

Technical Interaction Design
KSTEIND1KU

Final Assignment Report

Kristrún Helga Jóhannsdóttir
krijo@itu.dk
Min Cho Dorland
minc@itu.dk

Software Design
IT-University of Copenhagen
Autumn 2021



Preface

This report is the result of group project as part of Technical Interaction Design course for Master Program Software Design at IT University of Copenhagen.

This report documents the whole process of a web application user interface design throughout the semester starting from defining tasks and user requirements from Problem Statement provided in the first week of the semester to the system implementation in React.

The first half of the report is to describe how the group has theoretically approached the user interface design with the empirical project. Developing a minimum viable product has been also carried out where the group has practiced the fundamental programming patterns commonly used in programming user interfaces.

The system we developed, AsoPlan, is to assist in administering activities of annual excursions for two different user groups - both 1) organizers and 2) members of an association. Problems the system tries to tackle include holding excursions, managing duties, car sharing, and a shopping list as part of planning an excursion are assignments the system should solve.

Table of contents

| | | |
|----|--------------------------------------|----|
| 1 | PROBLEM STATEMENT | 3 |
| 2 | TASK DESCRIPTIONS | 7 |
| 3 | E/R DIAGRAM | 9 |
| 4 | VIRTUAL WINDOWS & CREDO CHECKS | 10 |
| 5 | LO-FI PROTOTYPE..... | 14 |
| 6 | USABILITY TESTING REPORT | 22 |
| 7 | HI-FI PROTOTYPE..... | 23 |
| 8 | GRAPHICAL DESIGN DECISIONS | 31 |
| 9 | IMPLEMENTATION | 33 |
| 10 | REFLECTION | 42 |
| | APPENDIX | 43 |

1 Problem statement

1.1 Administration of annual excursions

Some associations organize longer excursions for members and their families. One can for instance organize a weekend trip to a scout hut or a Swedish cottage. The members bring food themselves. Some go by their own car; others try to find someone to drive with.

On such an excursion there are many duties to carry out, e.g., shopping for food and drinks, cooking, washing-up, entertaining, and cleaning before traveling back home. If everyone just helps as he or she pleases, experience shows that some work hard while others only give a hand now and then. Often a negative mood spreads because some people gain on behalf of others. For that reason, it is important to plan who does what so that everyone knows when they have something to do and when they can relax with a clear conscience. It turns out to work well in practice.

When more than 20 people are going, however, it starts being difficult to manage. Who has space for more people in the car? Who has to do which duties? How much food and drink should be bought for 40 adults, 12 teenagers and 14 kids?

The database contains, among other things, the following information about each member:

- first name, last name, address, phone number, mobile phone number, work phone number, email address.

1.2 The role of the organizers

The organizers of the excursion are in principle just ordinary members of the association. They plan where and when the excursion takes place and announce it through the usual news channels of the association. In addition, they create the excursion through the web interface of the excursion system.

They also define the duties of the excursion that need manning. It is hard to remember all the things that need to be done. Even though it is nearly the same duties as last time an excursion was made, it is often forgotten what they are. **Figure 1** shows the duty list from an earlier excursion.

Figure 1. Duty list

| Cottage trip 2007 | |
|--|--|
| Friday-Sunday, June 8-10. Vilshult in Småland. | |
| Duty | Manning (boss is underscored) |
| Shopping in Denmark | <u>Erik R.</u> , Kirsten R., Ole D. |
| Shopping Friday in Sweden | <u>Henrik M.</u> , Bente M. |
| Shopping Saturday in Sw. | <u>Julie G.</u> , Else B. |
| Dinner Friday | <u>Per G.</u> , Annette G., Susan G. |
| Lunch Saturday | <u>Flemming S. P.</u> , Hanne S. P. |
| Dinner Saturday | <u>Anne H.</u> , Niels-Peter J., Jens L., Benedikte D.T. |
| Lunch Sunday | <u>Kirsten L.</u> , Ole M. |
| Bartender Saturday | Annemette N. |
| Table setting Friday | <u>Jens N.O.</u> , Lis N.O., Alice H. |
| Clean up Friday | <u>Niels H.</u> , Alice H. |
| Coffee and tea Friday | <u>Lisbeth R.</u> , Hans R. |
| Breakfast Saturday | <u>Uffe L.</u> , Karina L. |
| Breakfast Sunday | <u>Karen H.</u> , Ole H. |
| Clean kitchen | <u>Klaus B.</u> , Bodil B. |
| Clean shared rooms | <u>Dorthe T.</u> , Svend O. P. |
| Copy songs | Peter S. |
| Accounting | Jesper D. |
| Treasure hunt | Susan R. P. |
| Music system | Henrik M. |

Now the members can sign up themselves and their families for the excursion. When the registration deadline has passed, the organizers distribute the duties among the participants. They try wherever possible to give the participants the duties they have wished for. When the duties have been distributed, they send out a list by email telling who has been assigned to which duties. It can be hard to have an overall view of whether a reasonable number of participants are assigned to all duties and whether the duties are fairly distributed. Some organizers use a spreadsheet, but it is hard to update.

In addition the organizers make a list of what to buy. It is hard to remember what to buy and how much. The idea is to start with the list from the previous excursion and adjust it according to the number of participants and their age. Adjustment must of course also be made according to the menu to be served. **Figure 2** shows part of the shopping list from an earlier excursion.

Another idea is to collect experience data when the excursion has finished. How much was left of bread, milk, beer, lettuce, etc.? And what proved to be insufficient?

Figure 2. Shopping list

| Cottage trip 2007 | | |
|---|--------|-------|
| Friday-Sunday, June 8-10. Vilshult in Småland. | | |
| Adults: 40 Teenagers: 12 Kids: 14 | | |
| Item | Amount | Unit |
| Wheat bread | 12 | kg |
| Rye bread | 10 | kg |
| Rice | 8 | kg |
| Spaghetti | 6 | kg |
| Grill meat | 12 | kg |
| Sausages | 2 | kg |
| Canned fish | 4 | kg |
| . . . | | |
| Plain milk | 10 | liter |
| Light milk | 12 | liter |
| Juice | 15 | liter |
| Yoghurt | 8 | liter |
| . . . | | |

1.3 The role of the participants

A member of the association who wishes to participate in the excursion signs up online. The member can also enlist one or more persons in the family. They are not registered as members of the association, only their name is recorded, not their telephone number and email address. The association member is the **contact person** for the entire family. To make it possible for the organizers to distribute the duties and calculate what to buy, the contact person should state for each family participant whether he or she is an adult, a teenager, or a child.

To achieve a good distribution of the duties, it is assumed that up to three preferred duties for each family participant can be stated. Some, for instance, like to cook, while others prefer to clean up. Some like to get up early in the morning and do not mind taking care of the breakfast. Children of course do not need to have any duties.

The contact person should also state the car (or cars) the family will use, and how many seats are available in each car for other participants.

Participants without a car must be able to find spare seats and reserve them. They need to be able to see who offers spare seats and, for instance, if it is someone living close to themselves.

They of course also need to be able to contact the member in question in order to plan where to meet. It is also practical that they know the color and registration number of the car.

It may happen that a member has to change his or her registrations even after the registration deadline has passed.

Participants responsible for shopping, arrange with the organizers who buys what. It is managed without support from the system.

1.4 Other information

You can assume that there are at most 150 participants in an excursion and that there are at most 6 members in a family. There are at most 50 duties to man. Often there are two or more participants assigned to a duty. In that case, the organizers nominate one of them as the “boss” for the duty.

You can omit everything about accounting. You can also omit everything about accommodation. Moreover, you do not have to worry about access rights on the website for organizers and members. You can assume that everyone behaves reasonably

2 Task descriptions

T1. Plan Excursions

These entail initial practices done by one or a few organizers. Organizers create a new excursion, release and edit the excursion.

- Start: The organizer(s) got informed that the excursion is agreed.
- End: The new excursion is ready to take place.
- Difficult: Medium; Not so many modifications are expected.
- Frequency: 3-5 times over a month of excursion planning period by one or two users and once after the excursion.

| <u>SUBTASKS AND VARIANTS</u> | <u>EXAMPLE SOLUTIONS</u> |
|------------------------------|--------------------------|
| 1. Create excursion | |
| 2. Release excursion | |

T2. Manage Duties

These practices are carried out by organizer(s) and participants. Organizers prepare a list of duties, while participants sign up for the excursion and their preferred duties. Then organizers distribute duties to the participants.

- Start: A new excursion is created (T1)
- End: Every adult participant has been informed of their distributed duties.
- Difficult: Hard; Many participants wish for a single duty (i.e., music playing).
- Frequency: 100 access if 100 participates in. Frequent uses are expected by diverse users.

| <u>SUBTASKS AND VARIANTS</u> | <u>EXAMPLE SOLUTIONS</u> |
|---|--|
| 1. Initiate duty list | |
| 2. Sign up for excursion and preferred duties | Users can select a category of preferred duties (type/time) with i.e., a dropdown. |
| 1. Distribute duties | |

T3. Record Drive

These practices are carried out by a subgroup of participants. Some participants (drivers) have cars and would like to share available seat(s) with other participants (requesters).

- Start: Once participants signed up for the excursion.
- End: Once every request is closed.
- Difficult: Easy.
- Frequency: Max 30 uses by multiple participants.

| SUBTASKS AND VARIANTS | EXAMPLE SOLUTIONS |
|---|-------------------|
| 1. Register car seat(s) | |
| 2. Request seat(s) | |
| 1. Close the request <ul style="list-style-type: none">a. Acceptb. Partly acceptc. Reject | |

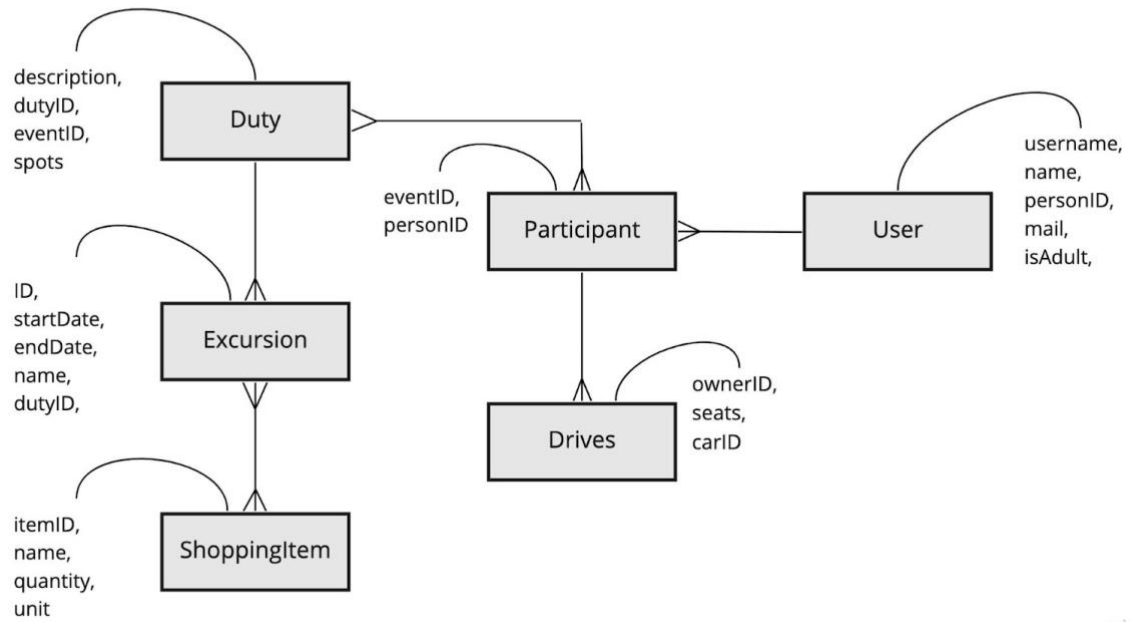
T4. Manage Shopping

These practices are carried out by one or a few organizers.

- Start: Once the sign-up period has expired.
- End: Once the review for the shopping list is complete.
- Difficult: Easy.
- Frequency: Max 5 uses per excursion.

| SUBTASKS AND VARIANTS | EXAMPLE SOLUTIONS |
|--------------------------------|-------------------|
| 1. Pull previous shopping list | |
| 2. Adjust the list | |
| 3. Publish the list | |
| 4. Add comments | |

3 E/R Diagram



4 Virtual Windows & CREDO checks

4.1 Virtual Windows

T1. Plan excursion

SETUP

title:

date:

location:

T2. Manage duties

SIGNUP

name:

adult:

preferred duties:

T2. Manage duties

DUTIES

| Type | Date | # Required | Participants | Head |
|----------|------|------------|--------------|-------|
| Cleaning | 1 | 2 | Chris, Jenny | Chris |
| Cooking | 2 | 3 | Hans, Fie | Fie |
| Plates | 1 | 1 | Mia | |
| ... | | | | |

T3. Record drive

Drives

| Drive by | Requested by | Available seats | Status | Notes |
|----------|--------------|-----------------|-----------|-------|
| Hans | Jenny | 2 | confirmed | |
| Johan | Kenneth | 1 | pending | |
| Alice | Mia | 1 | rejected | |

T4. Manage Shopping

Shopping

| item | qty | unit |
|---------|-----|------|
| Milk | 2 | pcs |
| Lettuce | 10 | pcs |
| beef | 2 | kg |

5 Virtual Windows are identified: Setup, Signup, Duties, Drives, and Shopping

- **Setup:** This window shows general information for an excursion. This virtual window may appear almost as it is in the final computer screens.
- **Signup:** This window displays basic information about a participant, and their preferred duties. If a member wants to bring his/her family members, these input cells (i.e., name, adult, preferred duties) are to be repeated on a single screen.
- **Duties:** The window uses a matrix form (like a spreadsheet). There is a line for each duty. First three columns are default columns to be represented and filled out when

the window is created (by an organizer). In addition, the other columns (participants, head) are added in later views. There may be a long list of duties, so a scroll bar is implemented.

- **Drives:** The window also uses a matrix form. Each line is a drive request, requested by a participant to another driver participant. Available seats are marked as user input to indicate that the driver participant should initially put in. Status of the drive request can be also presented for particular tasks. As there can be a long list of drive requests, a scroll bar is introduced.
- **Shopping:** This window is a simple list of items and their quantity. Organizer(s) can modify the quantity and add/delete lines.

4.2 CREDO checks

4.2.1 Data model versus Virtual Windows

| | Excursion | Participant | Duty | DutyState | Request | Drive | RequestState | Missing Window data |
|---------------|-----------|-------------|-------|-----------|---------|-------|--------------|---------------------|
| Setup | CRED | | | | | | | |
| Signup | | CRED | | | | | | |
| Duties | | | CREDO | re O | | | | |
| Drives | | | | | CREDo | CREDo | re O | |
| Shopping | | | | | | | | |
| Missing fncts | O | O | | (C)D | | | (C)D | |

- **Excursion** entity: A new excursion can be created in Setup window (C), and the information can be read, edited and even deleted (RED). Overview of excursions including previous ones are not supported. Thus O at the bottom line.
- **Participant** entity: A new participant is created, read, edited, and deleted (CRED) in Signup window. However, we cannot get an overview of participants, so we put an O in the bottom line.
- **Duty** entity: All can be done in the Duty window, so there is no mark in the bottom line. Creating and deleting a duty can be done by organizers. Organizers also have a good overview of the duties.

- **DutyState** entity: It is more subtle. Each duty is wanted by some of the participants (C), and the organizer should have a good overview of duty wishes (O) to fairly distribute the duties. The system should automatically create the necessary database records, and Delete does not need to be shown to the users. Thus it is at the bottom line.
- **Request** entity: Participants can create new (ride) requests, read, edit, (CRED) and even have a partial overview of drive requests. Overall or partial overview is a matter of choice, in that users can see (all) ongoing/closed requests and/or requests and drives of all other users as well. (i.e., anyone can see who has requested whose cars etc.,) How much information is allowed to show is a matter of security permissions.
- **Drive** entity: It is similar to Request entity.
- **RequestState** entity: It is more subtle, but it has its one attribute - request status (i.e., accepted, pending, rejected) and it is represented on the Drives window. The system automatically creates the necessary database, so C is at the bottom line. As already closed requests can be hidden from the screen, reading RequestState is marked r instead of R. And e is marked instead of E, as only the driver can edit the status of his/her own car.

4.2.2 Data model versus Tasks

| | Excursion | Participant | Duty | DutyState | Request | Drive | RequestState | Missing task data |
|-----------------|-----------|-------------|--------|-----------|---------|--------|--------------|-------------------|
| Plan Excursion | CRED | | | | | | | |
| Manage Duties | | CRED | CRED O | Re O | | | | |
| Record Drive | | | | | CRED o | CRED o | Re O | |
| Manage Shopping | | R | | | | | | |
| Missing tasks | O | O | | (C) D | | | (C) D | |

- Excursion entity: PlanExcursion creates, read, edit and delete Excursion records (CRED), but misses providing an overview (O).
- Participant entity: Participant can be created, read, edited and deleted (CRED) during a subtask of Manage Duties (Sign up for excursion and preferred duties) done by

participants. Partial overview of participants can be used (R) for Manage Shopping task - i.e., how many adults/kids participate in the excursion.

- Duty entity: Duty is managed through Manage Duties Task.
- DutyState: This entity is more subtle. The system should fully handle this to support Manage Duties task.
- Request entity & Drive entity: Request and Drive are handled in Record Drive task. But o is marked instead of O, because it is a matter of design how much overview we will visualize on the screen. (only ongoing requests? can all see others' request status as well?)
- RequestState entity: The entity is more subtly handled to execute Record Drive task.

5 Lo-Fi Prototype

Asoplan
Excursion
Car
Shopping

Login
ID:
Password:

Initial Page: On the left, there is a sidebar where users can navigate. The 'admin' account is special- they are entitled to publish excursion, set up duties.

Asoplan Hello, admin
[Logout](#)
Excursion
Car
Shopping

Title:

Date: (3 days)

Location:

Description:

Set up Duties

| Type | Day | Manning |
|----------------|------------|---------|
| Set up a table | 1 / dinner | X |
| Coffee and tea | 2 | X |

Excursion page(admin) - Initially set up a new excursion, and duties.

Asoplan Hello, Harry
[Logout](#)
Excursion
Car
Shopping

Title: Autumn Breathe
Date: Sep 12- 14, 2022 (3 days)
Location: Hareskov
Description: Let's go on a hike in Nordsjælland and enjoy the fresh autumn air together :)

Sign up for excursion
Name:
Are you an adult?: ☒ Yes ☐ No
Preferred duties:

Select duty ▼
Cooking
Making coffee
Music System
Treasure Hunt

Select day ▼
Day 1
Day 2
Day 3

+ Add a family
Save

Excursion page (other users) - Can read the information of the excursion, and they can sign up for the excursion. Duty preferences are also set here, while signing up for the excursion.

Asoplan Hello, Harry
[Logout](#)
Excursion
Car
Shopping

Title: Autumn Breathe
Date: Sep 12- 14, 2022 (3 days)
Location: Hareskov
Description: Let's go on a hike in Nordsjælland and enjoy the fresh autumn air together :)

Sign up for excursion
Name:
Are you an adult?: ☐ Yes ☒ No
Preferred duties:

Select duty ▼

Select day ▼

+ Add a family
Save

In case of kids participating, duty preference selection is inactivated.

Asoplan

Hello, Harry

[Logout](#)

Excursion

Car

Shopping

Are you a driver or rider for Autumn Breathe?

Driver

Rider

Car page- Users can choose between two roles- driver/ rider.

Asoplan

Hello, Harry

[Logout](#)

Excursion

Car

Shopping

Car Information

Available Seats: 2 ▼

Notes:

Clear

Save

Driver view – Drivers can set up their available seats and notes.

| | |
|---|--|
| <div><div>Asoplan</div><div>Hello, Harry</div><div>Logout</div><div>Excursion</div><div>Car</div><div>Shopping</div></div> | <div>Your car (2/2 seats available)</div> <div>No request found.</div> |
|---|--|

Once they have no request from other participants.

| | |
|---|---|
| <div><div>Asoplan</div><div>Hello, Harry</div><div>Logout</div><div>Excursion</div><div>Car</div><div>Shopping</div></div> | <div>Your car (2/2 seats available)</div> <div>2 new request(s).</div> <div><div>Ron Weasley requested 1 seat(s).</div><div><div>Confirm</div><div>Reject</div></div></div> <div><div>Luna Lovegood requested 2 seat(s).</div><div><div>Confirm</div><div>Reject</div></div></div> <div><div>Save Changes</div></div> |
|---|---|

Drivers can see the new requests from other participants and make an action (confirm/reject)

| | |
|---|---|
| <div><div><div>Asoplan</div><div>Hello, Harry</div><div>Logout</div></div><div><div>Excursion</div><div>Car</div><div>Shopping</div></div></div> | <div><div>Your car (0/2 seats available)</div><div>Thanks for sharing your car. :)</div></div> |
| <div><div><div>Asoplan</div><div>Hello, Luna</div><div>Logout</div></div><div><div>Excursion</div><div>Car</div><div>Shopping</div></div></div> | <div><div>Are you a driver or rider for Autumn Breathe?</div><div><div>Driver</div><div>Rider</div></div></div> |

Rider view.

Asoplan

Hello, Luna

Logout

Excursion

Car

Shopping

Available Rides

Harry Potter is sharing 2 seat(s).

Cho Chang is sharing 1 seat(s).

Draco Lucius Malfoy is sharing 1 seat(s).

Save Changes

Requested

Cancel

Request

Cancel

Rejected

Cancel

Riders can see all available drives.

Asoplan

Hello, Luna

Logout

Excursion

Car

Shopping

Available Rides

Harry Potter is sharing 2 seat(s).

Cho Chang is sharing 1 seat(s).

Draco Lucius Malfoy is sharing 1 seat(s).

Save Changes

Confirmed

Cancel

Request

Cancel

Request

Cancel

And they can make actions. (Request)

| | |
|--|--|
| <div>Asoplan Hello, Admin Logout</div> <div>Excursion</div> <div>Car</div> <div>Shopping</div> | <div>Car Status</div> <div>Draco Lucius Malfoy has 1/1 seat(s).</div> <div>Cho Chang is has 1/1 available seat(s).</div> <div>Harry Potter has 0/2 available seat(s).</div> |
|--|--|

Admin account cannot do any activities related to cars (request, sign up for their cars) but can see the overview.

| | |
|--|---|
| <div>Asoplan Hello, Admin Logout</div> <div>Excursion</div> <div>Car</div> <div>Shopping</div> | <div>Shopping List</div> <div>Import a previous shopping list:</div> <div><div>Select one ▼<div>20 people / 3 days40 people / 1 day60 people / 2 days</div></div><div>Save</div></div> |
|--|---|

Shopping page - it enables importing a previous shopping list.

Asoplan

Hello, Admin

[Logout](#)

Excursion

Car

Shopping

Shopping List

Fill out a shopping list 40 people / 1 day as reference.

| Item | Qty | Unit | |
|------------|-----|------|---|
| Milk | 5 | L | X |
| Banana | 40 | pcs | X |
| Rice | 5 | kg | X |
| Pasta | 5 | kg | X |
| + Add item | | | |

Clear

Save

From the given list, the admin can adjust it and publish it.

6 Usability Testing Report

See Appendix (from our former hand in).

7 Hi-Fi Prototype

AsoPlan

Excursion

Cars

Shopping

Log out

Welcome to Asoplan
Please login to your account

Username

Password

Frontpage should offer a way to Login to Asoplan.

AsoPlan

Excursion

Cars

Shopping

Admin mode

Log out

Create excursion

Title

Date

Location

Description

Upload png

Set up duties

| Type | Time | People |
|---|---|-------------------------------------|
| <input type="text" value="Set up table"/> | <input type="text" value="Friday morning"/> | <input type="text" value="Text"/> ⊖ |
| <input type="text" value="Cook dinner"/> | <input type="text" value="Friday evening"/> | <input type="text" value="Text"/> ⊖ |
| <input type="text" value="Cleaning"/> | <input type="text" value="Saturday morning"/> | <input type="text" value="Text"/> ⊖ |
| <input type="text" value="Breakfast"/> | <input type="text" value="Saturday morning"/> | <input type="text" value="Text"/> ⊖ |

To create an excursion there needs to be a form to add in the data which is connected to a

database. Also, we need to be able to set up the duties from this page. Only an admin can create an excursion and the admin mode is stated in the sidebar.

AsoPlan

Excursion

Cars

Shopping

Log out

Wonderful Waterfall

Title

Wonderful Waterfall

Date


25/3/2022 - 28/3/2022

Location

Skógafoss

Description

Located on the Skógá river, this mighty cascade is clearly visible from Route 1 and is an excellent place to stop and stretch the legs while travelling Iceland's South Coast. The river below Skógafoss holds a large char and salmon population and is thus a favourite spot for fishermen in the summer.



Sign up for excursion

Name

Harry Potter

Agegroup

☐ Adult ☒ Child

Preferred duties

Select duty ▼

Select duty ▼

ADD FAMILY MEMBER


SAVE

When signing up for an excursion it is important to have information about the excursion available for the participant. When signing up the participant can also add a family member and select duties if the participant is an adult, if the participant is a child it is not possible to select duties.

AsoPlan
Excursion
Cars
Shopping
Log out

Wonderful Waterfall

Title Wonderful Waterfall
Date 25/3/2022 - 28/3/2022
Location Skógafoss
Description Located on the Skógá river, this mighty cascade is clearly visible from Route 1 and is an excellent place to stop and stretch the legs while travelling Iceland's South Coast. The river below Skógafoss holds a large char and salmon population and is thus a favourite spot for fishermen in the summer.



Sign up for excursion

Name

Agegroup ☒ Adult ☐ Child


Preferred duties


| Select duty ▼ | Select day ▼ |
|---------------|--------------|
| Select duty | Day 1 |
| Select duty | Day 2 |
| Select duty | Day 3 |
| Select duty | |

Selecting duties for an adult participant.

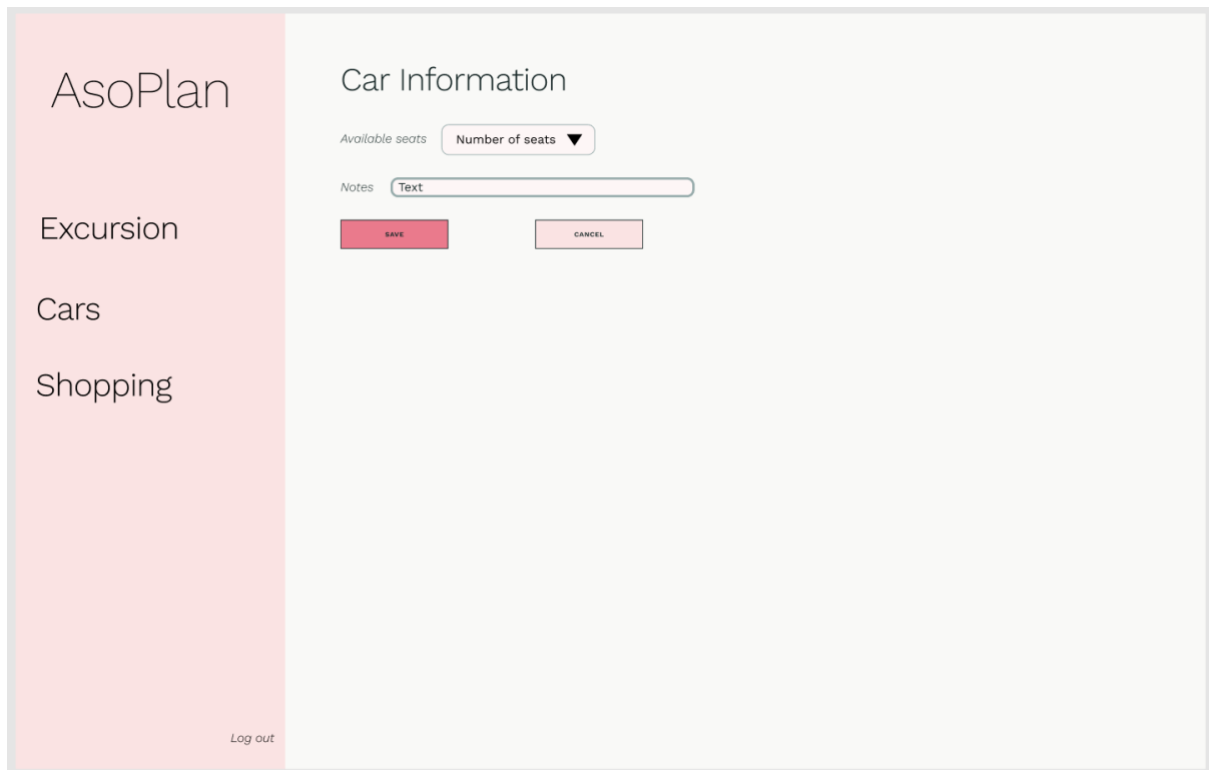
AsoPlan
Excursion
Cars
Shopping
Log out

Are you a driver or a rider for Wonderful Waterfall?


Driver

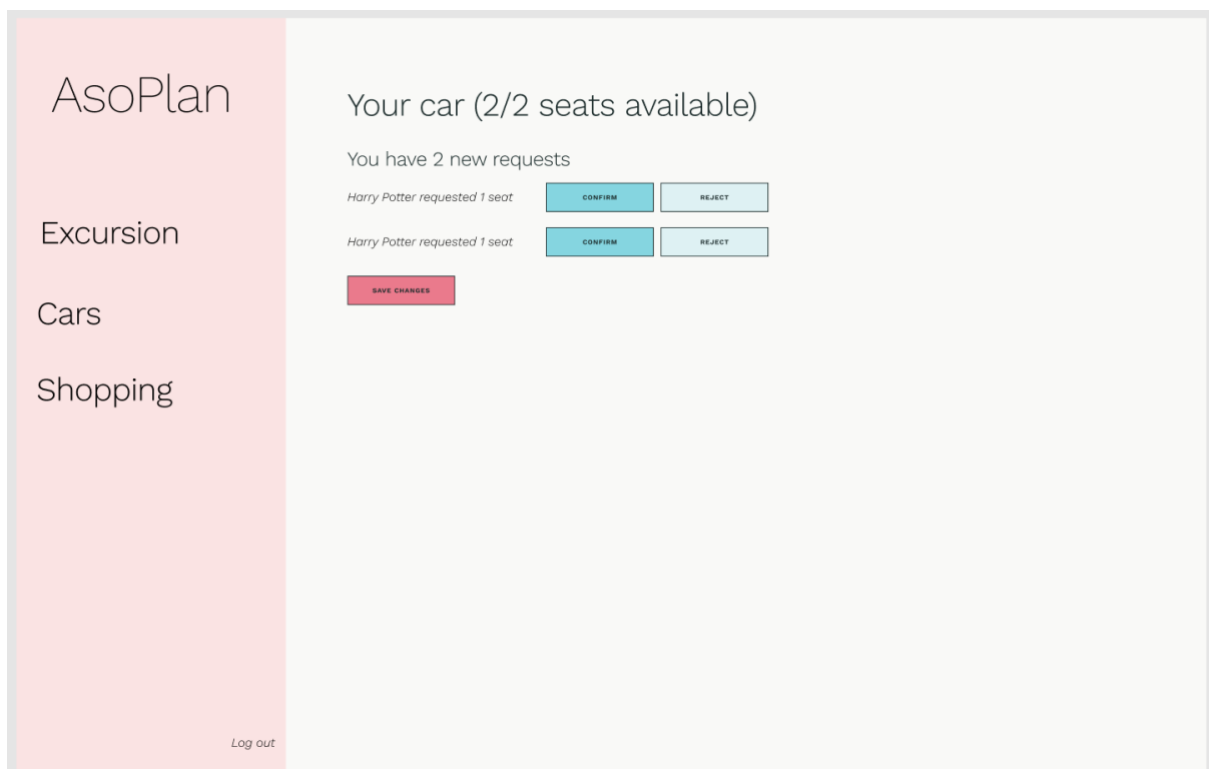

Rider

Participants can choose between being a rider or a driver in the excursion. Drivers provide their own car and offer available seats to the riders.



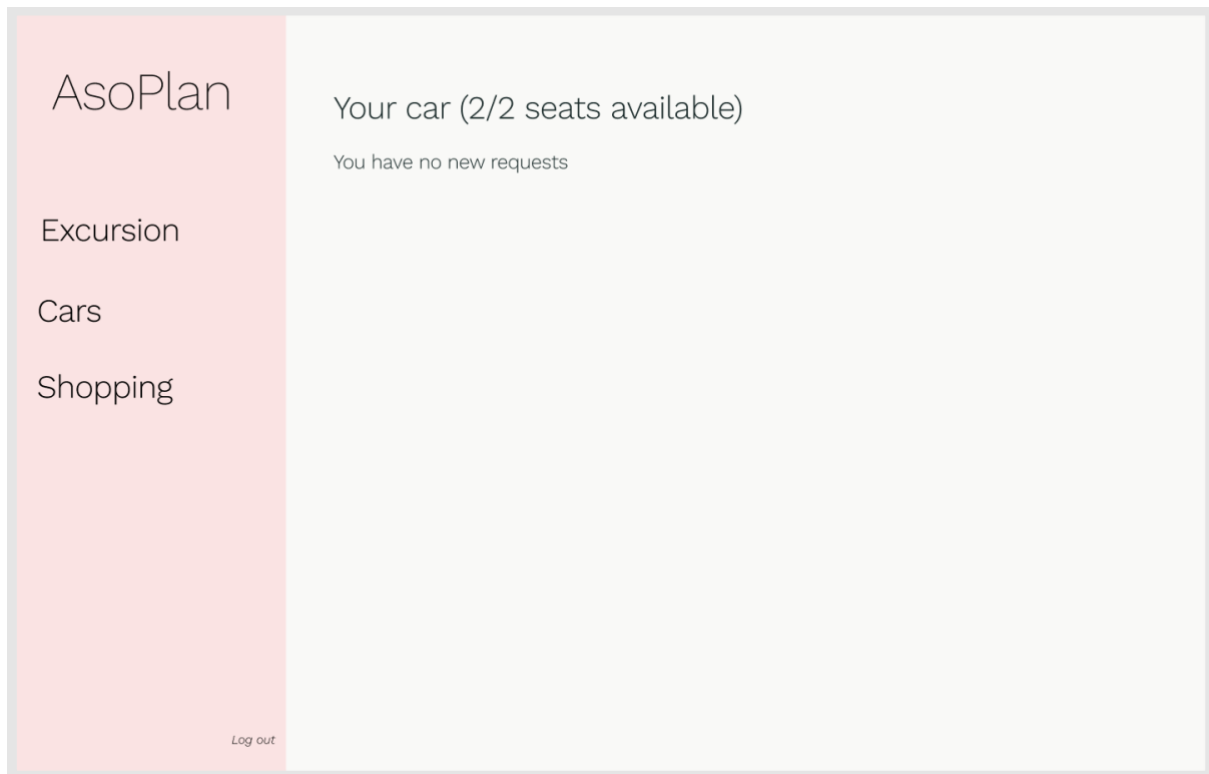
The screenshot shows the 'Car Information' form in the AsoPlan application. On the left is a pink sidebar with the 'AsoPlan' logo and navigation links for 'Excursion', 'Cars', and 'Shopping'. At the bottom of the sidebar is a 'Log out' link. The main content area is light gray and contains the 'Car Information' title. Below the title, there is a label 'Available seats' followed by a dropdown menu currently showing 'Number of seats'. Underneath is a 'Notes' label followed by a text input field containing the word 'Text'. At the bottom of the form are two buttons: a red 'SAVE' button and a light pink 'CANCEL' button.

The driver needs to add information about the status of its car.

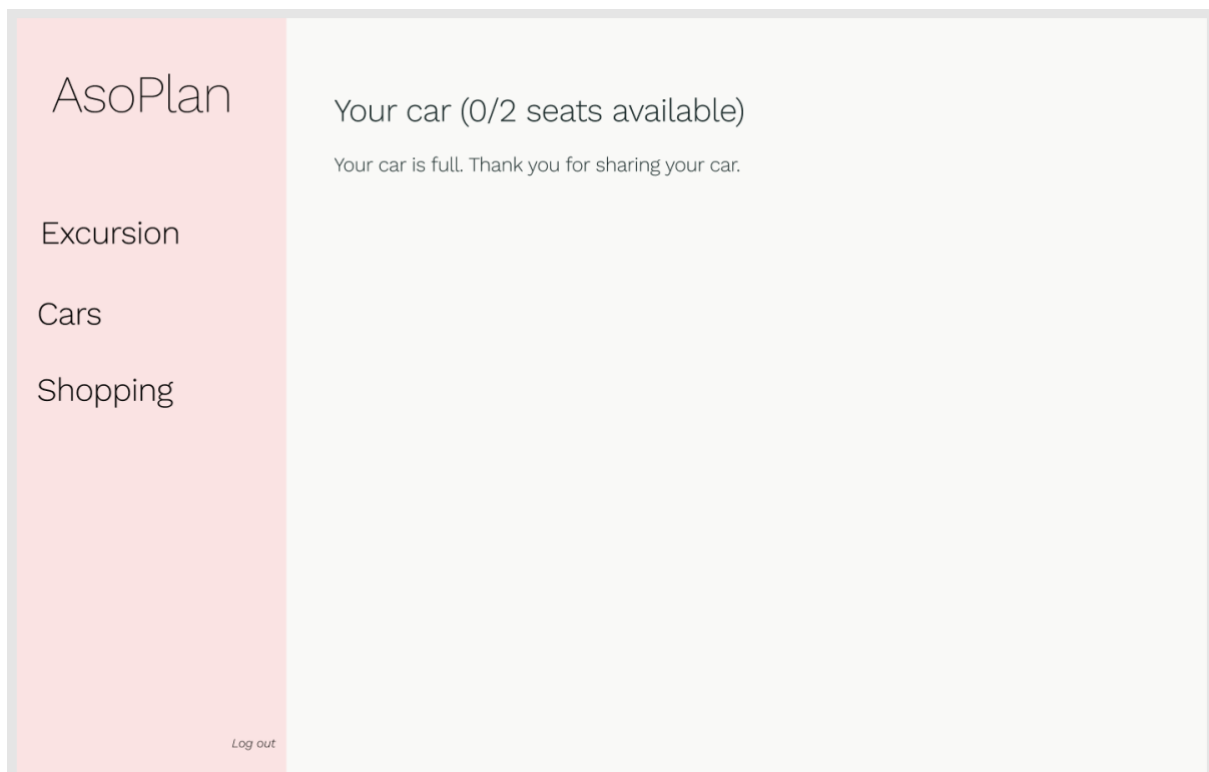


The screenshot shows the 'Your car (2/2 seats available)' screen in the AsoPlan application. The layout is consistent with the previous screen, featuring the same pink sidebar with 'AsoPlan' logo and 'Excursion', 'Cars', 'Shopping' links, and a 'Log out' link. The main content area is light gray and displays the title 'Your car (2/2 seats available)'. Below the title, it says 'You have 2 new requests'. There are two identical request entries, each showing 'Harry Potter requested 1 seat'. For each request, there are two buttons: a teal 'CONFIRM' button and a light blue 'REJECT' button. At the bottom of the request list is a red 'SAVE CHANGES' button.

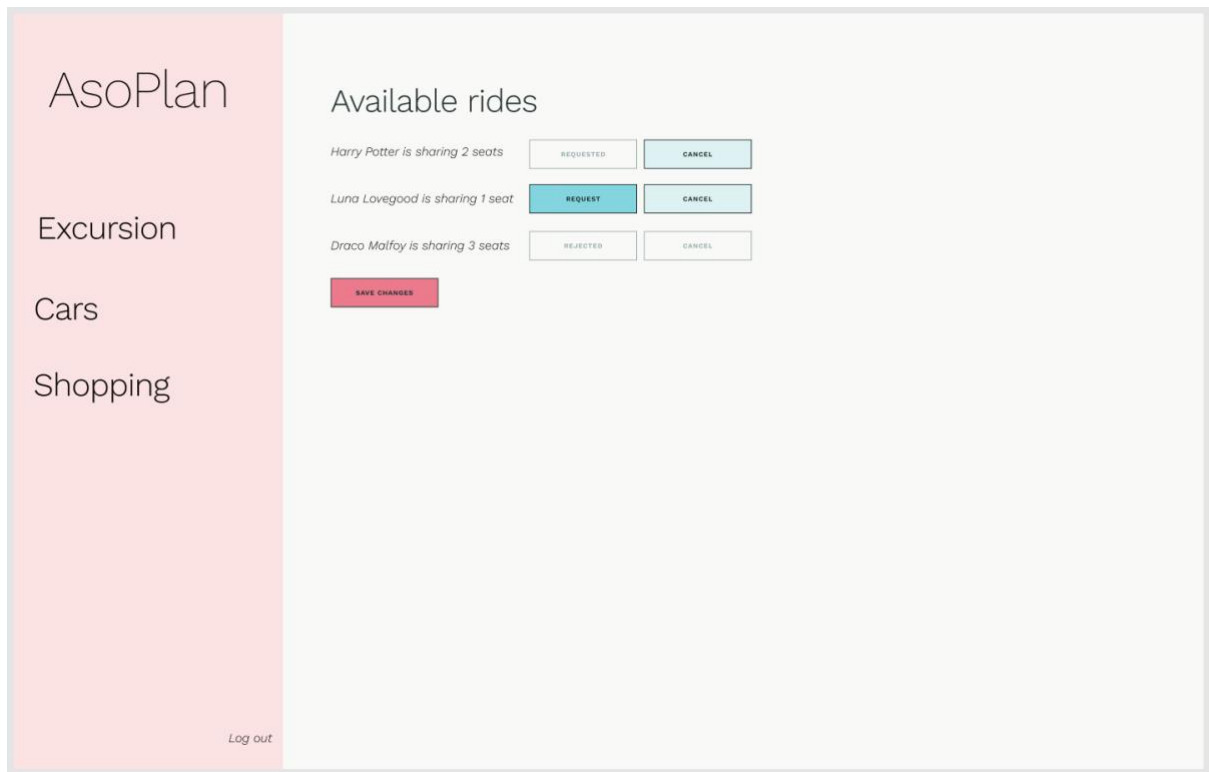
When the information has been added, the driver can see who has requested seats in their car. The driver has the option to both confirm and reject a rider.



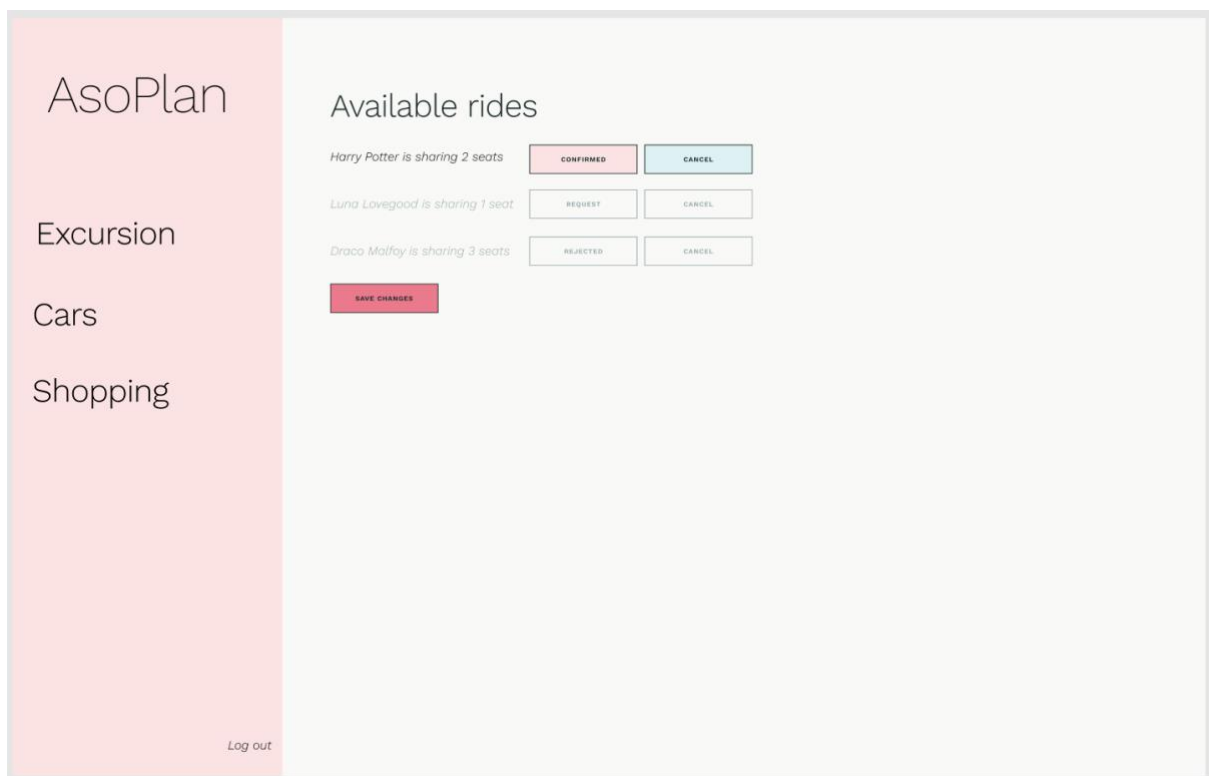
If there are no requests the driver should see this information.



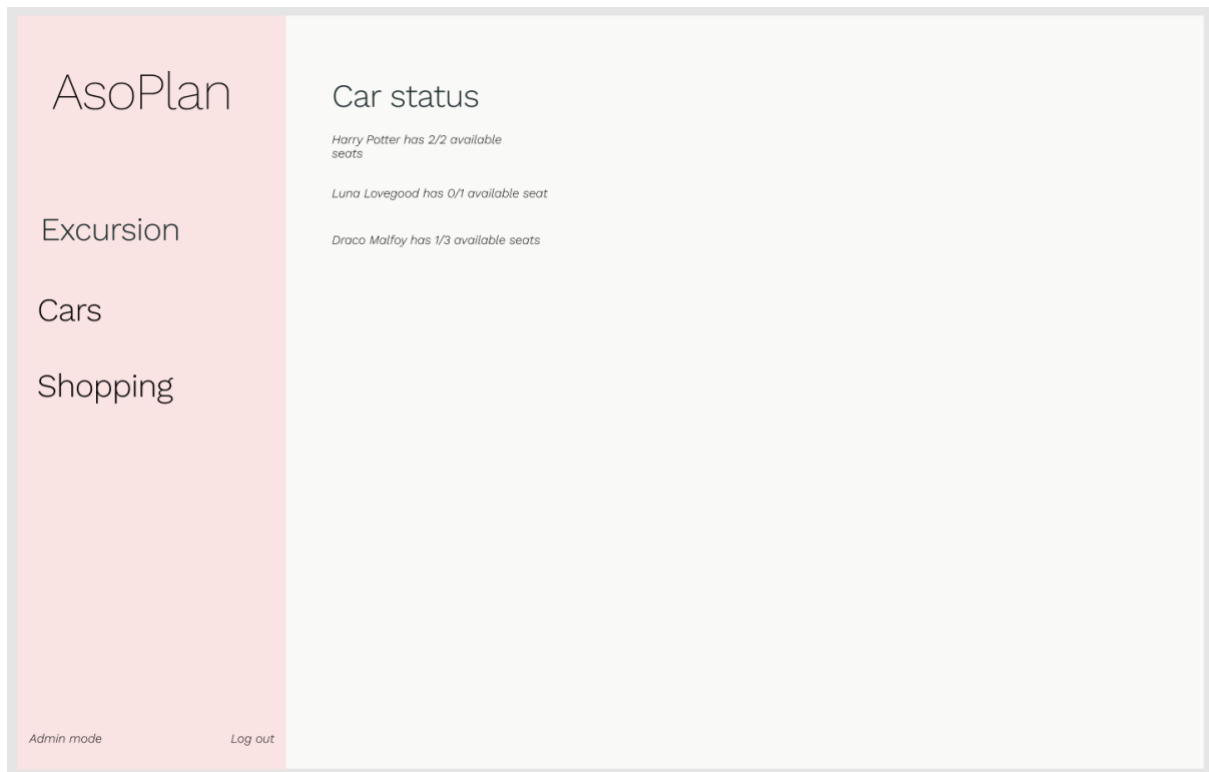
If the all the seats in the car have been filled the driver will be able to see information about that.



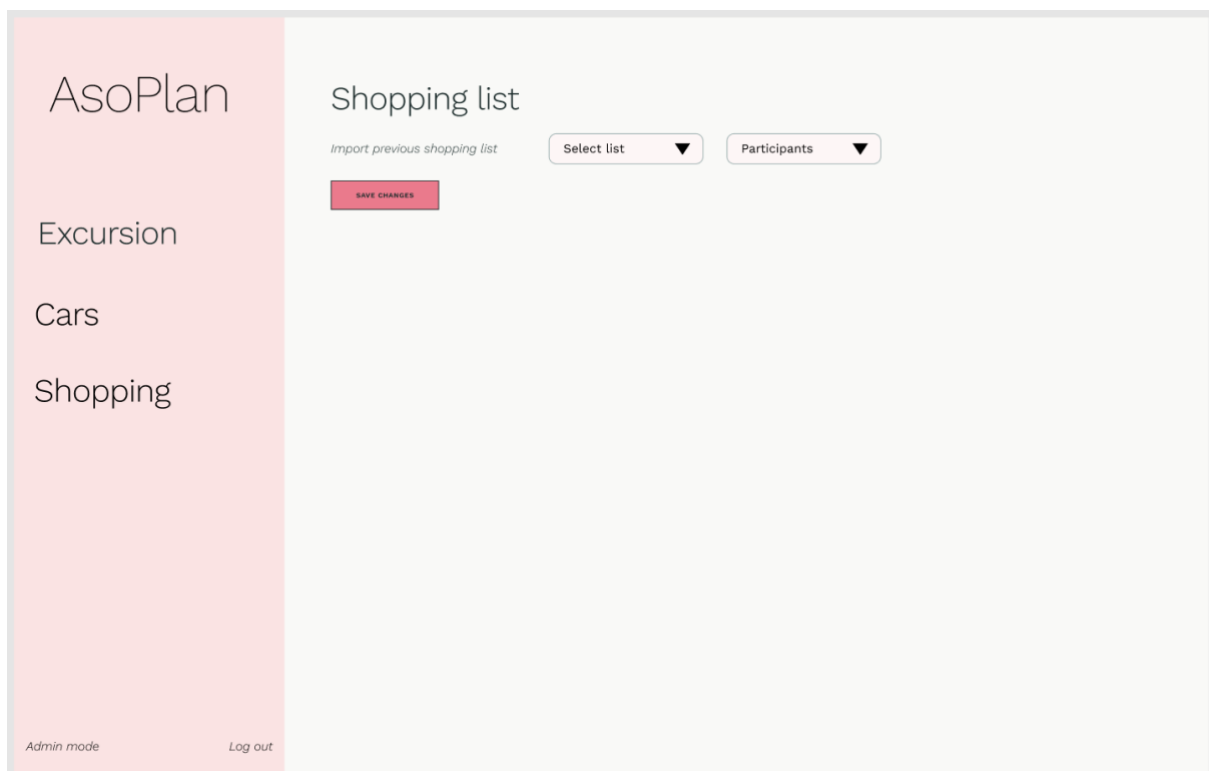
The rider should be able to see a list of available rides where it is possible to request seats.



When the rider has been accepted to a car he should only see that car, unless the rider cancels the request.



The admin should be able to see a list of all the cars in spite of the status of the car.



As an admin the user should be able to import a shopping list from a previous excursion.

AsoPlan
Excursion
Cars
Shopping
Admin mode
Log out

Shopping list

Import previous shopping list

SAVE CHANGES

Days
1 day
2 days
3 days
4 days

Participants
10
15
20
25

This is done by choosing the length of the excursion as well as the number of participants.

AsoPlan
Excursion
Cars
Shopping
Admin mode
Log out

Shopping list

Fill out 2 days trip with 40 participants

| Item | Quantity | Unit | |
|----------------|----------|------|---|
| Milk | 10 | L | ⊖ |
| Rice | 10 | kg | ⊖ |
| Banana | 80 | pcs | ⊖ |
| + Add new item | | | ⊖ |

SAVE

CLEAR

The shopping list should be able to render data from the database and show it as a table, where it is possible to delete and add items to the shopping list.

8 Graphical Design Decisions

The main principle behind our design system is that it should be calm, simple and welcoming. Since members with all kinds of backgrounds would be using our application, we tried to stick to common patterns that will be familiar to anyone that has used a modern web application before.

8.1 Colours

Since the association is organizing trips to nature we went with a green palette for the main color, supplemented by a contrasting red used for the 'administrator' organizer interface. With this big difference we try to make it very clear whether a user is in 'participant' mode or 'organizer' mode, since the organizers of a trip will almost always be participating as well. To complete the colors there are two accent colors, orange and yellow. By using shades of similar and not too many colors we tried to keep the interface calm and bright. The general color for standard users is green, with the yellow accent color used to highlight elements that require the user's attention. This includes both important information (such as the details of the trip), and any form elements and dialogs that wait for user input. When switched to organizer mode the interface will use the red color scheme, which makes it explicit that the user now switched to the "backside" of the interface.

8.2 Typography

For the typography we are using two fonts that we found easy to read, and that go well together. For titles this is "Raleway" and for general text "Montserrat". To keep it simple we only used a limited number of sizes, since the application does not require many different styles. There is a title and heading, body text and a bold version of the body text.

8.3 Icons

The icons are taken from the "Line Icons" set, which we picked for its simplicity and simple style, to keep the feeling light.

8.4 Spacing

For spacing we picked 24px as a base size and used multiples to create the different predefined sizes (0,25; 0,5; 0,75; 1; 1,5; 2; 3; 4; 6; 8).

8.5 Components

We attempted to keep any components we produced simple. One of the main components is the navigation/header bar which is combined with a simple line as a footer to frame the pages. The menu items are simply titles in the heading style. Further we created an overview component which presents the excursion's most important details. These components and the dialog/modal are square and use different shades of the same color. For more dynamic components such as buttons and form elements (interactive), and the table views we added some slightly rounded corners to avoid a too formal/business feeling.

9 Implementation

9.1 Source Code

The source code is at <https://github.com/mindorland/reexam>.

9.2 Development Status compared to Prototypes

Duty distributions are left for the future work. Users still set up their duty preferences while signing up for the excursion, but we have not reached to implement the distribution of the duties, and expanding duty list etc.,

Other than that, creating / browsing excursions, signing up for the excursions for the members and the family members of the family, offering/requesting car rides are implemented to a satisfactory level.

The app is not deployed, which means that it is still in development mode.

9.3 Accessing the Application

You clone the project and follow:

```
npm install → npm start.
```

9.4 User for testing

Logging as **admin** (password: admin) you will have organizer rights. Here you can create an excursion.

Logging as **rider** (password: rider) you can sign up for your participation for the excursion. In addition, you can add a rider's family members if you want.

9.5 Excursion states & Responsibilities

To give a better understanding of the life cycle of different components in AsoPlan, one must understand the different states an excursion can have.

The excursion is created, and the duty list is also published by 'organizer' for the other users. Then the new excursion is open for sign up.

The signups and duty preference set ups are closed. Cars are still open for sharing/requesting between other users.

'Organizer' can distribute the duties and work on the shopping list.

9.6 Index.js

Index.js is a top-rooted file where we initialized Parse App, enabling the entire app to be connected to the backend.

9.7 Routing

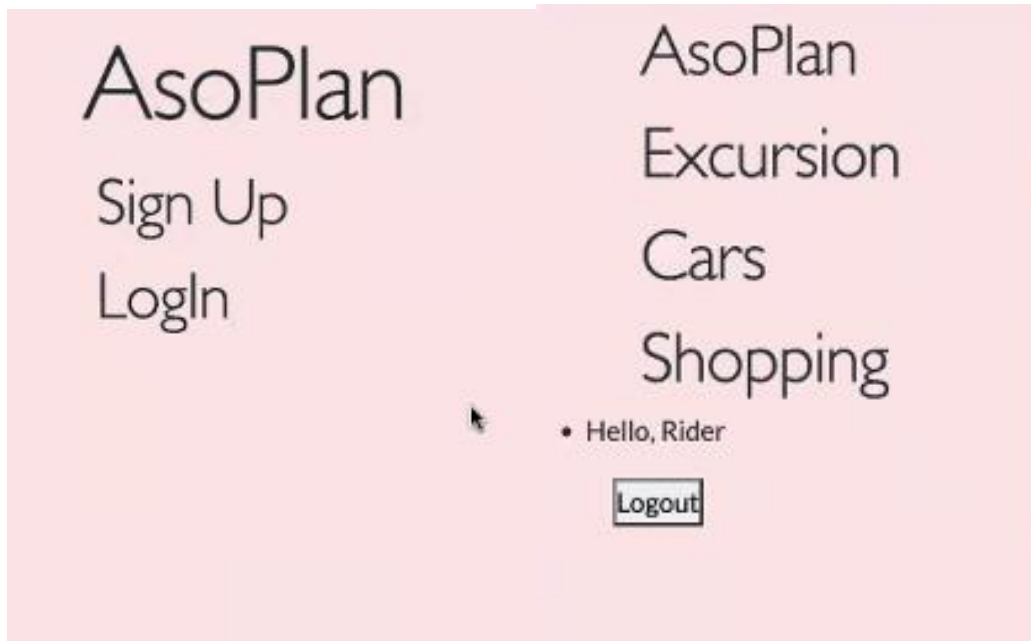
The system is managed in App.js. This page handles all the routings.

```
function App() {
  return (
    <>
      <BrowserRouter>
        /* <Navbar /> */
        <Routes>
          <Route path="/signup" element={<SignUp />} />
          <Route path="/login" element={<Login />} />
          <Route path="/" element={<Home />} />
          <Route path="/excursion" element={<Excursion />} />
          <Route path="/cars/driver" element={<Driver />} />
          <Route path="/cars/rider" element={<Rider />} />
          <Route path="/cars/driverstatus" element={<DriverStatus />} />
          <Route path="/cars/driverscomplete" element={<DriverCompleteSt />} />
          <Route path="/cars/riderstatus" element={<RiderStatus />} />
          <Route path="/cars" element={<Cars />} />
          <Route path="/shopping" element={<Shopping />} />
        </Routes>
      </BrowserRouter>
    </>
  );
}
```

9.8 Signup & Login

While a homepage is an initial page coming up when opening the application, the actual work can be done once a user is logged in, or signed up.

Sidebar shows only home (AsoPlan), Sign Up, and Log in when there is no current user in the database.



(left) - not logged in (right) logged in with rider account.

9.9 Shopping

When implementing the Table component for the shopping list we stumbled into a few problems. Our initial solution was to use the Button, Input, and List components from the Ant Design library. That solution was not perfect, since we could not get the table to render the data correctly.

| Title | Quantity | Unit | Delete |
|---|---|---------------------------------------|--------------------------------------|
| Wheat | kg | 10 | <input type="button" value="X"/> |
| Bananas | pcs | 8 | <input type="button" value="X"/> |
| <input type="text" value="New Shopping"/> | <input type="text" value="New Quantity"/> | <input type="text" value="New Unit"/> | <input type="button" value="+ Add"/> |

With this approach we could make new shopping items in the database as well as deleting items from the database. After some thought, we decided to cut out the Ant Design Library and only use React Bootstrap. With that change the problems with the table became more significant since the Form component in react-bootstrap does not work within a table and for some reason our solution to render the data from before did not work with ListGroup from react-bootstrap.

```

{readResults !== null &&
  readResults !== undefined &&
  readResults.length > 0 && (
    <List
      dataSource={readResults}
      renderItem={(item) => (
        <div>
          <td>{item.get("title")}</td>

          <td>{item.get("unit")}</td>
          <td>{item.get("quantity")}</td>

```

We fully know that there is a solution out there for us but because of time we could not manage to figure it out before the hand in, thus the plan is to work more on the table after the handin and have a fully functioning component before the exam.

9.10 Navbar.js

The sidebar is working with conditional rendering depending on whether the user is logged in or not. Using `useNavigate`, it directs the user to the right page. Also Sign out button is implemented in case the user is logged in.

9.11 Excursion.js

This page is to show different components depending on the user and state.

- If it is an admin and there is no excursion created previously, `InfoWrite` component will be shown, where the organizer can create a new excursion. Here, the Duty table should have been inserted as well.
- If it is an admin and he/she has already created the excursion, `InfoRead` component will be shown as well. The Duty table should have been inserted as well.
- If it is not an admin, currently, `InfoRead` and `ExcursionSignup` components will be shown, where users can see the excursion information and sign up for the excursion.

```

return (
  <div className="excursion">
    {username === "admin" && isExcursion === false && <InfoWrite />}
    {username === "admin" && isExcursion === true && <InfoRead />}
    {username !== "admin" && (
      <>
        <InfoRead />
        <ExcursionSignUp />
      </>
    )}
  </div>
);
}

```

9.12 InfoWrite.js

This page is for the organizer(admin) to create a new excursion by making a subclass of Excursion in the database. The admin uses Forms to fill out title, startDate, endDate, location, description, and an image. In terms of the states of the application, this page exists for the very first state.

```

<Form>
  <Form.Group className="mb-3" controlId="formBasicUsername">
    <Form.Label>Title</Form.Label>
    <Form.Control
      type="text"
      onChange={(e) => setTitle(e.target.value)}
      placeholder="Title"
      autoFocus
    />
  </Form.Group>
  <Form.Group className="mb-3" controlId="formBasicPassword">
    <Form.Label>Date</Form.Label>
    <Form.Control
      type="date"
      onChange={(e) => setStartDate(e.target.value)}
      placeholder="Date"
    />
    <Form.Control
      type="date"
      onChange={(e) => setEndDate(e.target.value)}
      placeholder="Date"
    />
  </Form.Group>
</Form>

```

9.13 InfoRead.js

This page is available after a new excursion is created through InfoWrite.js. And this page can be shown to both admin and other users. It queries Excursion and shows the data to give an

overview of the excursion. Fetching the excursion information is made using `useEffect` working at the first load.

```
const getExcursion = async () => {
  const Excursion = Parse.Object.extend("Excursion");
  const query = new Parse.Query(Excursion);
  const res = await query.find();
  // const Image = Parse.Object.extend("Image")
  // const imageQuery = new Parse.Query(Image)
  if (res) {
    // setExcursion(res)
    console.log(res[0]);
    console.log(res[0].id);
    const newQuery = new Parse.Query(Excursion);
    newQuery.get(res[0].id).then((ex) => {
      setExcursion(ex);
      //console.log(ex.attributes.title)
      setTitle(ex.attributes.title);
      setStartDate(ex.attributes.startDate);
      setEndDate(ex.attributes.endDate);
      setLocation(ex.attributes.location);
      setDescription(ex.attributes.description);
      //setImageFile(imageQuery.get(ex.attributes.image.id))
      //console.log(ex.attributes.image.id)
      // imageQuery.get(imageFile.id).file().link()

      //console.log(ex)
      setUrl(ex.get("image").get("file").url());
    });
  }
};
```

9.14 ExcursionSignup.js & SingleParticipant.js

Signup page is working with these two files. `SingleParticipant` first shows a participant's signup. And the data is saved in the 'Participant' class. Each user fills out their name, age group, and favored duty. And the signup component is expanded by adding a participant.

Besides storing each participant in the database, the contents to show on the screen is saved by operator to keep all the previous other family members' forms.

```

<DropDownButton
  id="dropdown-basic"
  title="Select Favorite Duty"
  onSelect={handleSelect}
  autoClose={false}
>
  <DropDown.Item href="#/action-1" eventKey="cleaning">
    Cleaning
  </DropDown.Item>
  <DropDown.Item href="#/action-2" eventKey="cooking">
    Cooking
  </DropDown.Item>
  <DropDown.Item href="#/action-3" eventKey="settingATable">
    Setting a table
  </DropDown.Item>
  <DropDown.Item href="#/action-4" eventKey="makingCoffeeTea">
    Making coffee/tea
  </DropDown.Item>
  <DropDown.Item href="#/action-5" eventKey="shopping">
    Shopping
  </DropDown.Item>
  <DropDown.Item href="#/action-6" eventKey="playingMusic">
    Playing Music
  </DropDown.Item>
</DropDownButton>
</>

```

9.15 Cars.js

It is the initial page of the car-related pages. Users can choose a role between driver and rider.

It simply navigates to cars/driver, or cars/rider page, respectively.

```

return (
  <Card style={{ width: "50rem" }} className="card-container jc-center">
    <div class="flex-parent jc-center">
      <Card.Title className="header1">
        Are you a driver or a rider for the excursion?
      </Card.Title>
    </div>
    <Card.Body className="jc-center">
      <div class="flex-parent jc-center">
        <Button
          className="car-button"
          onClick={handleDriverClick}
          type="submit"
        >
          Driver
        </Button>
        <Button
          className="car-button"
          text="subtitle"
          onClick={handleRiderClick}
          variant="primary"
          type="submit"
          style={{ background: "#fa728b" }}
        >
          Rider
        </Button>
      </div>
    </Card.Body>
  </Card>
)

```


9.16 Driver.js

This is for the user to register their car. They can set the number of rides, and notes and the data is stored in Drives class.

Our ambition was to control the integer number of available rides and reduce the number once the rider requests and the driver accepts the request.

```
<h3>Available Seats</h3>
<DropDownButton
  id="dropdown-basic-button"
  title="Dropdown button"
  onSelect={handleSelect}
>
  <DropDown.Item eventKey="1">1</DropDown.Item>
  <DropDown.Item eventKey="2">2</DropDown.Item>
  <DropDown.Item eventKey="3">3</DropDown.Item>
  <DropDown.Item eventKey="4">4</DropDown.Item>
  <DropDown.Item eventKey="5">5</DropDown.Item>
</DropDownButton>
<h4>You selected {seats} seats</h4>
{/* maybe change this to conditional rendering to show only if seats are not null */}
<Form>
  <Form.Group className="mb-3" controlId="formBasicNotes">
    <Form.Label>Notes</Form.Label>
    <Form.Control
      type="text"
      onChange={(e) => setNotes(e.target.value)}
      placeholder="i.e., Departure place/time"
      autoFocus
    />
  </Form.Group>
</Form>
```

9.17 DriverStatus.js

Once the current user registered their car, the user is redirected to this page instead of Driver.js. Here you can query all the driveRequest, filtering only the requests which contain the current users' id as driver id. And it shows in 'list' 'Requests from XX'. Initial ambition was to implement both 'Accept' and 'Reject' for each request, and store the final matches between driver and rider in the database. It however turned out disabling all the other new activities by the driver user once the user clicks 'Accept' button, and the logic ends. Currently, the final accept is not talking to the database.

```
const getRequests = async () => {
  // const DriveRequests = new Parse.Object.extend("DriveRequests")
  const query = new Parse.Query(DriveRequests);
  query.equalTo("driver", Parse.User.current().id);
  const results = await query.find();
  setRequests(results);

  const Users = new Parse.Object.extend("User");
  const userQuery = new Parse.Query(Users);
  const userQueryResults = await userQuery.find();
  setUsers(userQueryResults);
};
```

9.18 Rider.js

This page is activated if the user selects the 'rider' role. Users can see all the available drives by querying Drive class. And they can see 'XX is sharing YY rides' with their added notes.

```
<Card
  style={{ border: "none" }}
  style={{ width: "50rem" }}
  className="card-container"
>
  <p className="ptitle flex-parent jc-center">Available rides </p>
  {drivers && (
    <ListGroup
      className="ptitle flex-parent jc-center"
      style={{ border: "none" }}
    >
      {drives.map((drive) => (
        <ListGroupItem
          style={{ border: "none" }}
          className="subtitle picture-margin jc-center flex-parent"
          key={drive.id}
        >
          {
            drivers.find((x) => x.id === drive.get("driver").id).attributes
              .name
          }{" "}
          shares {drive.get("remainingSeats")} seats.
          <RequestCancelBtn
            drive={drive}
            key={drive.id}
            requestRide={handleRequest}
            cancelRide={handleCancel}
          />
        </ListGroupItem>
      )
    )
  )
}
```

10 Reflection

As soon as we decided to aim for the re-exam instead, we started out from the scratch, reflecting on the work we did for the last hand-in. One of the challenges for the last hand in was that we needed to rework for the previous assignments while continuing on the new work, and it inevitably led to inconsistencies between the earlier and later work.

That challenge has been mitigated for the reexam, and we had a clear overview, tasks, and visions of what we need to accomplish.

Another advantage was that we learned that it is very difficult to change the layout, data models once you have prototypes. While recognizing that it is not an ideal implementation from the previous hand in, it was too late to make a modification. Thus, we spent a lot of effort to come up with a better design from theoretical parts and prototypes.

Implementing React with Back4App was a challenging but rewarding process. The application does not have a full implementation as we aimed during the design process due to time constraints, but we are aware of what to do in the future. Back4App was a truly pleasant experience, due to its simplicity and straightforwardness. The only problem we had with the tool was the concept of 'Pointer' and deleting all the rows after a specific query. (i.e., deleting all the driveRequests with the driver of XX.)

Appendix

Usability Testing Report

Planning the test

| | |
|---------------------------|--|
| Which users | 1 user |
| Domain knowledge: | experience in booking systems (i.e., flights), grocery systems |
| IT knowledge: | basic IT literacy (active in using various systems on the web, or their phones) |
| Finding test users | Actual potential users |
| Test site: | ITU meeting room (a closed space, where facilitators and testers can focus on), alternatively online |
| Facilitators: | Main contact with user. – Jens <ul style="list-style-type: none">• Giving a short brief introduction/instruction to the users before test• Ask the users if they seem stuck. |
| ‘Computer’: | Designer (Niek) <ul style="list-style-type: none">• When the tasks change from one to the other, provide them with a right screen on Figma.• Designer might assist Facilitator and ask questions to the users if necessary.• So, Facilitator and Computer work in a team. |
| Log keeper: | Notes down – Min, Kristrun <ul style="list-style-type: none">• Writes down where the user encountered problems, what user believes about the system.• Minimize the interruption while the test is being carried out. |
| Test tasks: | <p>T1. You are planning to sign up for an event for yourself and your whole family. Your name is Min, and you want to bring Jens, and Kristrun to the excursion. You are also willing to offer two seats for other members.</p> <p>T2. You choose your duty preferences up to four, as every adult participant must contribute to the excursion by having one or more duties during the excursion – such as breakfast preparation, and cleaning.</p> |

T3. As you do not have a car and you are going to the excursion alone, you want to see if you can take someone else's ride. You request Johannsdottir's car. Then, just assume that you found another solution for the transport, so you may cancel the request.

T4. You, as an organizer, make a list of duties. Could you add a Friday dinner with two people?

T5. You, as an organizer, distribute duties when the deadline for sign up passed. Could you assign Friday dinner duty to a person who does not have any duty yet but had wished Friday dinner? After that, you assign the same person as a boss as well.

T6. You, as an organizer, create a shopping list based on the number and demographics of participants. First you import a previous excursion's shopping list, and you can either choose to adjust figures or import the previous shopping list as it is.

Presenting tasks

Written – to let the users revisit the tasks when needed.

Start-up state

T1. The system is already logged on with Min's account and knows the family members' names.

T2. The system already knows that Min's whole family signed up for the excursion.

T3. There are two cars available for car sharing.

T4. The excursion is created, and some of the duties are already listed.

| | |
|--|--|
| | T5. All other duties are filled up with the participants. |
| | T6. Previous shopping list(s) should be available. |
| User instruction | Tell the user roughly the same as we would have written. |
| Test method | Think aloud. Facilitator asks the user to think aloud and explain what he does and why before the test begins. Facilitator asks him to explain in case you cannot understand what he is doing or why he gets stuck. |
| Data collection | Video recording |
| What to say (1) – introduction | <p>(Intro of the system)</p> <p>We have been building an annual excursion planning system. Members can sign up for an organized event, and bring their family if they want, but make sure that every adult participant has to have a duty. You can also share a car seat for other members.</p> <p>(Objective)</p> <p>We want to find where the annual excursion planning system is hard to understand or inconvenient. From this week, we actually start programming, so we want to get proved how the system is. But we know the system too well, so we cannot see it ourselves. We need your help. If you have problems with the system, it is the system's fault – not yours. We are not testing you. We are testing the system.</p> |
| What to say (2) – Sequence of tasks | <p>We have prepared 5 tests, and we expect the task would take about half an hour. Let's start with the first task (and show the written task description.)</p> <p>Well done, thank you. Let's try another task.</p> |
| What to say (3) – | The questions to the users are listed below, but not limited to : |

debriefing

- How was the test? Is there anything you like about the system? Something you don't like?
- Do you think the system can do its work for distributing duties?
- Could you sign up for an excursion?
- Was car reservation easy to figure out?
- Are there any places where it seems too cumbersome to use it?

Instead, the questions for debriefing are going to be more extensive to test the users' understanding of the system, to see if the users like the system, and to see if the users can inspire us to make the system more suitable.

Test report

Observed problems marked with tasks and severity.

P1: (T1 - minor)

Not obvious to recognize the difference between 'our entire family will join' and individually selecting family members' names.

P2: (T1 - annoying)

As the member's name (Min) did not appear the first, it was not clear who is the association member (representative of the family).

P3: (T2 - medium)

Duty selection section (in two rows) is displayed next to members' names, so it took a while to understand the duty selection section is active for the first member only. Placing duty selection section below each name is suggested.

P4: (T2 – medium)

It did not seem clear if the selection is saved and completed.

P5: (T2 – set-up error)

Kid name is up on the duty selection page.

P6: (T4 – annoying)

Duty list is already stored in a table format, but user had to type in a separate section. The wordings such as 'type of duty' and 'duty type' were mixed up as well.

P7: (T4 – missing functionality)

User did not know how to complete the task. There was a button missing such as 'Save and exit.'

P8: (T5 – task failure)

User did not understand how two different icons (one with grey background, and the other with plus sign) distinguish at the first glance. With the facilitator's guidance, the user could click the plus sign, to choose a person.

P9: (T5 – Cumbersome)

When the plus button is activated: there is no way to close the pop-up window, until the user clicks a person.

P10: (T5 – Annoying, Bug)

An extra column to determine a boss for each duty seemed extraneous. While the system shall only show boss candidates among the ones who are already assigned for the duty, the system showed all the participants. Removing boss column and marking boss differently on the existing participant column was suggested.

P11: (T5 – Missing functionality)

User cannot add, modify, and delete a duty.

P12: (T6 – Minor)

Importing the previous shopping list was placed bottom right, so the user actively tried to find the button.

P13: (T6 – Minor)

'Import as-is', and 'Import adjusted' were not intuitive. It took a minute to understand the intention of the button.

P14: (T6 – Missing functionality)

User cannot add, modify, and delete shopping item on the table.

10.1.1 General Problems:

The user found that the system lacks 'save and exit' functionalities. The add/modify/remove functionalities were found insufficient. We should take these into the development phase. Our prototype has applied horizontal display to represent different information, while keeping a lot of information in a table format. Thus, vertical display is worth considered for the actual implementation.