ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

Faculty of Engineering,
Computer Science and Psychology
Institute of Psychology and Education
Human Factors

Protoype Development: Shopping

Report at Ulm University

Presented by:
Mariana Santos, mariana.dos-santos@uni-ulm.de
Carolin Schindler, carolin.schindler@uni-ulm.de

Primary instructor:
Prof. Dr. Martin Baumann

Secondary instructor:
M. Sc. Marcel Woide

Psychology of Automation, winter term 2020/21

Last updated March 26, 2021

Typesetting: PDF-LaTeX 2ε

Contents

1. Introduction

Life these days can be so overwhelming that every little detail that can be dealt with others help or even an automation assistance could help to safe time for more important things, also decrease workload and stress. As a master student for example, it is quite difficult to manage time, being even harder when stress is added to the context.
Thinking about all the activities that daily can take time we came up with this idea of an app that could help one manage its daily meals. This way it would be possible to spend less time thinking about all decisions involved in preparing the food, like deciding the weekly recipes to prepare, the necessary ingredients to buy and also going shopping for this items.
Essentially, the prototype will let the user select the meals the individual wants to prepare and it will purchase all the items after finalising the selection for the upcoming week. It will also be possible to inform the number of people a specific recipe will be prepared for and as a consequence of that the new number of ingredients will be shown on the list of ingredients (calculation based on the product of recipe number and number of people).

When opening the app four basic options will be shown: plan meals, add groceries, shopping list and user profile. On this report we will focus on "plan meals" as our main case. Moreover, it will be discussed here the important theories for automating this prototype, the problems we faced when thinking about the development and the found solutions.
The most relevant theoretical concept described here for developing "the shopping" prototype is situation awareness. In order to select and buy the meals it will be necessary to access the calendar that will inform the user all that is fundamental to know, such as, available days to select, which days have meals already chosen, and some more information. It is important to provide ways for the individual to perceive all this data, comprehend what this information mean in order to plan what the next steps will be. The concepts of Trust, Ironies and workload are also relevant for the development of the prototype. All this concepts will be explained along with their implications to our main-case, and possible solution to errors that might occur will be presented and discussed in order to improve our prototype.

2. Description of the Prototype

The shopping application includes multiple use-cases which are listed in the next section followed by the description of a persona for our application. Afterwards we explain how the tasks of the shopping procedure are allocated between the application and its user, which new tasks arise for the human and why we decided on this level of automation. Furthermore, we describe the automation concept that underlies our application. For the main interaction strategy, which comprises a subset of the listed use-cases, we then present the flowchart and the corresponding prototypical implementation in Axure.

2.1. Use-cases

Before starting with the use-cases, we introduce the following terminology: There are two shopping lists. The *online shopping list* contains groceries that are bought automatically by our application, the *offline shopping list* contains groceries that need to be bought by the user themselves.

Overall use-case: Create a shopping list and buy the groceries
1. As a user I want to have a calendar where I can select meals for each day.
2. As a user I want to be able to add single items/groceries to the shopping list.
3. As a user I want to be able to go shopping on my own with the created list.
4. As a user I do not want the application to buy things I am allergic to or that I do not like.

Reaction application: Shops all required groceries three days before they are required (to assure that they are delivered in time), taking the current groceries in the fridge into account if the fridge is connected.

Use-case 1: Calendar to select/plan meals
As a user, when I click on a date, I can select a meal and specify how many people. The meal is either from the list of my favourite meals (see “User profile” below for more detail) or from an internet search for a certain recipe queried by me. When my plans for the week change, I may want to modify or remove already planned meals.

Reaction application: Calculates the required groceries and puts them on the respective shopping list (normally online shopping list; offline shopping list is only used if online one does not work, see “Error Use-Cases” below for more detail).

Use-case 2: Add/Remove groceries to/from the shopping list
As a user I want to add additional groceries, which are not related to selecting a meal, to the shopping list. Furthermore, I want to be able to remove groceries from the shopping list.
Reaction application: Shows both shopping lists and allows to add/remove groceries and their amount.

Use-case 3: Create a combined shopping list for own shopping
As a user I may want to go shopping on my own with the list generated by the application.
Reaction application: Allows the user to go to the shopping lists and click “Shopping on my own”. Then a combined list is created which includes required items from the online and offline shopping list. The list can be checked and when the shopping is finished all bought items are marked as shopped.

Use-case 4: User profile
As a user I want to have a personalized profile stating: allergies, what I do not like, replacements of groceries, favourite meals (custom recipes or from the internet search), preferred shop, credit card information, fridge connection, ...
Reaction application: Considers my wishes (replacements of groceries, what I do not like, allergies, ...) and preferences (e. g. shop) during the creation of the shopping list and the shopping itself. Favourite meals are stored locally and are hence always shown when it comes to selecting a meal.
Connection to the fridge
As the application I need to know which groceries are in the fridge in order to optimize the shopping. Therefore, the fridge has to be connected to the application.
Reaction application: Allows the user to add a fridge connection in the user profile. If there is no fridge connected, the existing food in the fridge is not accounted for when shopping automatically.

Error Use-cases
Not able to order the groceries in time for the selected day
As the application I might not be able to order the groceries for the selected day as it is too soon. Anyway, the user wants to cook the meal and not create a shopping list by their own.
Action application: Inform the user about this issue and ask whether to continue with the offline shopping list.
No credit card information
As the application I need to know where to get the money to pay for the ordered groceries.
Action application: Ask the user to add their credit card information or to continue with the offline shopping list. The user is not allowed to use the online shopping list without a valid credit card information.

No internet connection
As a user I want to get food even though my internet is broken. Favourite meals can still be planned but no groceries can be shopped automatically.
Action application: Notify the user about this event early enough and move the groceries that need to be bought from the online shopping list to the offline shopping list. This way, the user is able to go shopping on their own.
Groceries unavailable
As a user I want to get all groceries listed on the online shopping list. It might happen that some of the groceries are temporarily not available at the online store.
Action application: Notify the user about this event early enough and move the respective groceries from the online shopping list to the offline shopping list. This way, the user is able to buy them on their own.

2.2. Persona

https://thispersondoesnotexist.com/
Analyzing and Improving the Image Quality of StyleGAN (Dec 2019) - Karras et al. and Nvidia

Alice Müller

"I can see great realizations in my future because I am always dedicated and work hard to achieve all my goals and dreams."

"I wish I could have more time dedicated to myself in order to not be so overwhelmed all the time while dealing with my university activities."

Age	28
Gender	female
Family Situation	single
Job	Student (Molecular Medicine Master)
Residence	Ulm Eselsberg
Hobbies and Interests	Hiking Cooking Reading
Archetype	Stressed Student

Goals and Wishes
- Work in the field of research and be a successful medic
- Having time to also meet new researchers behind new developments and create a better networking
- Passing the study degree with outstanding accomplishments
- Find a way to better schedule her day in order to have more time for personal activities that are unrelated to the study at university

Frustrations / Challenges, Fears, Problems
- It was hard to get a place in the medicine program
- No regular income as student and part time jobs only bring less money
- Struggling with time to do all the activities of the day, nearly no time left for hobbies and interests
- Not being able to finish the master in time and getting doctor degree will also take a lot more of time
- Afraid of Corona as in case of an infection she will not be able to continue working/studying
- Deal with the challenges of the learning process in an online semester

Values
- Responsible person especially when it comes to other's care and well-being
- Integrity
- Socializing for good connections
- Likes the nature

Activities
- Studying and learning
- Taking care of the household
- Trying to find time for personal activities and trips to relax

Solutions (optional)
- Save time spent on weekly occurring activities in the household (shopping, cleaning, washing, …) for different activities (social, sports, relaxing, …)
- Better control of products and ingredients used in the week: generating less waste of food and also saving money

Figure 2.1.: Persona describing a possible target group

Figure 2.1 shows the description of a persona that could use our shopping application. Alice Müller is a student spending a lot of time with her studies as she expects a better career from good grades. Besides studying and weekly occurring household duties, there is not much free time left. With the shopping application, Alice can plan her daily meals and saves the time of

creating a shopping list and going shopping in the grocery store. Thus, this saved time can be used for spare time activities. When the fridge is connected to the application, Alice will also safe money and waste as groceries in the fridge are accounted for when planning the meals (e. g. suggesting meals that use left over groceries from other meals) and creating the shopping list (e. g. do not buy groceries that are in the fridge and are not required for any upcoming meals). Moreover, in the light of the coronavirus pandemic, Alice does not have to leave the house in order to get the groceries for cooking fresh meals and hence does not expose herself to the risk of getting infected at the supermarket.

2.3. Function Allocation

Function Allocation is “one of the first and most important problems in [hu]man-machine system design” (Chapanis, 1965). One has to decide which functions should be assigned to the automation and which to the human, or a combination of both.
In order to do so, we first have to make up our minds what tasks or functions a human has to accomplish when going shopping and which of these tasks should be automated. Table 2.1 lists all activities from deciding on what to eat over creating a shopping list and going to the supermarket to stowing away the bought groceries. Next to the task it is specified whether the function primarily remains with the human or is getting automated. Thereby, the user may take over tasks from the automation if they wish so or even must take over tasks in case of error. Whenever a task is not allocated to the user of the application, we account it to be automated. In case the automation is not achieved by the shopping application itself but by another service, it is stated explicitly. We are going into more detail how the tasks are shared between human and automation later in this section when we discuss the level of the automation.

Tasks/Functions	**(primarily) Allocation**
Search for recipes	automation
Decide on a meal	human
Calculate the groceries for the amount of people Write the groceries down on a list (and account for the groceries in the fridge)	automation (accounting for groceries only if fridge is connected) human if they want to
Go to the supermarket Search and collect the groceries Pay for the groceries Carry the groceries home	 normally automation human in case of error or if they want to for carrying groceries automation ≙ parcel service
Stock the groceries	human

Table 2.1.: Tasks in the human shopping process and their primarily allocation

Secondly, there are arising new tasks for the human due to the automation, like:

- creating a user profile and keeping it up to date.
- checking whether there are groceries that could not be ordered and if so go shopping by oneself.
- planing meals more in advance in order to take the full gain from the automation. Otherwise, only the creation of the shopping list is possible and going shopping itself has to be taken over by the user.
- double-checking the ingredients that are put on the shopping list. This is technically not necessary but the user may do this.

The resulting level of automation can be classified according to Parasuraman et al.'s (2000) taxonomy, where an automation level scale exists for each of the four types of information processing, namely information acquisition, information analysis, decision making and action implementation. The former two types can be referred to as information automation. As Parasuraman et al. (2000) do not provide a scale with levels of automation for the different types, we apply the level of automation taxonomy by Save and Feuerberg (2012), which utilises the same types of automation but with explicitly defined levels, to our shopping application. Thereby, we refine the function allocation previously presented in Table 2.1 without considering the cases in which the user takes over a task because they want to do so. For a general definition of the levels of automation which are declared in the following, we refer the reader to Save and Feuerberg (2012).

Information acquisition is the search for a recipe and the collection of its detailed information, such as the ingredients and how to prepare it. This step is on level A4 as the search for recipes is automated but still gives the human the opportunity to interact. On the one hand, there is a list of favourite meals that is always shown. The user can add and remove recipes from this list and the favourite meals are stored locally in the user profile. On the other hand, the user can search for recipes in the internet by typing in the name of a meal and querying the shopping application with this. Hence, the criteria for the information retrieval are predefined by the system but are visible to the user with the possibility of manipulation.

Information analysis comprises the calculation of the correct amount of groceries and the replacement of ingredients, which are both automated. The former is defined by the user setting the number of people for the meal and the latter is defined by the settings in the user profile. Thus, we do not achieve a complete automation and set the level of automation to B4.

Decision making is the selection of a meal which is primarily allocated to the human. The user has the full power to decide on which of the listed meals to cook on the selected day. Consequently, the actual meal proposals are not made by the user but as described in the information acquisition process, the human can influence the proposals. On top of that, the user may add custom recipes as favourites in the user profile. Therefore, the level of automation is between C2 and C3.

Action Execution includes the creation of the shopping list and going shopping itself. In nominal behaviour, the automation level is between D3 and D4. After the user has planned the meal or added items to the shopping list, the application automatically creates the shopping list and

orders the groceries a few days in advance. The execution can be interrupted by removing meals or groceries from the shopping list. Additionally, a planned meal can be modified, for example in terms of the amount of people or even the meal itself may be changed, when the groceries were not ordered yet. Optionally, the user can check the created shopping list and monitor whether the groceries are ordered or not. In an “error” behaviour, the user has to go shopping by themselves, while the automated creation of the shopping list still works. This reduces the level of automation to D2.

We chose a high level of information automation as the application will produce the same result as the human with less time effort. Moreover, human errors are excluded, especially in the information analysis. In order to not completely rule out the user from these processes, we did not choose the next higher level of automation (A5 and B5 respectively) which is full automation. In contrast to the information automation, the decision making and action execution are only supported by the automation. This gives the user the freedom to choose whatever meal or groceries they wish and no action sequences are initiated by the automation without the user being aware of them. Thereby, the human operator is kept in charge, which supports our underlying automation concept presented in the next section.

2.4. Automation Concept

The main goal of our shopping application is to support the human in the permanently occurring activity of creating a shopping list whilst taking into account the meals that are going to be cooked and going shopping in the grocery store. Hence, the human plays a central role in the design of our automation.
Sheridan (1995, 2000) discussed several alternative and partially contradicting meanings of human-centered automation. In the following, we go through all of the ten meanings and explain how we considered these aspects of human-centered automation in the design or why we did not consider them, thereby creating a consistent human-centered automation design:

1. Assign the tasks to better suited. ≙ Fitts (1951) ManAreBetterAt-MachinesAreBetterAt List
   We did not apply this concept of compensatory function allocation (Bye et al., 1999) as we view the human and the automation as supporting and thus complementary entities. Of course, one has to keep in mind that there are tasks in our application which cannot be automated under certain circumstances, for instance the automatic shopping when there is no valid credit card information available in the user profile.

2. Keep the human in the decision and control loop.
   This aspect was important to us as we did not want to take away the decision and control authority from the human by the automation. The application should rather support the human than making the decision for them on what to eat on which day. For most people

it would probably be annoying when they are “dictated” by some computer to eat and cook whatever it has planned for them. Furthermore, our application does not include safety-critical scenarios which require a fast actions only possible by the automation.

3. Maintain the human as the final authority over the automation.
   In the scope of the abilities of the automation, the human is always in charge. When the human is executing an action that will cause our application to not work in its full extend, the user will merely be informed and asked to confirm before continuing the action in order to assure awareness of these circumstances.

4. Make the human’s job easier, more enjoyable or more satisfying.
   This point reflects the main goal of the shopping application by replacing “job” with “daily life” in the description. When realising such easing, one has to take care to find the correct amount of mental workload for the human and thereby consider failure cases, as well. In case of total failure, the user has to create a list and go shopping just as they used to before the utilization of the application and they also have sufficient time to do so. Hence, we can focus on making the automation as comfortable as possible under normal behaviour and call on the user in error situations with which the application cannot cope. The role of workload for the user of our shopping application in nominal and failure conditions is discussed in more detail in Section 3.3.

5. Empower/Enhance the human to the greatest extend possible.
   We did not focus on this aspect explicitly as it is not important on its own (Sheridan, 1995). It may be part of other aspects in a diluted form in order to contribute to their successful realisation.

6. Generate trust in the automation.
   The challenge here is to create the right amount of trust. Like in a human-human interaction trust should be deserved. We discuss the concept of trust in the context of the shopping application in more depth in Section 3.2.

7. Give the human information about everything they might want to know.
   The critical question is: What information is important for the user and how to present this information without overwhelming? We had a look at this question from the perspective of situation awareness, as well, and hence refer to Section 3.4 for further details on the realisation. Broadly speaking, the user needs to be aware of the details of the meals they are planning and whether the groceries are already bought by the application or might even need to be bought by themselves.

8. Reduce human error and keep response variability to a minimum.
   We tried to reduce human error by setting the information acquisition and analysis, which entail the collection of the data from the recipe, the calculation of grocery amounts and the replacement of ingredients, to a high level of automation. Of course, the user has to enter

the correct amount of people and maintain the replacement settings in the user profile in order to get expected results. The variability is quite minimal for the shopping application as the flow of actions is determined by the application and most actions by the user are button presses. Merely the query for recipes in the internet is variable. The variability from adding groceries, custom recipes or replacements for instance is taken away by allowing variable input but requiring the user to finally select one of the suggestions by clicking/touching.

9. Human as a supervisor.
   As the shopping application does not have to be monitored continuously, this aspect does not apply. In our context the human is a decision maker which has to sporadically check whether the application is notifying them about a failure.

10. Best combination of human and automation, where best is defined by explicit system objectives.
    Due to the general complexity of quantifying the system objectives which makes it near impossible, we did not consider this aspect of human-centered automation.

Additionally, we included the concept of adaptive automation (Wickens et al., 2013), which allows for flexible and context-dependent function allocation. Figure 2.2 shows a general sketch of adaptive automation as a closed-loop feedback system. The task manager decides based on external conditions and the cognitive state of the human, which is inferred by the workload and the capacity to perform, whether a task is executed primarily by the automation or by the human. Thereby, the task manager can either be automation, human or a combination of both. In case the task manager is performed completely by the human, the concept is called adaptable automation rather than adaptive automation.

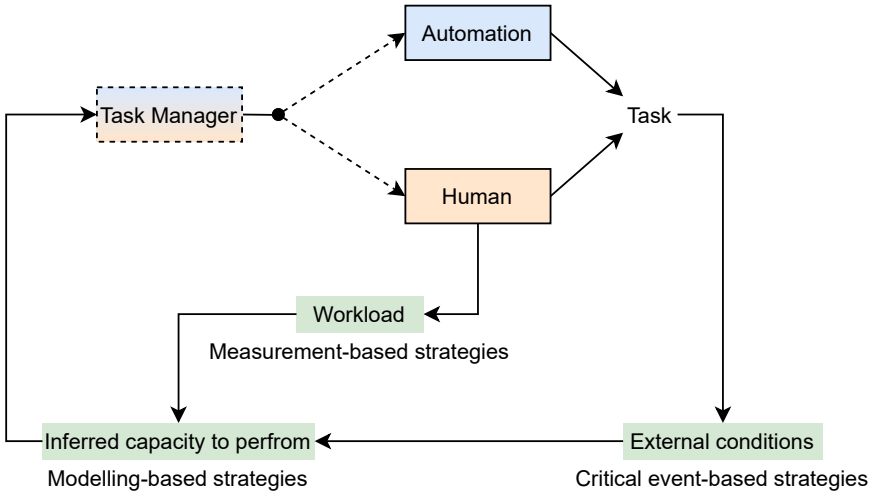


Figure 2.2.: Adaptive automation as a closed-loop feedback system
own illustration based on Wickens et al. (2013)

On the one hand, we have adaptive automation in our shopping application with the automation as the task manager. Due to external conditions leading to error use-cases, such as lacking a credit card information or planning a meal for a day that is too soon, the automation will not be able to perform the shopping in the grocery store. In such cases, the task is reallocated to the human by the automation. On the other hand, the user has the freedom to add and remove groceries to/from the shopping list allowing to create a list by themselves. Moreover, the user can also utilise the shopping list created by the application and go shopping with it by themselves even though the automation would be able to accomplish the automatic shopping. Usually, the automation is preferred over the human as a task manager (Wickens et al., 2013), but in our case adaptable automation is a valid option as there are no time- and safety-critical tasks and the user should be well able to estimate their daily workload and capacities.

In summary, the automation is a complementary and cooperative one that is centered around the human. Adaptive automation is exploited in order to handle error cases with the human as a fallback. Furthermore, the adaptable automation allows the user to take over tasks from the automation and accomplish these by themselves if they wish so.

2.5. Main Interaction Strategy

As we have identified several use-cases that make up our overall use-case, we have decided the planning of the meals to be our main interaction. After we have shown the flowcharts, we present our prototypical implementation of the main interaction in Axure. For the flowcharts of the use-cases which do not belonging to the main interaction, we refer to Appendix A.

2.5.1. Flowcharts

To a certain degree, the flowcharts match the use-cases presented above in Section 2.1. Hence, we use a similar structure in this section for easier reference.

Overall use-case: Create a shopping list and buy the groceries
This directly matches the use-case above and is sort of the main menu of the shopping application offering four different selection options as can be seen in Figure 2.3.

Use-case 1: Calendar to select/plan meals
The first selection option is the calendar. The flowchart of this use-case is depicted in Figure 2.4 and includes the error use-cases *no credit card information* and *not able to order groceries in time for the selected day*. The overlay to select the meal was modelled as a separate process in an own flowchart as it can be designed and implemented independently of the calendar. The

flowchart of this use-case only depicts what happens if the user taps/clicks on a calendar day without a planned meal. In case a meal is planned for the selected day, a modified meal overlay opens showing the details of the recipe and allowing for removing or changing the meal. For these actions, the same error cases as for the planning have to be considered.

Other use-cases

The flowcharts for the remaining three selection options and the error use-cases *no internet connection* and *groceries unavailable* can be found in Appendix A. They are not included here as they are not part of the main interaction.

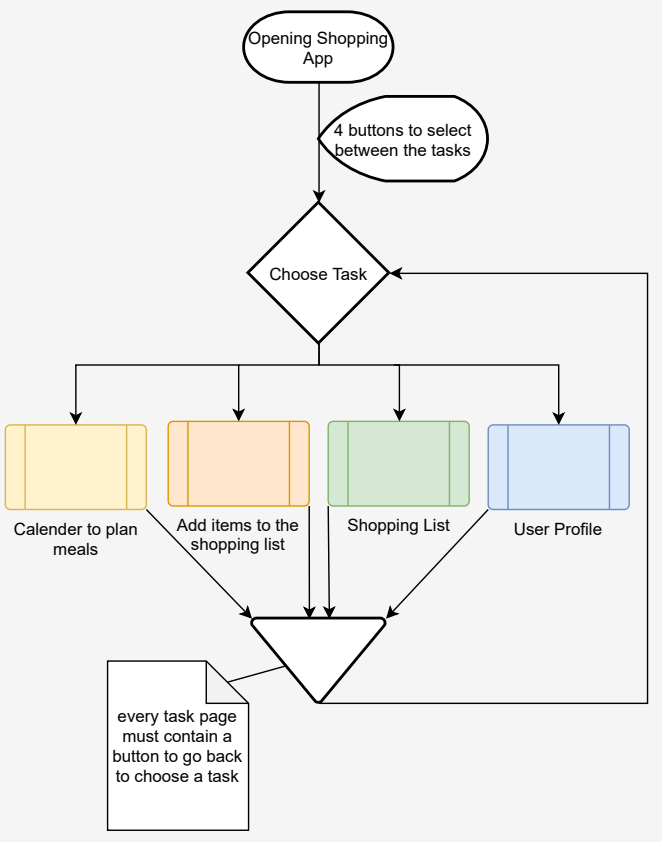
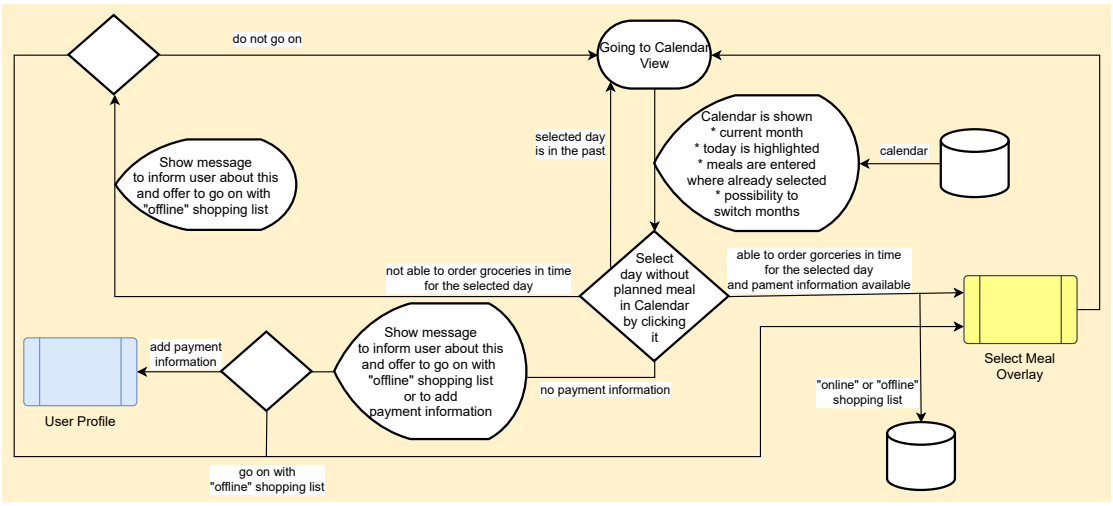
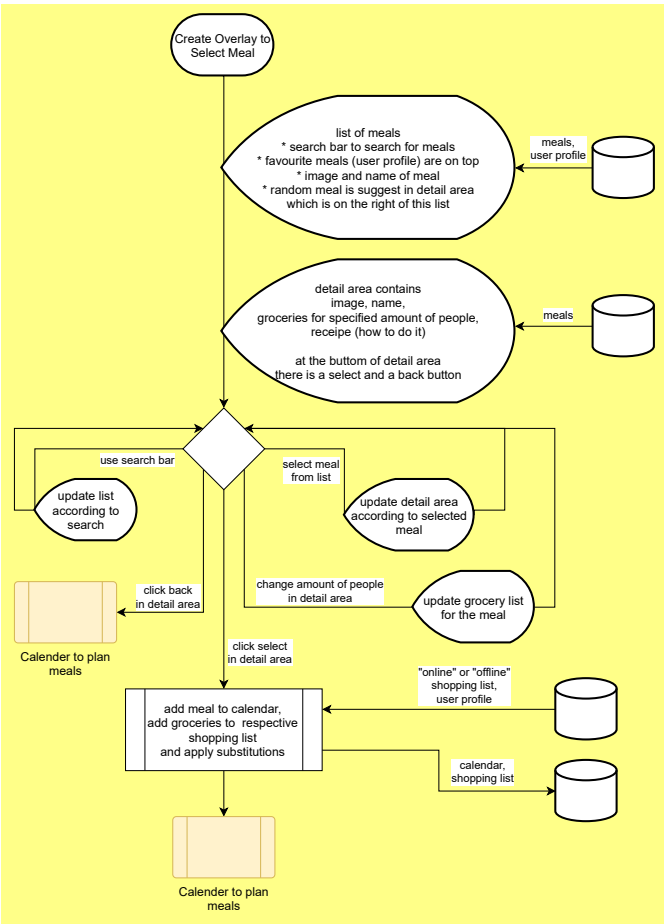


Figure 2.3.: Create a shopping list and buy the groceries use-case as a flowchart



(a) Calendar



(b) Select Meal Overlay

Figure 2.4.: Calendar to select/plan meals use-case as a flowchart

2.5.2. Axure

The resulting prototype implemented in Axure can be viewed and interacted with by opening the following link: https://f1zl3s.axshare.com. In the remainder of this section, we present the main interaction, which meets the flowcharts as described above, by showing screenshots from this prototype and explaining how the views link together in the interaction. Thereby, we give insight into our thoughts on the design of the prototype.

The first screen appearing when opening the shopping prototype is the main menu as depicted in Figure 2.5. This corresponds to the overall use-case *create a shopping list and buy the groceries*. The single menu points allow to access the respective sub-use-case. As only planning the meals is part of our main interaction, we continue with the content of this menu point and reach the view in Figure 2.6. Please, be patient whenever you go to this screen as the calendar is data persistent to a certain degree and hence a couple of variables need to be loaded and applied.

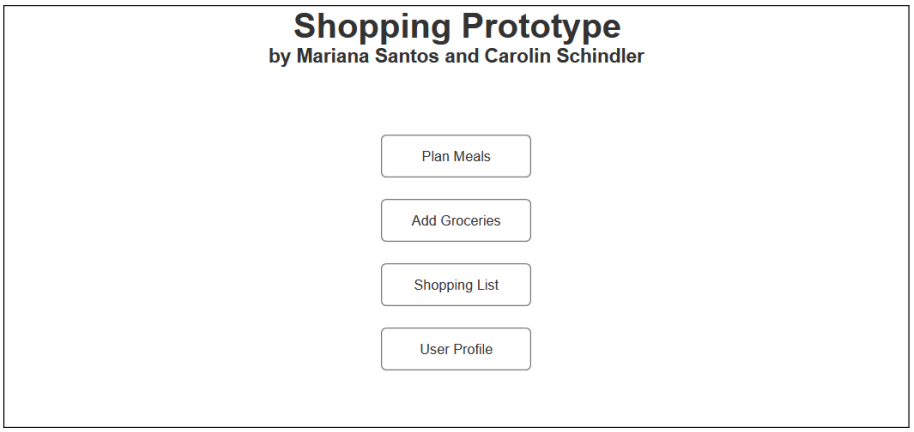


Figure 2.5.: Main menu / First screen of the shopping prototype

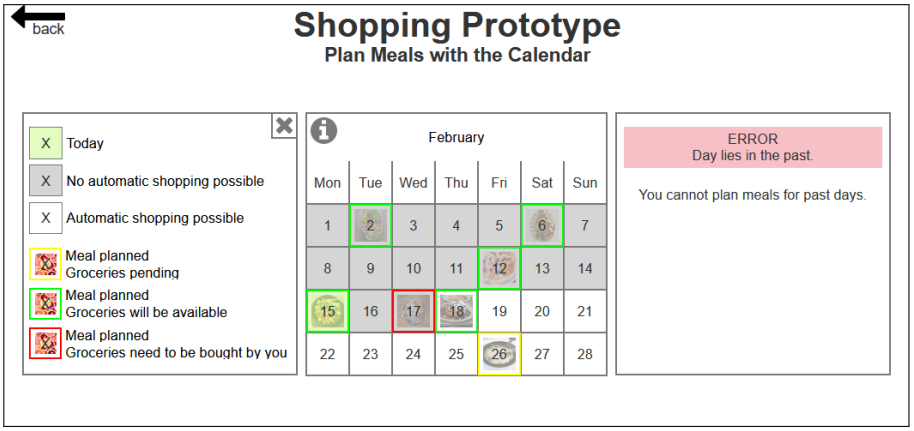
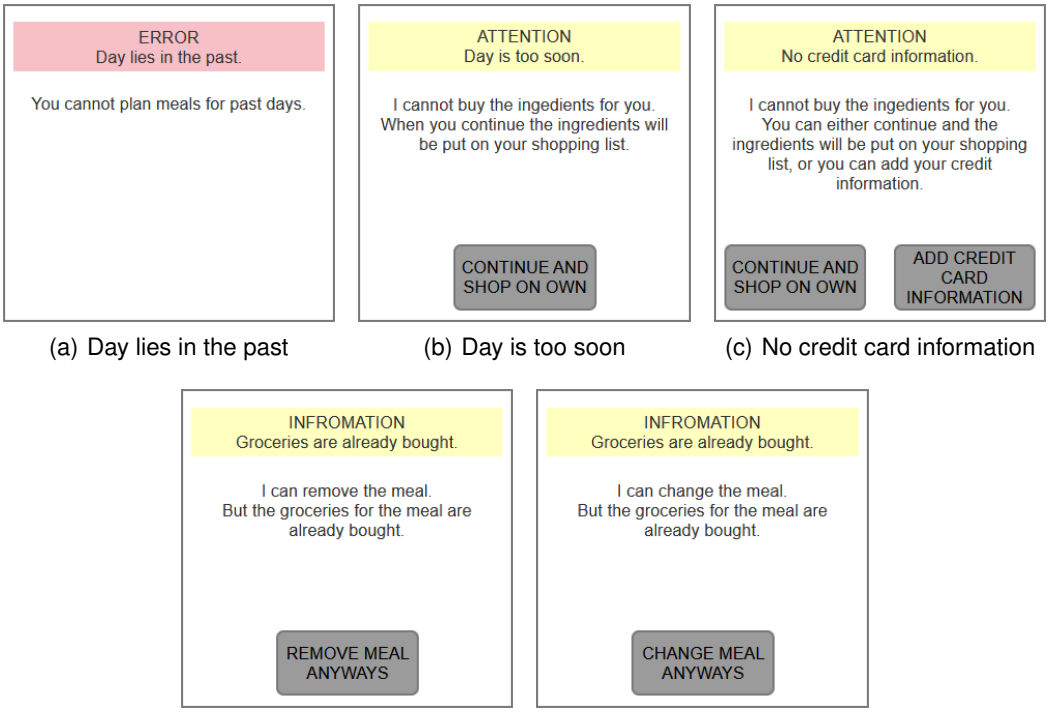


Figure 2.6.: Calendar of the shopping prototype

For simplicity, the prototype assumes that today's date is 15 February 2021. In the Axure implementation, you can only see the current month; in the actual product, however, the user should be able to switch between months. The calendar is colour-coded and when the user presses the information icon a legend appears left to the calendar, which can be closed again by the x-icon in the upper right corner. Since the colour-coding and the legend are part of our situation awareness concept and not of our interaction concept, they are not included in the flowchart and we refer the reader to Section 3.4 for further information. In Figure 2.6, a message box is shown to the right of the calendar. Initially, this area is white without any box visible. When the user selects a day in the calendar where no meal is planned, a message box might appear informing the user about an issue, as specified in the corresponding flowchart in Figure 2.4(a). All possible message boxes are displayed in Figure 2.7 and have a common style. The header of the box is colour-coded and contains two lines: The first line states the kind of message and the second line is a short description of the issue. The header is followed by a more in detail description of the reason for the appearance of the message. At the bottom there may be buttons allowing to perform actions depending on the issue.
Just as a quick reminder, the following scenarios are for selecting a day without a planned meal even though this is not explicitly stated. Message Box 2.7(a) appears when a day in the



(a) Day lies in the past (b) Day is too soon (c) No credit card information

(d) Groceries are already bought (remove meal) (e) Groceries are already bought (change meal)

Figure 2.7.: Message boxes in the shopping prototype

past is selected, indicating that no planning of meals is possible for these days. For today and greyed-out days in the future (15, 16, and 17 February), no automatic shopping is possible and hence Message Box 2.7(b) is shown informing the user that when they continue, they will have to shop on their own. Calendar days with a white background imply that automatic shopping is possible and if there is no credit card information available in the user profile, Message Box 2.7(c) becomes visible on selection of such a day. This box allows the user to continue shopping on their own or to add their credit card information. Of course, the adding of the missing information is the preferred action as then the automation can function in its full extent. When pressing the latter button in the box, the user reaches the user profile to which one can get to via the main menu, as well. The user profile is not implemented but has buttons to simulate the adding and removal of the credit card information. When a calendar day with a white background is selected and a credit card information is available, the user is directly navigated to the meal overlay to select a meal. This meal overlay is shown in Figure 2.8 and is reached when pressing the “continue and shop on own”-button in the message boxes, as well.

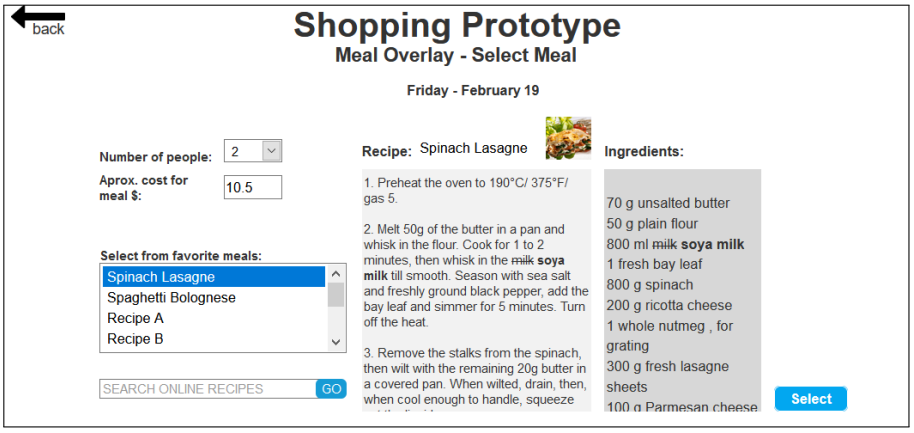
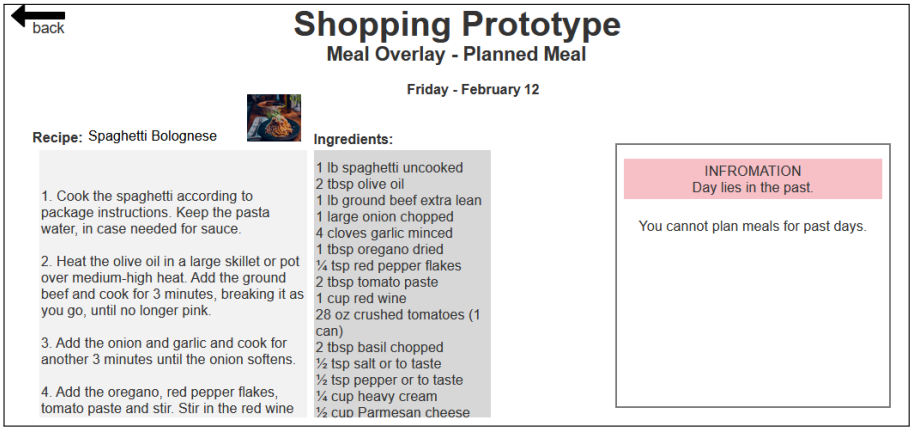


Figure 2.8.: Meal overlay to select meals in the shopping prototype

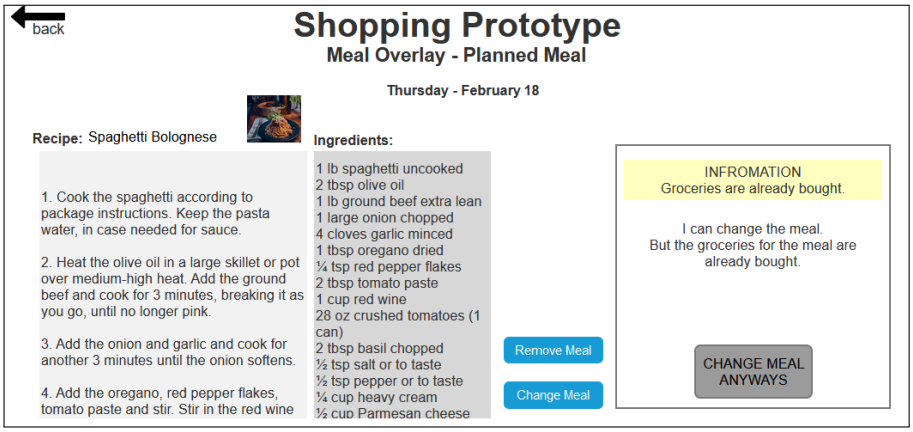
On this template it is possible to inform and visualise information about the recipes. The main interactions are: selecting the number of people and the desired recipe. This can be done in two different ways, selecting the meals from a previous saved list (favourite meals) or searching from an external website for a different meal. Saving recipes on the favourite meal list is not part of our main-case, thus is not here described but, it is our idea to allow that all recipes researched on the web can be saved for a future use. Being also possible to type by itself a family recipe, for example.

As shown in the Figure 2.8, after selecting a meal and the number of people, it is possible to see an estimated price for preparing the meal, this number will be pulled from the reports based on the previous purchases. Therefore, it will not be a definite price, only a prediction, as prices can vary daily (the final price could be visualised on the order list before purchase). This implementation in Axure was made, for simplicity, just showing the calculation of the product between number of people and meal price.

After selecting the recipe more information will be shown on the middle and left side, such as, the name of the recipe and a picture of the selected meal, at the top side. In addition, a list of how to cook the meal and a list of ingredients will appear. On Figure 2.8 it is possible to observe an example of an ingredient replacement. This action is shown here as a solution related with two theoretical concepts described on this paper, about the Ironies of Automation (described in Section 3.1) and trust in automated systems (explained on Section 3.2). Showing the information this way, it is easy and clear for the user to follow the replacements and modifications generated by the automation.
When the “select”-button is pressed, the currently displayed recipe is planned for the day, in this case 19 February, and the user gets back to the calendar. The state of the respective day is updated along with the colour-coding in the calendar. In the prototype, a default image is selected for the planned meal in the calendar; in the actual implementation, the indeed picture of the recipe should be used.



(a) Day lies in the past



(b) Day is today or lies in the future

Figure 2.9.: Meal overlay for planned meals in the shopping prototype

So far, we have illustrated the main interaction in case there is no meal planned for the selected day, such as shown in the flowchart in Figure 2.4. When a day with a planned meal is chosen, we decided to show the user the details of the recipe in an adapted meal overlay, depicted in Figure 2.9. With this, the user can see how to cook the recipe and which ingredients are required or share recipes they have cooked with others.
For a day that is in the past, the adapted meal overlay does not allow for any actions as can be seen in Figure 2.9(a). In order to not confuse the user by the missing buttons, Message Box 2.7(a) is shown with message type “information”. If the day is today or in the future, the meal overlay for planned meals shows the actions remove meal and change meal. Additionally, Figure 2.9(b) shows a message box on the right. Initially, this area is white and boxes may be shown when one of the actions is executed depending on the selected day. In case the groceries are already bought, Message Box 2.7(e) is shown for changing the meal and Message Box 2.7(d) for removing the meal. Thereby, we make the user aware of the fact that the groceries are already bought and hence will be left-over when the action is continued by pressing the button in the message box. After this check and when the meal is going to be changed, Message Boxes 2.7(b) (day is too soon) and 2.7(c) (no credit card information) are visible under the same conditions as for days without planned meals and functioning the same way.
When removing the meal successfully, the user gets back to the calendar and the state of the respective day, in this case 18 February, is getting updated along with the colour-coding. The “change meal”-button leads the user to the meal overlay to select a meal as in Figure 2.8 where changes can be performed and then applied by pressing the “select”-button.

Furthermore, it is noteworthy that the message boxes are consistently placed on the right side of the screens. Whenever a message box is replaced by another, there is a delay of 500ms after the old box is hidden and before the new box is shown. This way, there is a short period where the area for the message box is white and one can more easily recognize that the message box has changed.
Every view has a “back”-button in the upper left corner which does not statically lead to a predefined view but always to the actual previous view from which one came to the current view. This allows for a more intuitive navigation.

3. Theoretical Concepts in Relation to the Prototype

In the following, we briefly discuss the main theoretical concepts of automation in the context of our shopping application. Thereby, we especially focus on the main interaction strategy. In the beginning, we determine the ironies of automation that are relevant for the prototype. Subsequently, trust issues and the workload during the use of our application are analysed. Finally, we approach the concept of situation awareness, which plays an important role in the shopping application as trust, workload and identified ironies can improve from it.

3.1. Ironies of Automation

One important matter to take into consideration when talking about automated systems is the ironies involved in the process of automating this activities previously done by humans. Even though, Bainbridge's paper about ironies was published in 1983, it is still extremely relevant nowadays as one face the same obstacles related with the ironies identified by the author.
The purpose of automation is to take the human control related to decision making and activities that can be now done by automated systems. However, even years later it is not totally possible to completely take the human factor of this interaction. Also related to development of an automated system it is necessary to think about the ironies one can find while designing the prototype, after all it is a man-machine interaction.

According to Bainbridge (1983), there are seven ironies we must consider, which are:

1. The more advanced a control system is, the more crucial may be the contribution of the human operator.

2. Designer errors can be the main cause of operating problems.

3. Even if the designer tries to eliminate the operator there are some tasks the designer has no idea how to operate.

4. With automated systems the operator needs to be more rather than less skilled.

5. An operator in a manual take-over will only have a minimum of time so he cannot access long-term knowledge. The operator has to react quickly, but he only has a minimum information.

6. If the system works stable for a long time the operator will not be adequate vigilant

7. For well-defined problems automated system find appropriate solutions quickly. A human cannot check this in time, so he has to use intuition or some meta-level. To avoid proceedings computers were applied.

For the development of this prototype irony 2 is the most relevant one, though, we also need to pay attention to ironies number 3, 6 and 7. The ironies that are relevant or not for this prototype can be observed in more details in the Table 3.1.
Irony number 2 says that, designer errors can be the main cause of problems. Related to the development and main ideas about a prototype it is important to cover every aspect and make it as simple as possible. Developers can design an interface that in their belief it is intuitive and easy, but it can be complicate for others. With this in mind, interface needs to be clean and easy to navigate in order to prevent mistakes or errors from happening. Thinking about our prototype it could make it easier if at the beginning a tutorial was shown to explain all the functions of the app and to show everything that is possible to do and how to do it.
Besides the layout being difficult to navigate it is also possible to have a programming issue if not well developed. For this prototype could be an issue if the we face problems with adding the ingredients to the shopping list; or not excluding or substituting ingredients that were in the not wanted/allergic list, for example. Situations like this can be one of the reasons that will cause discontentment with the app leading to trust issues, which will be covered on another topic.
This irony should not be taken lightly because layout and functionality of the system it will be one the first contact that the user will have with the app. Even though the purpose of the automation could make a huge difference in the user life one will not make use of it if it is extremely complicated to navigate, or continuously shows errors. This instead of facilitating someone's life could cause overall more workload.

Considering irony number 3, in this case it is not a problem that the designer did not know how to automate. However, we had to take into consideration the fact that having technical issues (which means having no connection with the internet in this case) the system will not be able to deliver everything that was promised. Two situations could occur here in case of connection failure. Firstly, the user would be limited to choose the meals only from the favourite list, in case the intention was to try to find a new/different recipe on the web to prepare for the week. Secondly, it would not be possible to complete the purchase of the list that was generated after selecting the meals.
This irony is not extremely concerning in our case because even though, it is something we

Irony	Relevant	Why?	Solutions
1	No	It is not required to have any previous or further knowledge/skills of the automated tasks in order to use it. The system is not too advanced.	-
2	Yes	If the app it is not intuitive or easy to navigate can lead to misinformation and errors.	Clean and intuitive applications. (situation-awareness is also involved here).
3	Yes	In case of not having internet the system cannot finalise the operation.	Create an alternative where the operator has the option of completing the task themselves. (Notify the user if it has to take over → adaptive automation.)
4	No	- Tasks automated do not require a complex set of skills in order to understand/use the automation. - Skill-degradation: shopping is a soft skill, so no degradation expected.	-
5	No	In case of a failure the human is not need to take over (immediately), this not being a dangerous task. Even so, it would have plenty of time to finalize the task (shopping or go to buy themselves).	-
6	Yes	Although, it does not require monitoring, rely on automation to execute this activity could generate overtrust and as consequence user could not be attentive to information displayed (e.g. connection failure connection and having to go shopping on own), which could lead to an error.	- Make notification visible and maybe audible. (e.g. popup that does not go away until button is pressed) - "Finalise process" button will be disabled in case of no internet connection. - Colour code at the calendar.
7	Yes	Ingredients substitution will be made very fast by the system. In case, user wants to check those information it will not be able to follow them in real time (as it happens).	When selecting the recipe the automated system will verify if there is any item to delete/substitute and those will be shown at the ingredients box. Previous ingredients (before change) will be shown with a strike-through line and new ingredients will be in bold (e.g. ~~Onion~~ **Garlic**).

Table 3.1.: Analyses of Ironies relevance for development the Prototype.

need to notice that can be an issue, it is manageable in case of error. The system will inform the user when there is no internet connection and it will be possible to choose, according with the user necessities, if they want to wait for the connection regarding they still have enough time available to finalise the process (shopping) later or to go themselves to the supermarket and buy the groceries.
Equally as irony number 3, irony 6 it is not extremely important for our prototype but it needs to be taken into consideration. It would be more relevant in the case of needing to monitor the system in the human needs to interact and make a decision in case of failure. However, for our prototype after the system working well for a long time the user can be not very attentive, causing it to miss an error or messages, thus the purpose of the app might not be achieved.

Therefore, the only malfunction would be in case of failure to connect to the internet, which leads to not actually buying the groceries. If the user do not notice the internet connection issues in time it might not have options available. To prevent this from happening some measures can be done, e.g. a pop-up appearing to inform that there is no internet connection and user will be required to click to confirm in order for the message to disappear. Also, when all meals are finally selected for the week and user wants to proceed with the purchase the “finalise purchase” will be disabled and the only option will be “buy groceries myself”. In case of error during finalising purchase the user will receive a message informing the impossibility of finishing the process (e.g. bank problems, wrong card details, no money). A last alternative for the automated system failure is a visual way (colour-coding) to inform the user of this errors. One can check that in case of not completing the purchase the days in the calendar will show a pattern of colours:

- Red for the days that is not possible to buy the groceries, not sufficient time;
- Yellow for the pending transactions;
- Green for completed orders.

Another expected activities of this prototype is to be responsible to “check” the *"dislikes"* and *"allergies"* list where the user can inform ingredients they would not like to use in the recipes or that they are allergic to. Two actions can be performed in this situation; delete the selected ingredient or to substitute it (e.g. if one does not like onions but would like to always replace it for garlic or to simply delete the item onion from "buy list"). Once adding this information to the list the automation will be responsible to execute the necessary action (delete or replace the item). In case of failure it is not a critical error, even if the user have severe allergy because the worst case scenario is for the app to not delete/replace the item, which would lead to a unnecessary cost (buying the wrong item).
Consequently, it will not be a critical error. Even though, the user receives the item and it will not use it, this will not be the action he was expecting from the automation. Therefore, this is why irony 7 will be relevant for this prototype. The user, for example, in one side can be sceptical of the app’s ability to produce an accurate list, deleting or replacing the unwanted ingredients

correctly. On the other hand, this replacements are done extremely fast by the system and the user can not follow all the changes as it happens.
Basically, the challenge here is in how to make the user feel more secure about this changes? One solution we could present to solve this problem is for the user to have a way to check these information. After selecting the meal it is possible to see all the necessary ingredients and the recipe for the selected meal. This way the user can see if there is any ingredient changes. Also, in case of distrust (this concept will be elaborated in the following section about Trust, 3.2) it is possible to double check all the ingredients to find a possible error (not replacing nor deleting an specific item) of the automation. That verification will be possible because instead of just deleting or replacing it the groceries will still be shown on the list in a way that is possible to visualise all the modifications:

- the deleted items will be presented with the strikethrough format;
- and if the item was replaced the new product will be shown in bold (e.g. replacing onions for garlic it will be shown as follow, "~~Onion~~ **Garlic**" (this interaction can be observed in Figure 3.1).

Thus, it will be visibly easy for the user to find the modifications generated by the application, according with the information previously presented by the human in the unwanted/allergies list.

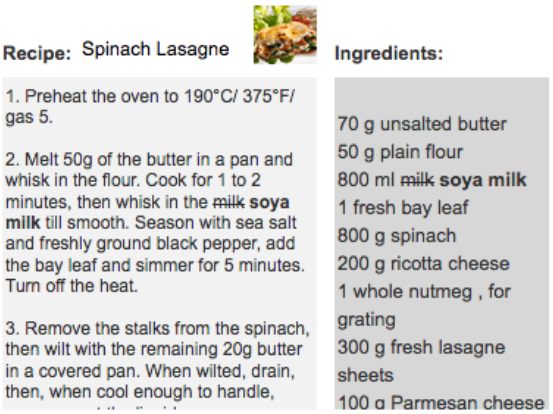


Figure 3.1.: Demonstrates how an ingredient replacement is shown on the prototype.

3.2. Trust in Automation

Trust is an important subjected to consider when developing an automated system. The point of this human-machine interaction is to accomplish in a successful way activities now performed

by machines, also to accomplish it in a better way that humans would do it (e.g. faster, more precisely).
Lee and See (2004), define trust as an individual action towards an agent in order to achieve a goal. Most of the time this moment of trust is characterized for the individual seeking help in moments of uncertainty and vulnerability. Considering that, overall this is a weak moment for the individual, the trust on this agent can be built or destroyed depending on the outcome. For this point of view "the agent" could be another person or an automation, as long as, it is interacting actively on behalf of this individual. Therefore, this interaction between human-machine can also be mediated by the level of trust.

For our prototype Trust is extremely important as it can define how the user will use the automation. This interaction is based on trust. Therefore, if trust is a missing piece in this equation there will be no motive to use the automation at all. However, Lee and See (2004) mentions that trust can guide how the user will act towards the automation but it does not completely determine the human reliance on automation. What we can understand using the theory of trust is that trust will increase or decrease in relation of how the automation is functioning. In on hand, the correct functioning in delivering what it was proposed to do will increase the level of trust, on the other hand this level will decrease by incorrect functioning of the prototype.
In order to understand some aspects of trust involving our prototype we need to explain the concept of mistrust and overtrust. "Inappropriate reliance associated with misuse and disuse depends, in part, on how well trust matches the true capabilities of the automation." (Lee and See, 2004). It is important to have the appropriate trust in order to avoid disuse and misuse of the automated system. To achieve this "appropriate trust" Lee and See (2004) mentioned calibration factor that is described by him as a correlation between the user's trust in the automation and the automation's efficiency. Thus, overtrust is a poor calibration that will lead to an exceed trust the user have on the capabilities of the automation. While mistrust cause for the user to not believe the automation will be able to execute the proposed task. Both concepts can be an issue related to our prototype. We considered that ranking the level of trust of our prototype would not be extremely helpful. However, we collected some points (related to our main-case) and it is possible to observe in Table 3.2 how we defined which tasks would require more attention regarding user's possible reaction to errors (high level of trust) and which ones would not have a huge impact on the individual's trust (low level of trust).

As briefly mentioned before in the section 3.1, about Ironies, the replacement of ingredients can bring some attention to the conceptual theory about trust. The automation task in this situation would be to go over the ingredients on the selected meal and match with the *"allergies list*" and *"unwanted list"* (if there is any information provided), being then responsible to exclude or replace them. Some skeptical user could not rely on the generated final list and consequently, not trust or not use the automation. To fix that possible issue our idea is to show both deleted and replaced items. This process was thoroughly explained when talking about irony number 7 (section 3.1 and can be observed in Figure 3.1) in case it is necessary to better comprehend

Low level of Trust required	- favorite meals - adding groceries on your own
High level of Trust required	- credit card information - replacement of groceries (especially allergies) - quantity of ingredients (calculated correctly) - prototype finalisation of order

Table 3.2.: Level of trust required for main use-case tasks.

the notion. Hence that with this action the user will be able to see automatically all the changes made by the system and check if some ingredient change was missed.
Informing the user about this modifications by the system it will keep it in the loop of decisions and will help in "building" the trust and avoiding mistrust. Also, it will facilitate the human to perceive its own mistakes as well. For example, when checking the ingredients to know if the automation responded correctly,it might realise that a particular ingredient was not crossed not because the automation did not worked but because the user forgot to provide this extra information. Mistrust issue needs to be addressed because if the human keeps double checking all the actions made by the automation it will cause more workload (this subject will be better discussed in a following section). Overtrust is also a big issue related with this prototype. If the human directs an exceeded trust in the automation capability it might lead to losing some alerts, as it is usual for humans to lose attention or focus when doing more than one activity at the same time. Consequently, it could close the box alert that recommends it to buy the groceries by its own because there is no internet connection or no sufficient time for the groceries to be delivered.

The automation is programmed to be 100 percent reliable, but it is also dependent of some external conditions, like failure in connecting with the internet. Those are not problems related to the automation but the discontent will be directed to it. We thought in some ways to compensate this malfunctioning in order to not generate trust problems towards the automation. All information about the processes done buy the automation will be shown as messages to the user. In the case of not having credit card information, for example, the user will be required to click in inform this missing data or agree on going to buy the groceries by itself before selecting the recipes. The warnings have options and need confirmation so it would be less likely that information will be missed. To finalise, we did not take it lightly the impact of this concept for a successful interaction between human and automation. Which is why the automation is very interactive and transparent with the decisions made, even the ones the human cannot follow in the moment that occurs. It is necessary to keep the human in the loop decision in order to create a calibrated level of trust towards the automation.

3.3. Workload and Automation

One of the point of an automation system, if not the main purpose, is to decrease or eliminate the human workload. In our prototype the idea is to unburden the user from some daily tasks, being that physical or mental workload. What we need to take into consideration to this point is that what if the automated system fail? This, in many occasions could lead to extra workload and create stressful situations.
Parasuraman and Wickens (2008), mentioned that we can find parallels between workload and situation awareness, as both are important to develop actions that will be linked to behaviour and performance. The Cambridge dictionary defines workload as "the amount of work to be done, especially by a particular person or machine in a period of time". We need to consider, as mentioned before, the mental workload (mental resources) required to execute an activity.
According with Parasuraman and Wickens (2008), excessive workload can lead to errors. The user can be stressed or not paying much attention caused by the high number of actions required from them to execute some tasks. However, while utilizing the automated system will help with the decrease of the workload, the slightly change (error) produced by the app could increase to a great extend the workload.

Our prototype will help the user to decrease or eliminate some workload experienced in daily life, it will help with the action of writing all the ingredients to buy, this action can takes time if one is planning meals for the whole week. The human can also not want to write a list in order to save time, and in this case could forget some ingredients, leading to more workload, if it is necessary to go to the supermarket only to buy the missing ingredients. While using the shopping app the individual will not have to worry about this. Looking through a perspective that it is not easy to program meals for a whole week, which requires some mental workload to plan and look for the recipes, here is another point where the prototype will help. It will not be required of the user to look into a lot of different sources for recipes, like recipe books, internet or even some family recipes, because all this information could be found at the same place along with some suggested meals. Moreover, in case of doubling the recipe this would require for the individual to double all the ingredients by itself. Using the "shopping app" will facilitate this action as all the ingredients will be automatically calculated in order to inform the right number of items to buy. In addition, the individual when using this prototype it will not have to worry about going to the supermarket to buy groceries, which will imply in more time being saved when thinking about the time spent on the journey, at the supermarket itself (looking for groceries and on the check out line). In this situation it would be possible to also eliminate the physical workload of carrying all the groceries bought through the supermarket, the parking lot to the car, or in case of not having a car, it would be necessary to carry on a public transport or by foot.
Although, the automated system will help with the workload, at the beginning it could present some extra workload, such as, filling the profile information (personal information, credit card) and keeping the list of ingredients not to buy (products not liked or that cause allergies) up to date. However, as mentioned before, this information will be required to be entered only once or

whenever might have a data change (e.g. credit card number). Another possible extra workload created because of the prototype is double checking all the recipes (low trust) to verify if the items were replaced (according with allergy and unwanted list) or if the number of items are correct, in case of doubling some recipes.

We also would like to emphasize is that in case of errors unexpected workload would be caused. In case of malfunctioning (e.g. no internet connection), it will require the person to go shopping by its own. Even though, this action by itself could be thought to not being to prejudicial as it is something the individual was used to do it before the automation, it is necessary to remind that the time usually spent on this activity it would probably being used for something else. After trusting that the prototype would always work the user might be using this "created" spared time for another activity, which could cause stress as it will have to choose between this two actions or the individual could not have time available to buy the groceries.
It is also possible that a malfunction could cause losing partial or total data information, e.g. losing the groceries list, bank information, allergic details. Besides causing stress it could generate extra mental workload, as a result of having to understand what happened and to go through the data to check what was lost. This would increase workload and further could also lead to the decrease of level of trust in the automation. The previous tasks mentioned and analysed her can be observed altogether at the Figure 3.3 .

Level of workload generated by automation	Required Tasks	Physical workload	Mental workload
Less Workload	- Do not have to spend more time planning and searching for the meals.		X
	- Do not have to calculate the ingredients for each meal in case of inviting more people to eat.		X
	- Do not have to write which ingredient on a list.	X	
	- Do not have to carry the groceries from supermarket on the way home.	X	
	- Do not have to go shopping on its own.	X	
	- Do not need to spend time on trajectory to supermarket and back home. As well as searching for groceries and queuing at the checkout.	X	X
Extra Workload	- User profile - Fill information (it will be required only once or in case of update).	X	
	- Double checking information such as items replacements or if the number of items were correctly modified after adding more people (caused by low trust)		X
Unexpected workload (In case of error → No internet connection)	- No programmed time to go shopping (shopping time replaced by some other activity) → stress		X
	In case of application get an error leading to total or partial loss of information, e.g. losing the groceries list, bank information, allergic details: - make up for information loss, understand what is going on → stress (this would also highly decrease the trust). - double checking all information in case of partial loss or including everything again in case of having all information deleted.	X	X

Table 3.3.: Different levels of workload generated by automation.

3.4. Situation Awareness and Automation

As mentioned earlier, situation awareness is an important theoretical concept in our prototype. In general, situation awareness (Endsley and Jones, 2016) is achieved in three levels, where higher levels cannot be gained without obtaining the preceding levels:

- Level 1 – Perception
  One needs to be aware of the information in the environment and its reliability. This means for the designer that the system should present all the required information in an easily perceivable way to the user. Thereby, we have to keep in mind that there may be lots of sources of information competing for attention.

- Level 2 – Comprehension
  One needs to understand the meaning of the perceived information. To this end, a good mental model is required, which develops through experience.

- Level 3 – Projection
  One is able to predict the future state of the information and hence of the system. This allows the user to make proactive and fast decisions.

Our main focus in terms of situation awareness lies on the state information of the single days such as: Is there a meal planned for the day or not? What is the state of the groceries for a planned meal, e. g. are they already bought or is shopping by the user required? In order to make these information easily available to the user, we decided to colour-code the calendar as can be seen in Figure 3.2. Hence, level 1 situation awareness is achieved by perceiving the colours of each day containing information about its state. For a better understanding of the meaning of the colours, the user can open a legend explaining the colour-coding by clicking/touching the information icon. This way, we want to support level 2 situation awareness. The background

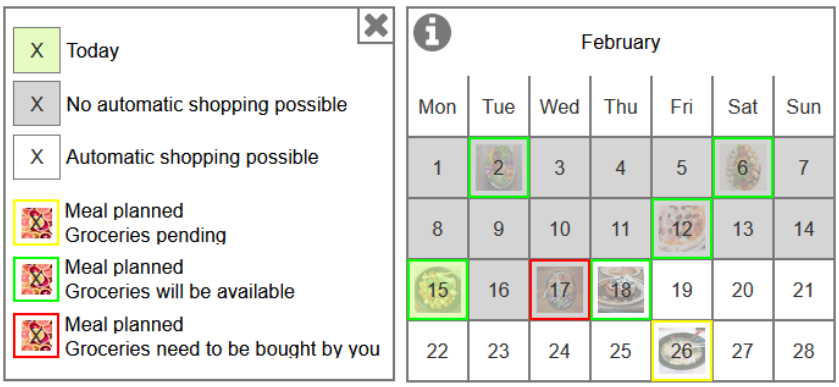


Figure 3.2.: Colour-coding of the calendar for situation awareness

colour of the day shows whether automatic shopping by the application is possible or not and the current day is highlighted as well. When a meals is planned for a day, the respective image from the planned recipe is shown behind the number of the day and the border gives further information on the state of the groceries: Yellow means that the groceries for this meal are pending, they are not bought yet but the automation is taking care of it. Green indicates that the groceries are ordered and thus on their way or are already at one's home (either shopped by the automation or by the user). Red implies that the groceries are not bought yet and the automation will not be able to do so, calling the user for action. The colours were selected this way as they are intuitively understandable from the traffic light rating system and other common status indicators. Level 3 situation awareness is the prediction of what will happen with the colours in the calendar on the next day or after executing an action.

Other aspects of the prototype contributing to the situation awareness are the message boxes, which were discussed in Section 2.5.2. They have a colour coding according to the traffic light rating system, as well. Furthermore, both meal overlay types always show the selected date from the calendar in the top and an estimate for the price of the groceries required to cook the displayed recipe. The former is a help to not loose track of the selected day in case of distraction and the latter is important as this cost can be expected to be booked from one's bank account. In addition, the meal overlay to select a meal indicates the replacement of groceries. This information is not necessarily relevant for the user but helps detecting wrong replacement settings in the user profile. Moreover, it may support building trust in the application as the user can verify the correct execution of this task by the automation.
Beyond the prototypical main interaction, situation awareness plays a role in the error use-cases *no internet connection* and *groceries unavailable*. The user needs to be aware of the notification and understand that they now have to buy the groceries on their own, otherwise they will not be able to cook the planned recipe. This directly links to the presented colour-coding in the calendar. Another important aspect is that groceries which are already at home and required for an upcoming meal should not be used for unscheduled food. Therefore, the user needs to keep track of the purpose of the groceries. An intelligent fridge connection might be able to help, warning the user when they take "reserved" groceries from the fridge.

4. Discussion

The relevant concepts used for developing this prototype are: situation awareness, workload, ironies and trust. Situation awareness being the most important one considering that most interface is based on the user identifying and assessing information through the colour-coding implementation at the calendar and boxes notes. The concept of trust is also significant for this project as it has a huge impact on the user decisions and it could dictate whether the app it will be reliable or not for its use. As mentioned, the workload before and after are a little more complicate to measure as we have mental workload involved, and we also observe the problem of generating extra workload in case of system failure. Of the seven ironies discussed on the section 3.1 four of them are relevant for our prototype (as illustrated at Table 3.1), even if some would be only relevant in case of an error we could not disregard that they are important.

The main purpose of the automation is to help the person using it to have more free time. All the tasks completed by the automation does not require excessive workload but it will for sure improve the daily life tasks. Taking our persona as example, Alice is a young student worried about her masters requirements along with her prospects for the future. The shopping app will release her for the worries of having to make lists and go to the supermarket, taking away some physical and mental workload and giving her more time to do more important activities. It might seems like not so much improvement but having more time for herself and one less thing to worry about it is extremely helpful. Think about the point of view of someone that is cooking for a big family, all this plans of having different meals every day, calculate all the ingredients, write them down and have to go shopping. All this tasks would take a long time from a person that is already in charge of a lot, so imagine how this automation could help not only people like Alice. Such as parents (they will have more time to spend with the kids), elderly people as sometimes locomotion and carrying weight can be an issue. For some other targets maybe some new measures must be implemented, for example for elderly, a tutorial on how to use the app and the possibility of having bigger fonts to choose. Or the possibility to pay by transfer or when the groceries arrive, in case the user does not have a credit card.

During this process we observed that this automation can have its limitations, no automated system is invulnerable to errors but our prototype main action rely on a external factor, internet connection. If there is no connection, it is possible to interact with the system, looking through selected meals, look at the list created, delete some information and even select a new meal, considering one have created a list of favorite meals. However, it will be impossible to actually buy

the items. Taking this into consideration some measures was designed to help in this situation, popup messages letting the user know that is no connection, for example. And the possibility to import the list in case there is no time to wait and it is necessary to buy the groceries by its own. The system will also keep the person informed if it is possible to wait for the connection issues to be fixed and to finalise the order later.
Another element to think of is the possibility of a fridge and storage connection, it might sound futuristic, but this integration could help the automation to manage all the ingredients available in order to minimise waste. Without this connection it would require organizational skills and for the user to maintain the app updated with the not used food, as the person could also add items to the final list that are not required for the selected meals. Furthermore, most of all interactions developed so far are based in a colour-code system. We did not took into consideration, though, the possibility of the user being colour blind. That would be definitely something to think about for the next steps implementations, if it would be possible to change something or add extra features to help in this matter.

When we started the development of the prototype, the practical sessions introducing the use-cases, persona and flowcharts were of good help to get a bigger and more complete picture of the prototype. Since we have identified several use-cases, it was at first a bit challenging to discuss the theoretical concepts until we decided on a main interaction that only includes a subset of the use-cases on which we could focus then. With the flowcharts, we have identified the calendar and the meal overlay as two components that have to be implemented in Axure. Hence, the latter was created by Mariana, while the former, the message boxes and the interactions between the pages with persistence were realised by Carolin. As we both were not familiar with Axure, the first steps were quite hard even though we participated in the Axure exercise in the lecture, but over time one learned more and more about how to prototype with Axure. Moreover, it was quite helpful that we have created an Axure “Team Project” and not used local Axure files. This enabled us to simultaneously work on the prototype in a single file and the complete history of updates in the Axure Cloud allows to see what was changed by whom and may include a comment with further details by the author of the change. In case of fatal changes, not updated ones can be easily reverted and if they were already updated to the “Team Project”, one can go back to an earlier working state of the prototype. In order to work on the report at the same time and not make the efforts of installing LaTeX locally, we utilized the overleaf[1] platform. The content of the single sections in the report were discussed together. Carolin wrote the Description of the Prototype and added Appendix A as the flowcharts were created by her. In Section 2.5.2, each of the authors presented their own contributions to the prototype in the Axure. Mariana wrote down the Introduction, the Theoretical Concepts in Relation to the Prototype, the Discussion and the Conclusion. Section 3.4 about situation awareness was written by Carolin as this concept was mainly implemented in the calendar.

[1] https://www.overleaf.com/

5. Conclusion

The shopping app was think of to facilitate in a daily activities of not only our target group but everyone else that want to cut some extra activities from the list of things to do. It will help those to plan the meals for a selected period and actually buy all the needed items, eliminating some workload and freeing some time of the agenda.

In this report, we illustrated how this application would work, its functionalities (use-cases), function allocation and we described in more details about the chosen main interaction strategy, selection of meals. To better define and explain the interactions happening we used relevant theoretical concepts about situation awareness, trust in automation, workload and automation and the ironies of automation. Which helped to guide us through the decisions and development of the prototype, such as showing us when to think about alternatives to fix/prevent errors.

We believe this application could help a large number of people (not only directed to our selected target audience) to enjoy having some tasks cleared from their daily activities. However, it is clear we might have to adapt some of the interactions or think of a different layout in order to make it easier for different targets, for example, a bigger font for people with vision problems (elder people). Also, the colour-coding system (used for situation awareness) it is utterly important as well to facilitate the human-machine interaction in case of an older people using it.

Although it was challenging to develop this work we are very pleased with all the results and aware that this would be only the beginning. A lot more work would be involved to delivery a functional application at the end. We also believe that based on the theory used we were able to think of all the problems that are involved on the development of this prototype and we worked on a better way to help minimise or eliminate possible errors.

A. Additional Flowcharts

The following flowcharts depict the interaction in our remaining use-cases that are not part of the main interaction and hence were not implemented in Axure. The structure of this section matches the presentation of the use-cases in Section 2.1 for easier reference.

Use-case 2: Add/Remove groceries to/from the shopping list
The second option in the main menu allows for adding specific groceries to the shopping list as shown in the flowchart in Figure A.1. The *no credit card information* use-case is considered here, as well. The removing of groceries is not depicted.

Use-case 3: Create a combined shopping list for own shopping
The third selection in the main menu enables the user to go shopping on their own with the list created by the shopping application. The process is depicted in Figure A.2 as a flowchart.

Use-case 4: User profile
The last options allows to edit the user profile. The flowchart in Figure A.3 contains the entries in the user profile including the fridge connection. How single entries can be edited is not specified explicitly.

Error use-case: Groceries cannot be ordered
The error use-cases *no internet connection* and *groceries unavailable* could not be incorporated into any of the other flowcharts. Figure A.4 shows possible outcomes of the application shopping groceries on its own when there is no more attempt allowed in order to get the groceries in time.

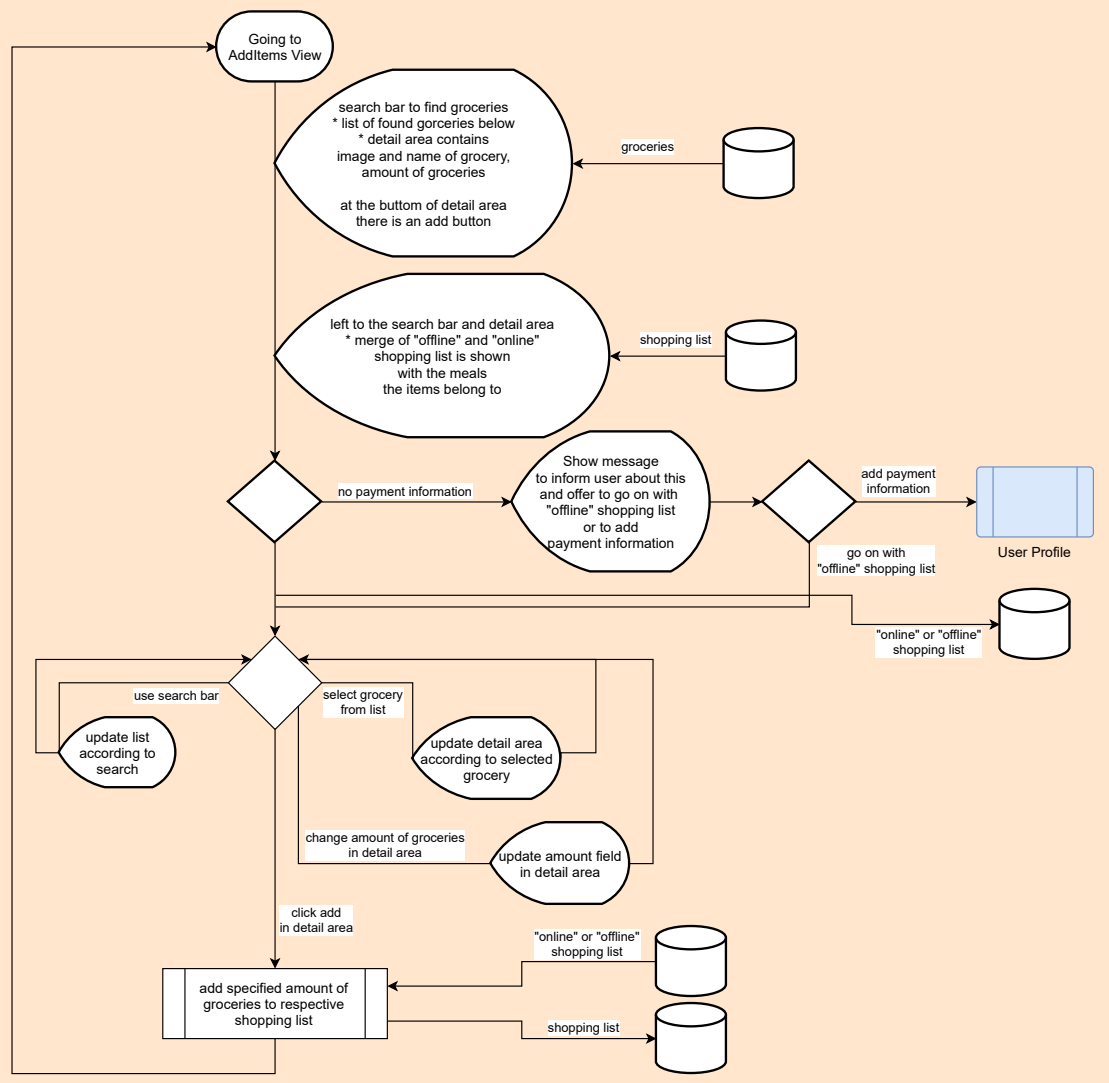


Figure A.1.: Add items/groceries to the shopping list use-case as a flowchart

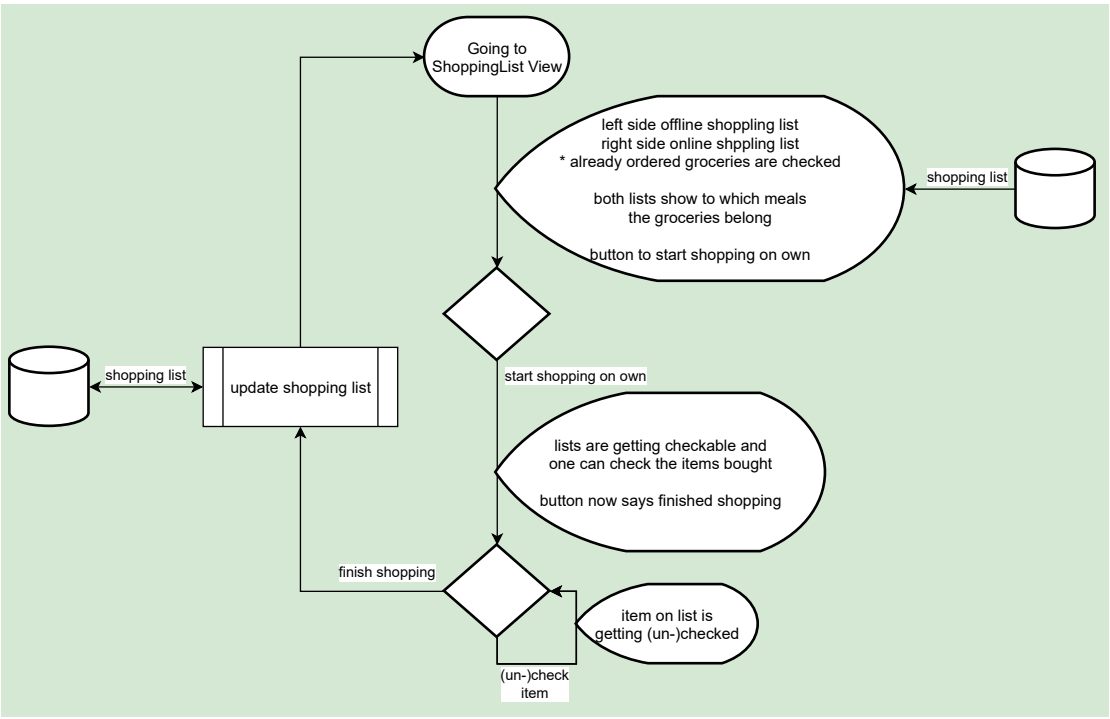


Figure A.2.: Create a combined shopping list for own shopping use-case as a flowchart

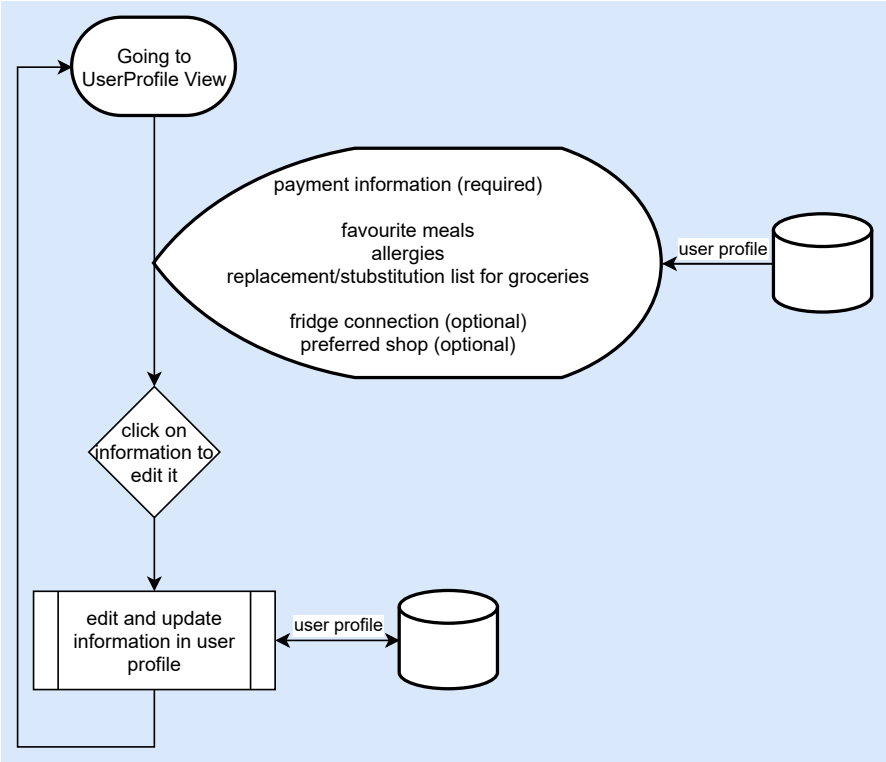


Figure A.3.: User profile use-case as a flowchart

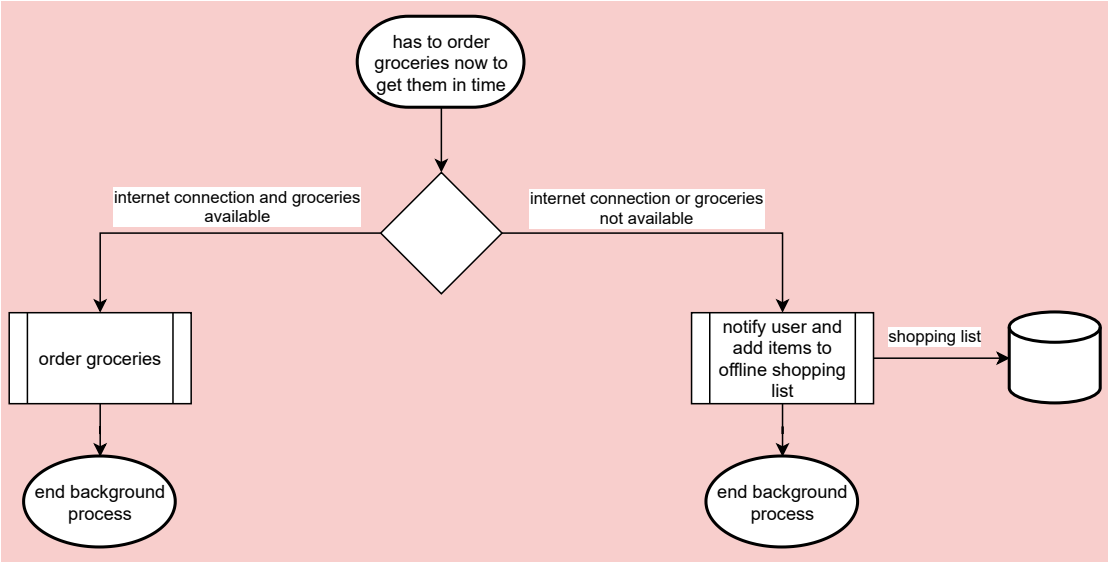


Figure A.4.: Error use-case groceries cannot be ordered as a flowchart

List of Figures

List of Tables

Bibliography

Bainbridge, L. (1983). Ironies of automation. *Automatica*, 19(6):775–779.

Bye, A., Hollnagel, E., and Brendeford, T. S. (1999). Human–machine function allocation: a functional modelling approach. *Reliability Engineering & System Safety*, 64(2):291–300.

Chapanis, A. (1965). On the allocation of functions between men and machines. *Occupational Psychology*, 39(1):1–11.

Endsley, M. R. and Jones, D. G. (2016). *Designing for situation awareness: an approach to user-centered design*, chapter 2, pages 13–29. CRC Press, Boca Raton ; London ; New York, second edition.

Fitts, P. M., editor (1951). *Human engineering for an effective air-navigation and traffic-control system*. National Research Council, Div. of., Oxford, England.

Lee, J. and See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80.

Parasuraman, R., Sheridan, T. B., and Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):286–297.

Parasuraman, Raja, S. T. and Wickens, C. (2008). Situation awareness, mental workload, and trust in automation: Viable, empirically supported cognitive engineering constructs. *Journal of Cognitive Engineering and Decision Making*, 2:140–160.

Save, L. and Feuerberg, B. (2012). Designing human-automation interaction : a new level of automation taxonomy. In de Waard, D., Brookhuis, K., Dehais, F., Weikert, C., Röttger, S., Manzey, D., S.Biede, Reuzeau, F., and Terrier, P., editors, *Human Factors: a view from an integrative perspectiv*. HFES Europe Chapter Conference Toulouse.

Sheridan, T. B. (1995). Human centered automation: oxymoron or common sense? In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 1, pages 823–828.

Sheridan, T. B. (2000). Function allocation: algorithm, alchemy or apostasy? *International Journal of Human-Computer Studies*, 52(2):203–216.

Wickens, C. D., Hollands, J. G., Banbury, S., and Parasuraman, R. (2013). *Engineering Psychology and Human Performance*, chapter 12, pages 345–349. Always learning. Pearson, Munich, fourth edition.