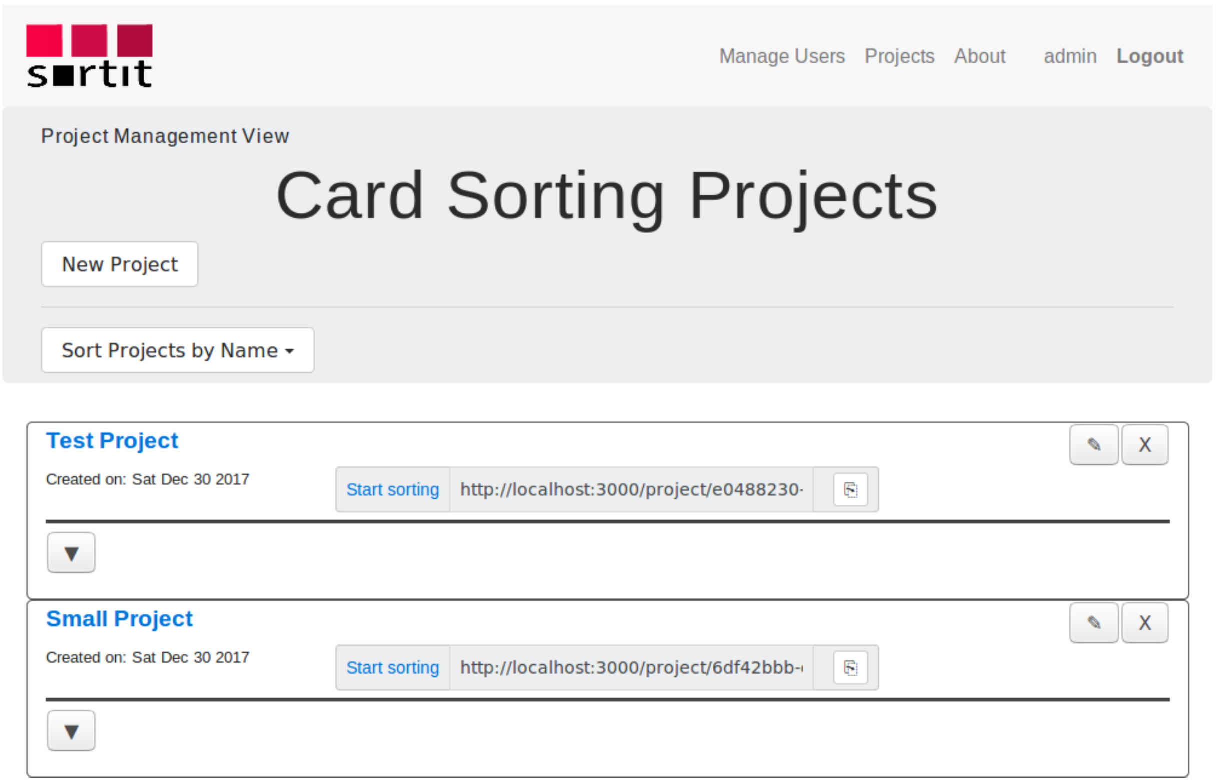
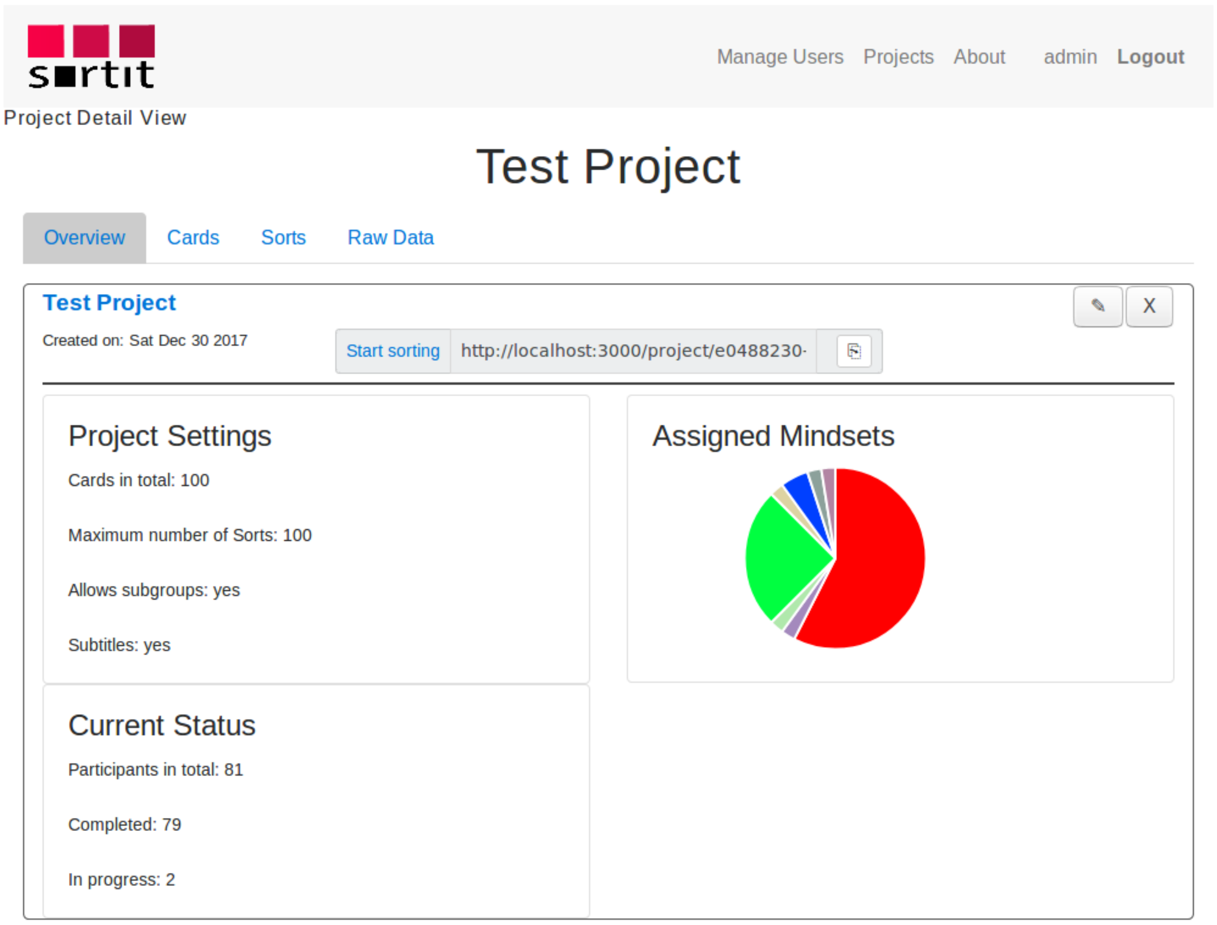


Figure 4.2: The new project management interface of Sortit v2. Projects can be ordered by Name or Creation Date. An expandable panel shows overview statistics for the project.

The Raw Data tab is displayed in Figure 4.6. It displays the raw data for the entire project with one sort in each row of the table. First, the Sort ID, User Name, User Thinking, and Assigned Mindset are displayed for each sort. Then, the group chosen for every card is listed. If there are subgroups, the most inner group name is used. On the right side, the mindsets which should be included in the table can be selected. The color assigned to a mindset can be changed by clicking the colored square. The Export button creates a CSV file containing sorts from the selected mindsets, which can be used for analysis in external tools.

The mindset selection and the color picker were implemented by Markus Lienbacher as part of the Information Architecture and Web Usability course in WS 2016/2017. The table was since restyled, the elements rearranged, and the export functionality implemented.



Test Project

Test Project

Created on: Sat Dec 30 2017

http://localhost:3000/project/e0488230-

Project Settings

Cards in total: 100

Maximum number of Sorts: 100

Allows subgroups: yes

Subtitles: yes

Current Status

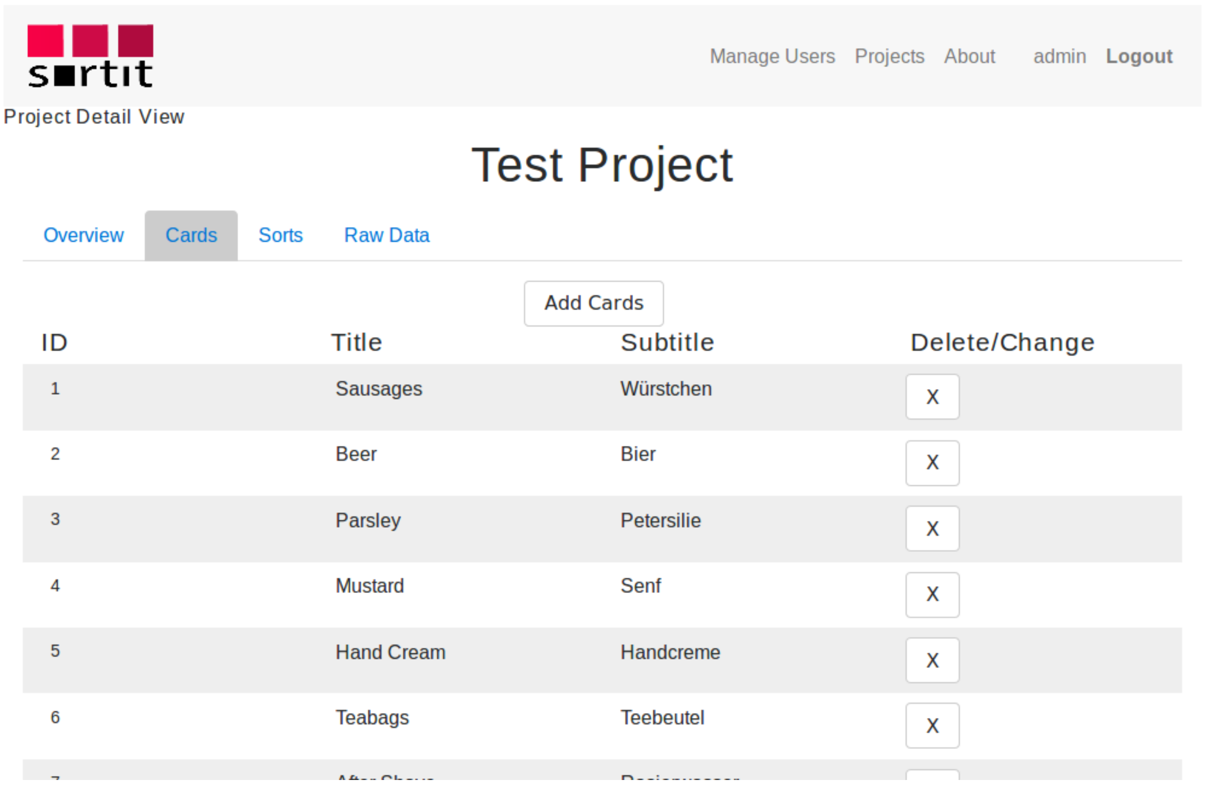
Participants in total: 81

Completed: 79

In progress: 2

Assigned Mindsets

Figure 4.3: The Overview tab of the Project Detail View.

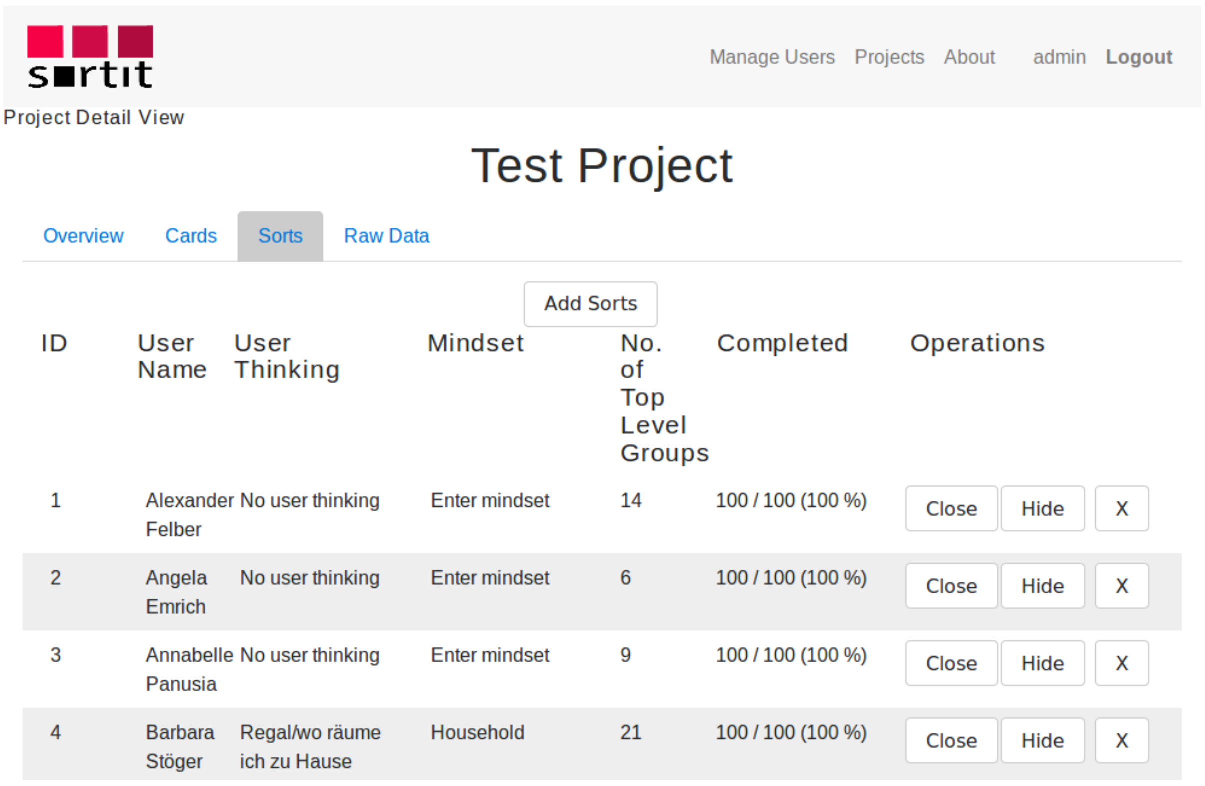


Project Detail View

Test Project

ID	Title	Subtitle	Delete/Change
1	Sausages	Würstchen	X
2	Beer	Bier	X
3	Parsley	Petersilie	X
4	Mustard	Senf	X
5	Hand Cream	Handcreme	X
6	Teabags	Teebeutel	X

Figure 4.4: The Cards tab of the Project Detail View.

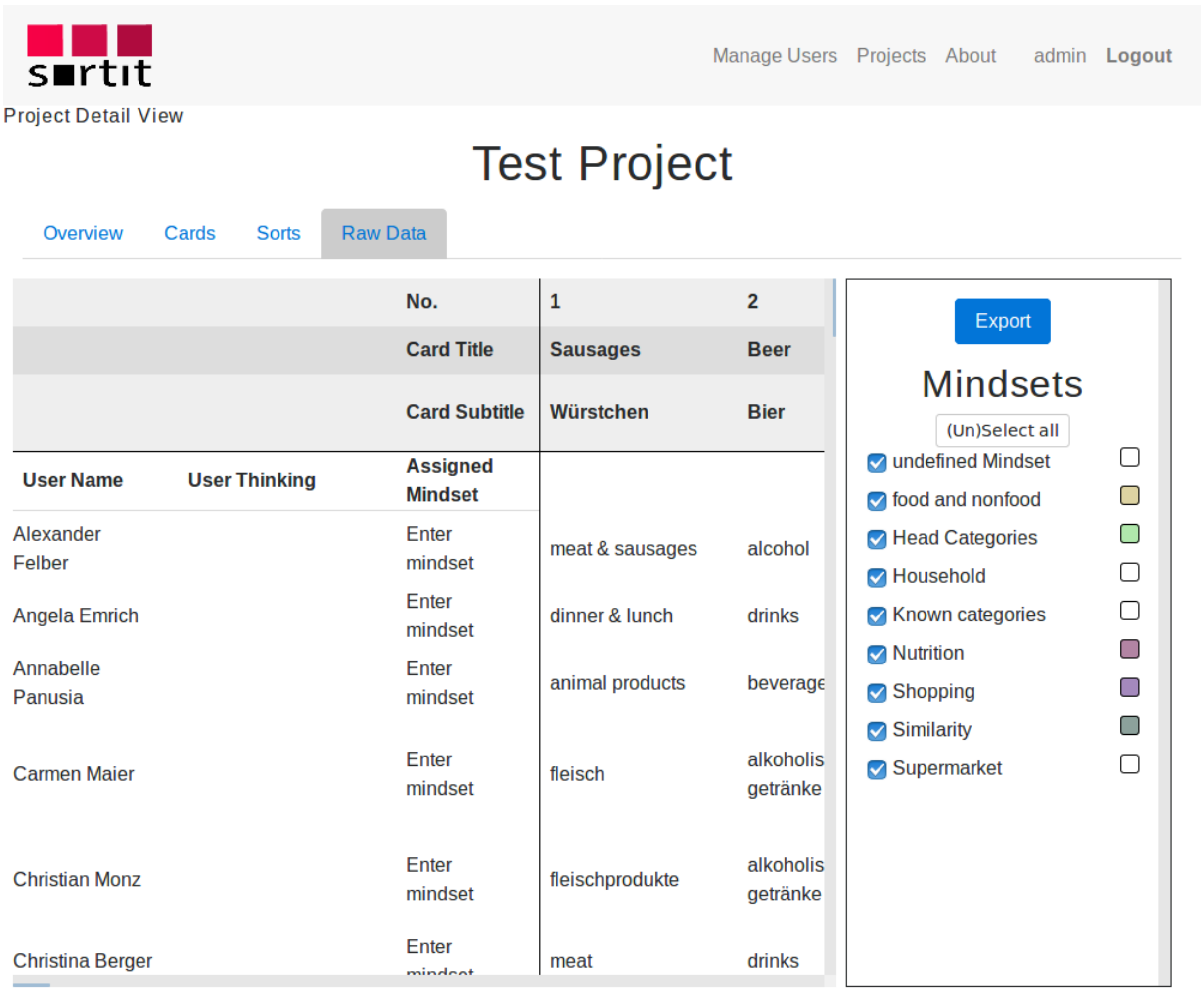


Project Detail View

Test Project

ID	User Name	User Thinking	Mindset	No. of Top Level Groups	Completed	Operations
1	Alexander Felber	No user thinking	Enter mindset	14	100 / 100 (100 %)	Close Hide X
2	Angela Emrich	No user thinking	Enter mindset	6	100 / 100 (100 %)	Close Hide X
3	Annabelle Panusia	No user thinking	Enter mindset	9	100 / 100 (100 %)	Close Hide X
4	Barbara Stöger	Regal/wo räume ich zu Hause	Household	21	100 / 100 (100 %)	Close Hide X

Figure 4.5: The Sorts tab of the Project Detail View.



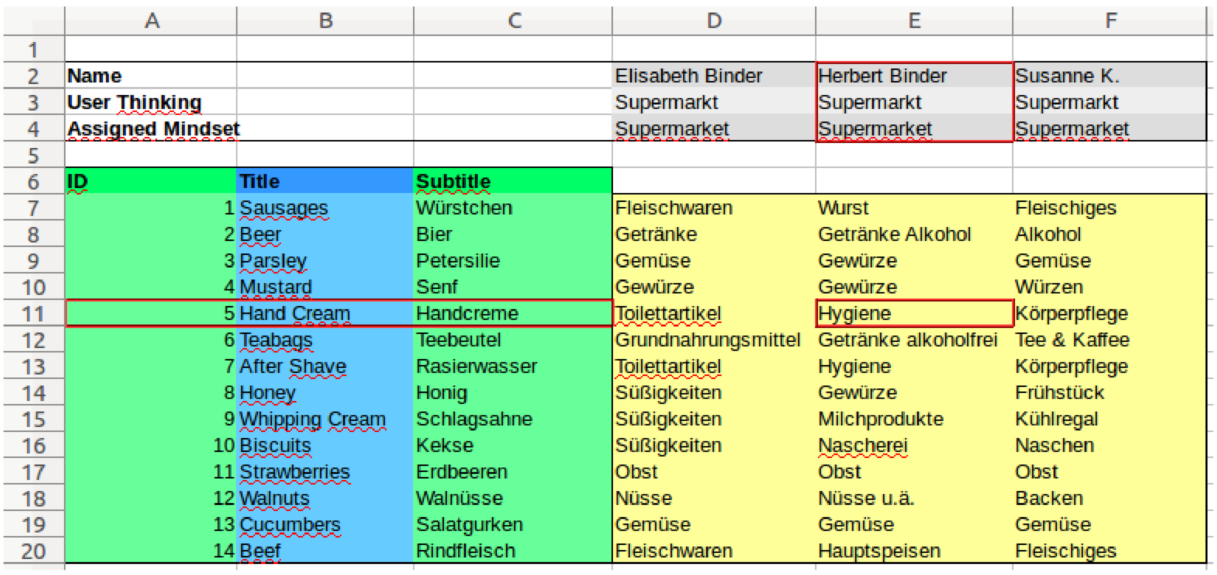
Test Project

		No.	1	2
		Card Title	Sausages	Beer
		Card Subtitle	Würstchen	Bier
User Name	User Thinking	Assigned Mindset		
Alexander Felber		Enter mindset	meat & sausages	alcohol
Angela Emrich		Enter mindset	dinner & lunch	drinks
Annabelle Panusia		Enter mindset	animal products	beverage
Carmen Maier		Enter mindset	fleisch	alkoholis getränke
Christian Monz		Enter mindset	fleischprodukte	alkoholis getränke
Christina Berger		Enter mindset	meat	drinks

Mindsets

✓ undefined Mindset	☐
✓ food and nonfood	🟨
✓ Head Categories	🟩
✓ Household	☐
✓ Known categories	☐
✓ Nutrition	🟪
✓ Shopping	🟪
✓ Similarity	⬛
✓ Supermarket	☐

Figure 4.6: The Raw Data tab of the Project Detail View.



	A	B	C	D	E	F
1						
2	**Name**			Elisabeth Binder	Herbert Binder	Susanne K.
3	**User Thinking**			Supermarkt	Supermarkt	Supermarkt
4	**Assigned Mindset**			Supermarket	Supermarket	Supermarket
5						
6	**ID**	**Title**	**Subtitle**			
7	1	Sausages	Würstchen	Fleischwaren	Wurst	Fleischiges
8	2	Beer	Bier	Getränke	Getränke Alkohol	Alkohol
9	3	Parsley	Petersilie	Gemüse	Gewürze	Gemüse
10	4	Mustard	Senf	Gewürze	Gewürze	Würzen
11	5	Hand Cream	Handcreme	Toilettartikel	Hygiene	Körperpflege
12	6	Teabags	Teebeutel	Grundnahrungsmittel	Getränke alkoholfrei	Tee & Kaffee
13	7	After Shave	Rasierwasser	Toilettartikel	Hygiene	Körperpflege
14	8	Honey	Honig	Süßigkeiten	Gewürze	Frühstück
15	9	Whipping Cream	Schlagsahne	Süßigkeiten	Milchprodukte	Kühlregal
16	10	Biscuits	Kekse	Süßigkeiten	Nascherei	Naschen
17	11	Strawberries	Erdbeeren	Obst	Obst	Obst
18	12	Walnuts	Walnüsse	Nüsse	Nüsse u.ä.	Backen
19	13	Cucumbers	Salatgurken	Gemüse	Gemüse	Gemüse
20	14	Beef	Rindfleisch	Fleischwaren	Hauptspeisen	Fleischiges

Figure 4.7: Template for importing sorts from a CSV file. The keywords that are bold have to be in the file. The first three keywords specify the cells for the sorter's name, the assigned mindset, and the user thinking. The keywords ID, Title and Subtitle mark the cells for the card data. Column D onwards contain the individual sorts. The yellow part is the place for the group names (result of each sort) assigned to each card. The corresponding CSV file is shown in Listing 4.1.

4.3 Import and Export

Sorts can be both imported and exported as CSV files. The data structure of the imported and exported CSV file is the same and is illustrated in Figure 4.7 and Listing 4.1. A message indicates a successful import. The page has to be reloaded in order to load the results in the project management interface.

The import box also indicates when the import is not successful. Error causes include not found keywords, no sorts or no cards, keywords in the wrong position, and uploaded cards not matching the cards stored in the project. The feedback includes a warning message describing the error and if necessary a snippet of the used CSV file, as shown in Figure 4.8.

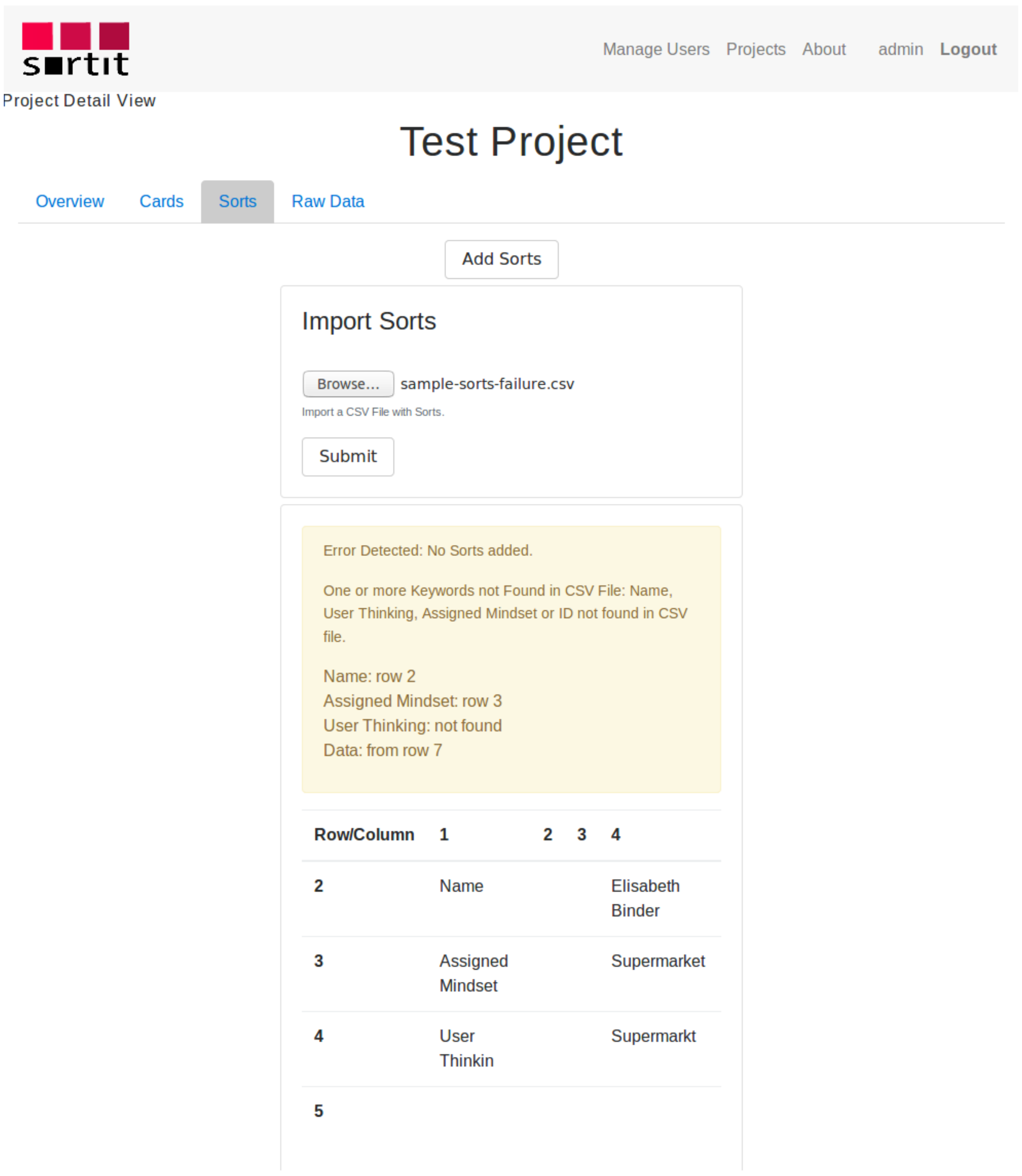
In the Cards tab, it is possible to upload cards from a CSV file. A header row containing the keywords ID, Title, and Subtitle is mandatory, as shown in Figure 4.9 and Listing 4.2. The upload fails if cards do not exactly match existing ones, or if a card ID is already in use with another name, as shown in Figure 4.10.

```
,,,,,
Name,,,Elisabeth Binder,Herbert Binder,Susanne K.
User Thinking,,,Supermarkt,Supermarkt,Supermarkt
Assigned Mindset,,,Supermarket,Supermarket,Supermarket
,,,,,
ID,Title,Subtitle,,,
1,Sausages,Würstchen,Fleischwaren,Wurst,Fleischiges
2,Beer,Bier,Getränke,Getränke Alkohol,Alkohol
3,Parsley,Petersilie,Gemüse,Gewürze,Gemüse
4,Mustard,Senf,Gewürze,Gewürze,Würzen
5,Hand Cream,Handcreme,Toilettartikel,Hygiene,Körperpflege
6,Teabags,Teebeutel,Grundnahrungsmittel,Getränke alkoholfrei,Tee & Kaffee
7,After Shave,Rasierwasser,Toilettartikel,Hygiene,Körperpflege
8,Honey,Honig,Süßigkeiten,Gewürze,Frühstück
9,Whipping Cream,Schlagsahne,Süßigkeiten,Milchprodukte,Kühlregal
10,Biscuits,Kekse,Süßigkeiten,Nascherei,Naschen
11,Strawberries,Erdbeeren,Obst,Obst,Obst
12,Walnuts,Walnüsse,Nüsse,Nüsse u.ä.,Backen
13,Cucumbers,Salatgurken,Gemüse,Gemüse,Gemüse
14,Beef,Rindfleisch,Fleischwaren,Hauptspeisen,Fleischiges
```

Listing 4.1: Example CSV file for importing sorts.

```
ID,Title,Subtitle
1,Sausages,Würstchen
2,Beer,Bier
3,Parsley,Petersilie
4,Mustard,Senf
5,Hand Cream,Handcreme
6,Teabags,Teebeutel
7,After Shave,Rasierwasser
8,Honey,Honig
9,Whipping Cream,Schlagsahne
10,Biscuits,Kekse
11,Strawberries,Erdbeeren
12,Walnuts,Walnüsse
13,Cucumbers,Salatgurken
14,Beef,Rindfleisch
```

Listing 4.2: Example CSV file for importing cards.



Test Project

Import Sorts

sample-sorts-failure.csv

Import a CSV File with Sorts.

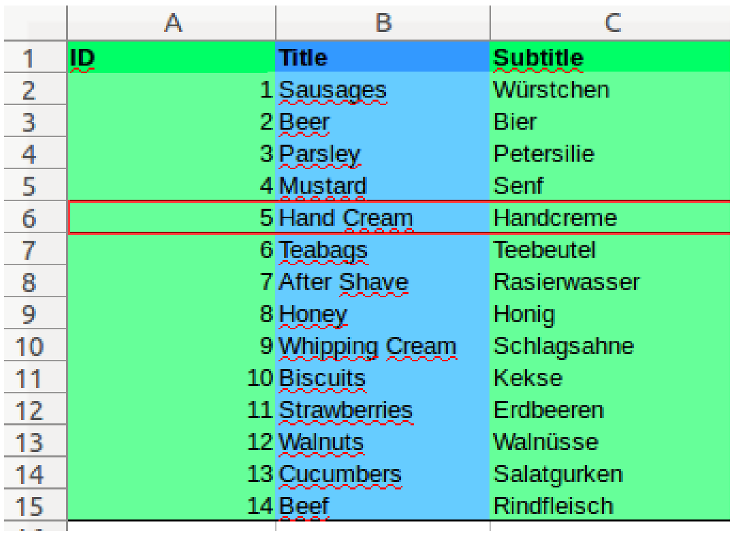
Error Detected: No Sorts added.

One or more Keywords not Found in CSV File: Name, User Thinking, Assigned Mindset or ID not found in CSV file.

Name: row 2
Assigned Mindset: row 3
User Thinking: not found
Data: from row 7

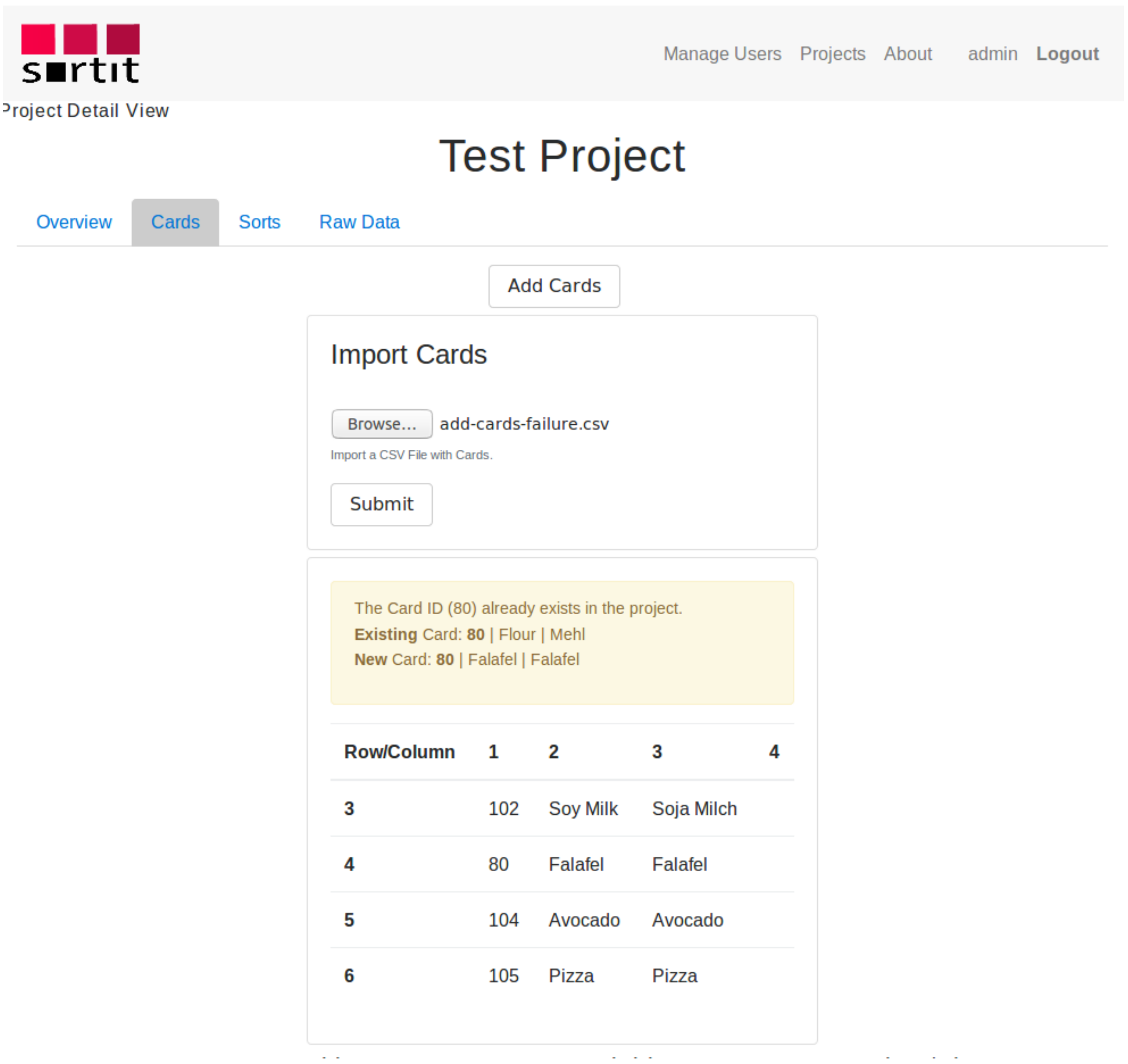
Row/Column	1	2	3	4
2	Name			Elisabeth Binder
3	Assigned Mindset			Supermarket
4	User Thinkin			Supermarkt
5				

Figure 4.8: Feedback of an error when importing sorts. The yellow warning message indicates that the row where the User Thinking is defined has not been found. A g is missing at the end of `User Thinkin`. The snippet of the CSV file shows where the error has occurred.



	A	B	C
1	**ID**	**Title**	**Subtitle**
2	1	Sausages	Würstchen
3	2	Beer	Bier
4	3	Parsley	Petersilie
5	4	Mustard	Senf
6	5	Hand Cream	Handcreme
7	6	Teabags	Teebeutel
8	7	After Shave	Rasierwasser
9	8	Honey	Honig
10	9	Whipping Cream	Schlagsahne
11	10	Biscuits	Kekse
12	11	Strawberries	Erdbeeren
13	12	Walnuts	Walnüsse
14	13	Cucumbers	Salatgurken
15	14	Beef	Rindfleisch

Figure 4.9: The Template for uploading Cards from a CSV file. The keywords ID, Title, and Subtitle in the header row are necessary. The corresponding CSV file is shown in Listing 4.2.



Test Project

Import Cards

add-cards-failure.csv

Import a CSV File with Cards.

The Card ID (80) already exists in the project.
Existing Card: **80** | Flour | Mehl
New Card: **80** | Falafel | Falafel

Row/Column	1	2	3	4
3	102	Soy Milk	Soja Milch	
4	80	Falafel	Falafel	
5	104	Avocado	Avocado	
6	105	Pizza	Pizza	

Figure 4.10: Feedback of an error when importing cards. The yellow warning message indicates that the ID 80 is already in use by another card. The corresponding snippet of the imported CSV file is shown.

Chapter 5

Future Work

The project management and sorting interfaces of Sortit still have scope for improvement and extension.

5.1 Sort ID

The Sort ID displayed in the `Sorts` tab (shown in in Figure 4.5) is not part of the database, but is dynamically assigned at runtime depending on the place in the array where the sorts have been added. If the user deletes a sort, the IDs will change after a refresh. They are not consistent. A consistent unique sort ID would have to be maintained in the database.

5.2 Ordering

The card and sort lists in the project management interface currently have fixed ordering. It would be desirable to allow a project manager to sort the table based on any column, for example by clicking a triangle in a column header.

5.3 Responsive Web Design

Responsive web design is a new paradigm for web design whereby web pages adapt to the size and other characteristics of the display device. Some of the pages in Sortit work well with narrower screens, others do not. The responsive design of some of the tabs in the project management interface could be improved.

The Boostrap 4 components in the `Overview` tab are inherently responsive. The tables of cards and sorts in the `Cards` and `Sorts` tabs respectively are usually too wide to be inherently responsive. Check boxes are provided to selectively choose columns for display. A possible future enhancement would be to implement responsive tables where the initial set of columns displayed depends upon the available screen width. The user interface of the `Raw Data` tab could also be further improved for narrower screens.

The sorting interface works well on screens with circa 600px or wider. The performance for narrower screens could be improved.

5.4 Sorting Interface

The presentation of groups in the sorting interface could be improved. Currently, groups are defined as `section` elements and cards (and subgroups) as items in an unordered list `ul` within the `section`. Groups are positioned using `float: left;` and the tallest group determines the height of the apparent row of groups. In future, CSS Flexbox or CSS Grid could be used to better control the visual layout of groups.

Furthermore, it is currently not possible for the user to collapse and expand groups temporarily for more clarity. This would increase the usability of the sorting interface on smaller screens when a large number of cards is to be sorted.

5.5 Raw Data when Subgroups

In the Raw Data tab the sort is flattened. For each card the most inner group is displayed and if exported, saved in an CSV file. A change from the inner to the outer most should be done.

Bibliography

Andrews, Keith [2018]. *Information Architecture and Web Usability*. TU Graz. 17 Jan 2018. `http://courses.iicm.tugraz.at/iaweb/iaweb.pdf` (cited on page 1).

Bartel, Torsten [2017]. *It is the user who decides on the success of your products*. usability.de. 2017. `https://usability.de/en/services/methods/card-sorting.html` (cited on page 1).

Catlin, Hampton, Natalie Weizenbaum, and Chris Eppstein [2013]. *SASS: Syntactically Awesome Style Sheets*. 13 Jul 2013. `https://sass-lang.com/` (cited on page 5).

Cederholm, Dan [2013]. *SASS for Web Designers*. A Book Apart, 2013. ISBN 1937557138 (cited on page 5).

Coyier, Chris [2017]. *A Complete Guide to Flexbox*. CSS-Tricks. 12 Nov 2017. `https://css-tricks.com/snippets/css/a-guide-to-flexbox/` (cited on page 3).

Hernandez, Eddy, Chris Burrell, and Evan Sharp [2017a]. *Layout*. 16 Nov 2017. `https://reactstrap.github.io/components/layout/` (cited on page 3).

Hernandez, Eddy, Chris Burrell, and Evan Sharp [2017b]. *reactstrap*. 16 Nov 2017. `https://reactstrap.github.io/` (cited on page 3).

Ljajić, Haris [2016]. "sortit: Web Card Sorting Backend". Master's Thesis. TU Graz, 2016. `http://ftp.iicm.tugraz.at/pub/theses/hljajic-2016-msc.pdf` (cited on pages VII, 1, 7).

Marcotte, Ethan [2014]. *Responsive Web Design*. 2nd Edition. A Book Apart, 2 Dec 2014. 153 pages. ISBN 1937557189. `http://abookapart.com/products/responsive-web-design` (cited on page 3).

Meteor [2017]. *Introduction | Meteor Guide*. 15 Nov 2017. `https://guide.meteor.com/` (cited on page 1).

MongoDB [2017]. *Move at the Speed of Your Data*. MongoDB, Inc. 2017. `https://www.mongodb.com/` (cited on page 1).

Snook, Jonathan [2011]. *Font sizing with rem*. Snook.ca. 2011. `https://snook.ca/archives/html_and_css/font-size-with-rem` (cited on page 5).

Stefanov, Stoyan [2016]. *React Up & Running*. O'Reilly, 2016. ISBN 1491931825 (cited on page 1).

Strukchinsky, Vsevolod and Vladimir Starkov [2017]. *BEM – Block Element Modifier*. 24 Nov 2017. `http://getbem.com/` (cited on pages 3, 5).

TBA [2017]. *Bootstrap*. The Bootstrap Authors. 19 Oct 2017. `https://getbootstrap.com/` (cited on page 3).

Tullis, Tom [2017]. *Card-sorting Tools*. 23 Sep 2017. `http://measuringux.com/CardSorting/` (cited on page 1).