

|  |  |
| --- | --- |
|  | |
| Staff Management System  **User Requirements Specifications** | |
| **Team no: LST3**  Salih Ekici, Mohammed Ajimi, Nick Bulen, Sian Lauwers, Emalisa Antonioli, Siebe Camerman | **2nd phase IT Factory** |
|  |
|

**Table of contents**

[**Table of contents 2**](#_heading=h.gjdgxs)

[**Introduction 3**](#_heading=h.30j0zll)

[**1**](#_heading=h.1fob9te) **Assignment Description 4**

[**1.1**](#_heading=h.2jxsxqh) **Background information 4**

[**1.2**](#_heading=h.z337ya) **Objectives and target groups 4**

[**2**](#_heading=h.3znysh7) **Requirements analysis 5**

[**2.1**](#_heading=h.2et92p0) **Functional Requirements 5**

[2.1.1](#_heading=h.tyjcwt) Use Case Diagram 5

[2.1.2](#_heading=h.3dy6vkm) Use Case Descriptions with prototypes 6

[**2.1.2.1**](#_heading=h.1t3h5sf) **Log in:** 6

[**2.1.2.2**](#_heading=h.4d34og8) **Edit profile:** 6

[**2.1.2.3**](#_heading=h.2s8eyo1) **Manage schedule:** 7

[**2.1.2.4**](#_heading=h.17dp8vu) **Manage to-do list:** 7

[**2.1.2.5**](#_heading=h.3rdcrjn) **Archive paperwork:** 8

[**2.1.2.6**](#_heading=h.26in1rg) **Manage invoices:** 8

[**2.1.2.7**](#_heading=h.lnxbz9) **Create user:** 9

[**2.1.2.8**](#_heading=h.3j2qqm3) **Manage user:** 9

[**2.1.2.9**](#_heading=h.35nkun2) **Manage meals:** 10

[**2.2**](#_heading=h.1ksv4uv) **Non-functional requirements 11**

[**3**](#_heading=h.44sinio) **Priority per functionality 12**

# Introduction

This document presents an analysis of the user requirements specifications for the Staff Management application. First the assignment will be introduced, followed by the use case diagram and use case descriptions.

# Assignment Description

Introductory information for this section of the report.

## Background information

In this project an application will be created to help an au-pair, who helps run a household that contains staff and an occasionally visiting child, to direct household staff and keep track of the housework, as well as schedule childcare. The staff will be able to manage their schedule and to-do-list, as well as upload invoices all in one central place. This will help the head of the household, the au-pair, to easily maintain an overview of everything that is happening, as well as plan the meals for the staff.

Currently everything is organized in a messy manner; everything is tracked on separate paper, emails, etc. This causes confusion and wastes time. Having a central application would make everyone’s work easier as everyone knows where what information can be found.

The household is also frequently visited by a grandson with autism spectrum disorder. A house filled with people could be a stressor for him, as could certain foods, which could ruin his entire day. If he had an overview of who works when and what is eaten, he would be able to plan his visits to his grandparents more efficiently.

## Objectives and target groups

The head of the household, the au-pair and the family, will be able to keep a better overview. They will have a central place in which they can see their entire schedule (i.e., who of the staff is coming in when, when is their grandson visiting, what is the meal plan for the week). They will also have a central place to keep track of their paperwork, making it less likely for anything to get lost or forgotten.

For the staff it will be easy to plan their work activities. It would be easy for them to manage their schedule and their to-do-list. They could also easily send any invoices to the head of the household.

For the grandson it will be beneficial to be able to easily get an overview of who will be present at what time and what food will be served on what day, as this will help him avoid any stressors and avoid him coming in when the house is crowded and ultimately make his stay at his grandparents more pleasurable. Furthermore the application would be an easy way for him to notify his grandparents and the au-pair of when he is visiting.

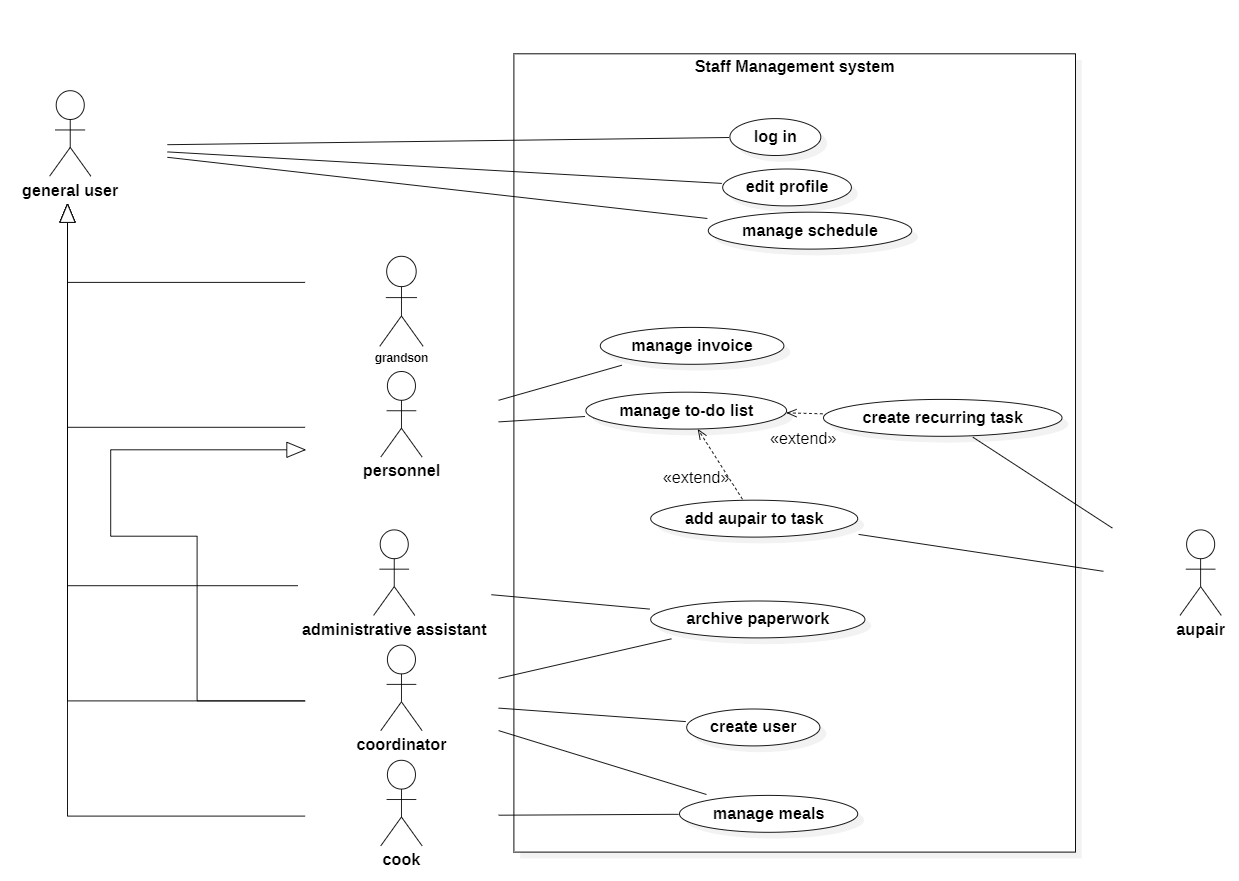
# Requirements analysis

Introductory information for this section of the report

## Functional Requirements

Introductory information for this section of the report

### Use Case Diagram



### Use Case Descriptions with prototypes

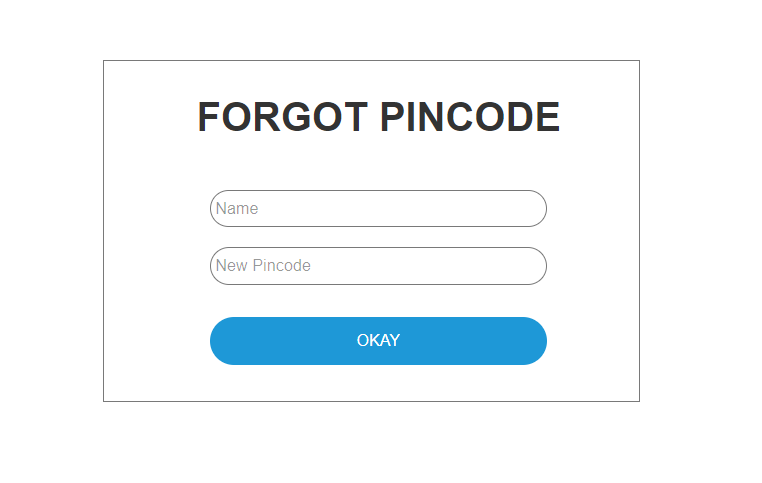
#### **Log in:**

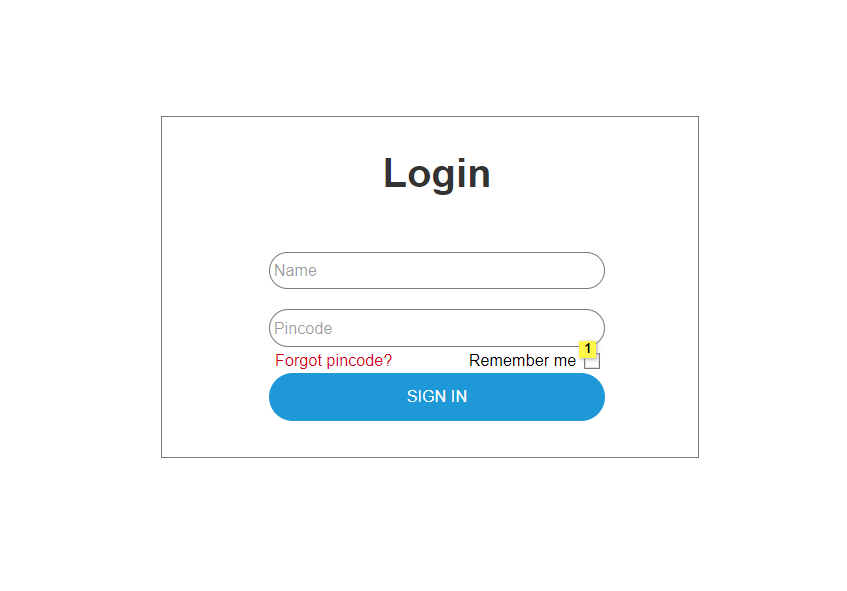
Functionality: As a general user, personnel, administrative assistant, coordinator and cook, I can log in to my profile.

Normal flow: The system asks the user to enter their name and personal PIN code. The actor enters their name and their personal PIN code. System displays a confirmation message. The actor confirms.

Alternatives:

Credentials incorrect: The system presents an error message notifying them their username or PIN is incorrect. The actor re-enter his credentials.

Forgot pincode: The user can press forgot pincode, the system will present a screen where you can change your pincode. The actor gets send back to the log in page and can log in again.

**Screens:**

|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | When checkbox is enabled, name and pincode is automaticaly filled in next time opening this page |
| SIGN IN | Goes to homepage signed in as a user |
| Forgot Pincode | Goes to the forgot pincode to get a new pincode |
| OKAY | Goes back to login to login with your new pincode |

#### **Edit profile:**

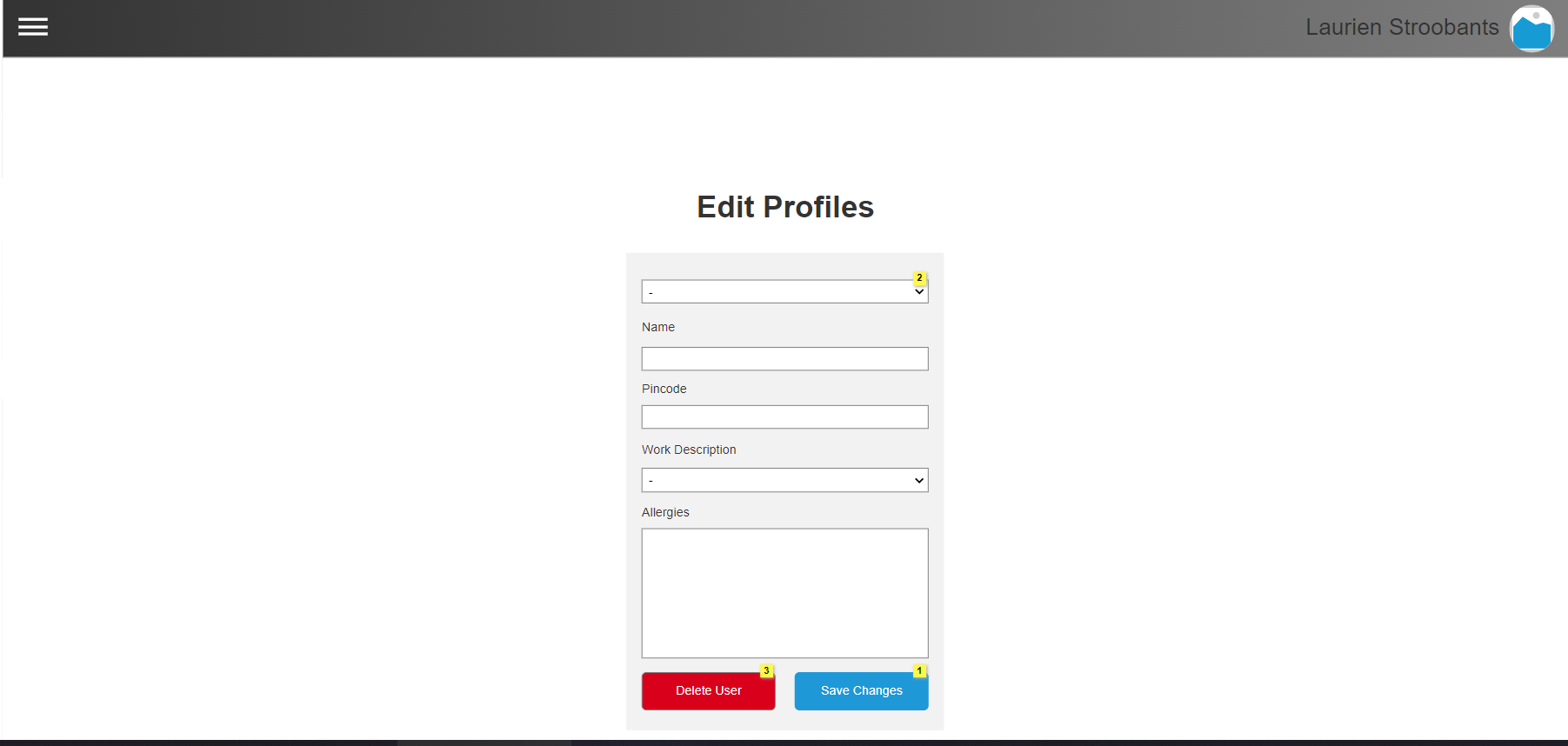
Functionality: As a general user, personnel, administrative assistant, coordinator and cook, I can edit my profile.

Precondition: General user is logged in.

Normal flow: The system displays the name, pincode, job description, and the allergy overview of the actor. The actor changes their allergy list. The actor then saves this information. The system displays the updated information.

Additional remarks:

* The coordinator can edit everyone’s profile.

**Screens: **

|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | When clicked, update the users information in the database and update the list of profiles |
| 2 | When a profile is selected from the droplist, display all of the current information regarding that profile in the text boxes below. |
| 3 | Delete selected user from database |

#### **Manage schedule:**

Functionality: As a general user, personnel, administrative assistant, coordinator (,the grandson) and cook, I can manage the schedule.

Precondition: User is logged in.

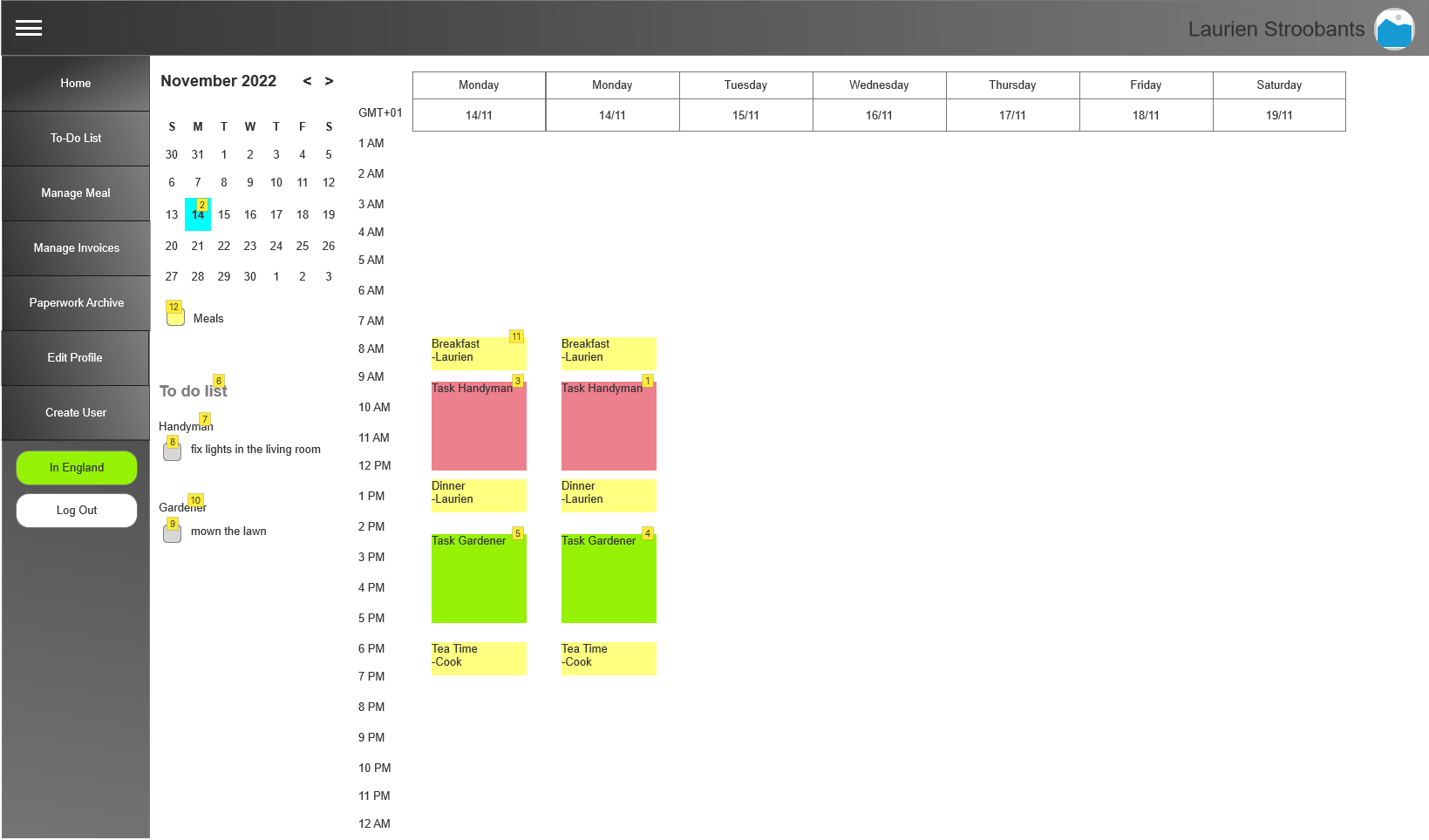
Normal flow: The system shows the actor’s weekly schedule. The system will also show what the meal will be for the week.

Alternatives:

* Adjust schedule:The system shows the current state of the schedule. The actor adds themself to a day and time slot. The system shows the new entry. The actor confirms the new entry. The system shows the updated schedule.
* The day has already passed: The actor schedules themself to a date and time slot. The system displays an error message indicating that the day has already passed.

Additional remarks: The coordinators and the grandson can see everyone’s schedule.

**Screens:**

****

|  |  |
| --- | --- |
| Action | Explanation |
| 1 | Only Handyman and Coordinator can see |
| 2 | 1 click selects the week, double click select the day only |
| 3 | Only Handyman and Coordinator can see |
| 4 | Only Gardener and Coordinator can see |
| 5 | Only Gardener and Coordinator can see |
| 6 | to do list of tasks of everyone is seen by the coordinator  only the personal can see their own |
| 7 | click shows the to do list of the handyman  handyman can see his own tasks already |
| 8 | checked equals task is done |
| 9 | checked equals task is done |
| 10 | click shows the to do list of the gardener  gardener can see his own tasks already |
| 11 | On click, edit the content of the meal. |
| 12 | Checkbox to show this in the schedule. |

#### **Manage to-do list:**

Functionality: As a personnel and coordinator, I can manage my to-do list.

Precondition: User is logged in.

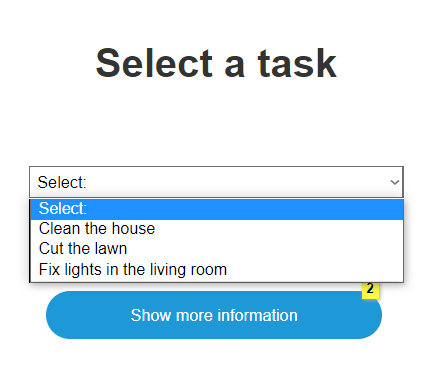
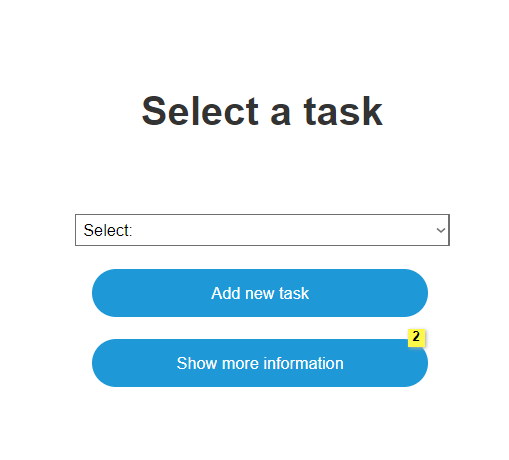
Normal flow: The system displays the to-do list of the actor. The actor creates a new task. The user enters a description of the task. The actor saves the new task. The system displays the updated to-do list.

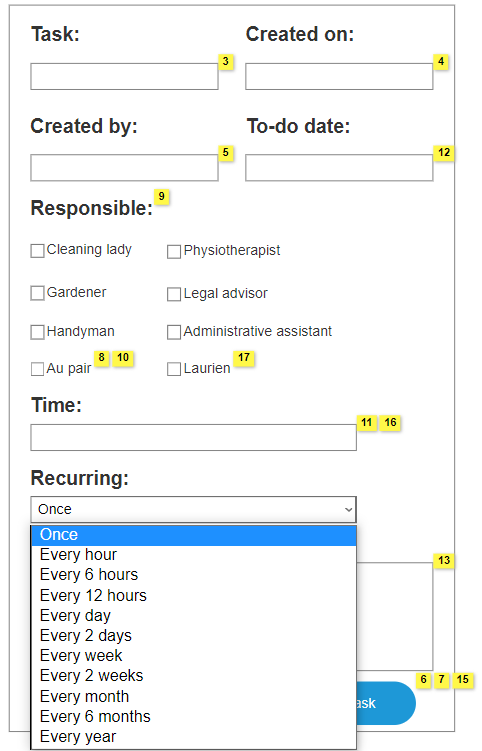
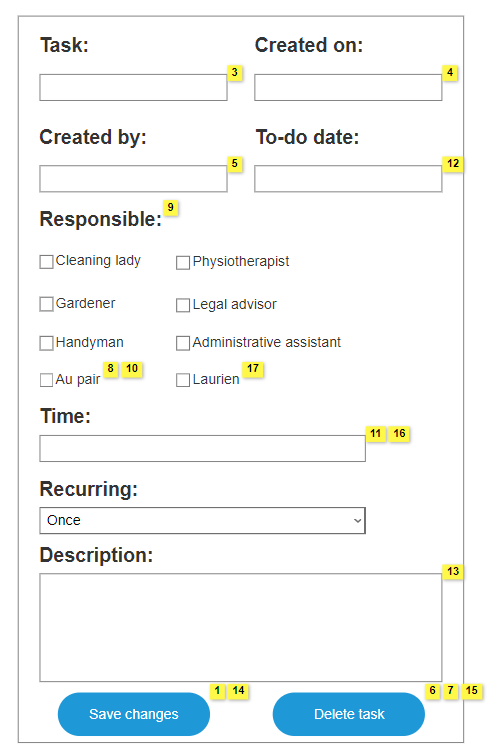
Alternatives:

* Add au pair to task: The actor adds the au pair to the task. The system sends a notification to the au pair.

Additional remarks: The system updates the schedule when a task is added.

**Screens:**

****

****

|  |  |
| --- | --- |
| Action | Explanation |
| 1 | When adding a new task this will change to Save and will go to the middle. |
| 2 | Everyone can press this but only a coordinator/person that created the task can change information in this. |
| 3 | Automatically fills in with the chosen task |
| 4 | Automatically fills in with the date it was made/day it is you create it |
| 5 | Automatically fills in with the name of the actor that is logged in |
| 6 | Only shows when more information is pressed |
| 7 | When pressed the task will be archived |
| 8 | au pair = Laurien? |
| 9 | Everyone can invite anyone. |
| 10 | When selecting the au pair, they will get a notification. |
| 11 | Set a time for when this task needs to be done. For example: 8:00-11:00 |
| 12 | The date it has to be done |
| 13 | Extra Information of the task |
| 14 | When clicking this button, it should automatically be added to the schedule. |
| 15 | Will be deleted from the schedule |
| 16 | Format: hh:mm - hh:mm |
| 17 | Can not add Laurien if she is signed off out of England. |

#### **Archive paperwork:**

Functionality: As an Administrative Assistant / coordinator, I can archive paperwork.

Precondition: Administrative Assistant / coordinator is logged in.

Normal flow: The system shows a list of archived paperwork sorted by category then by date. The actor chooses to archive new paperwork. The system asks to fill in the name, date, and category, and to add files. The actor fills in all the information and adds the files. The system displays the updated list of archived paperwork.

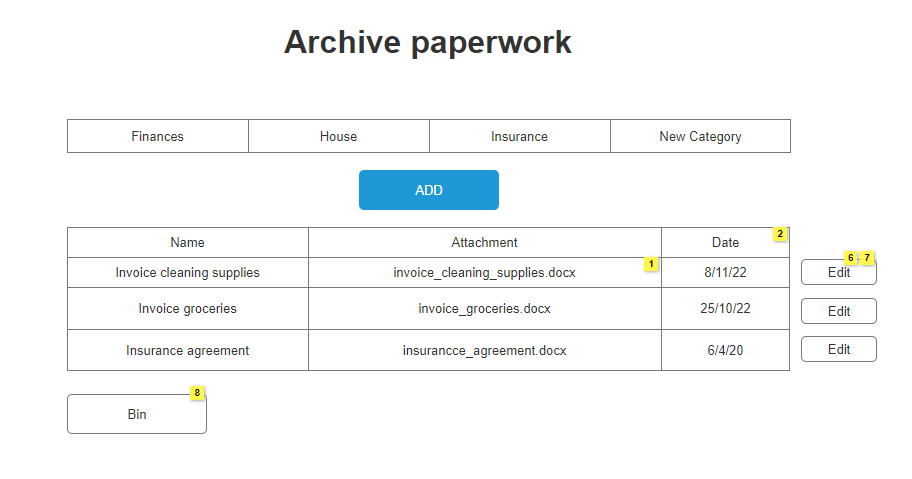
Alternatives:

* View paperwork: the actor clicks on a piece of paperwork in the list of archived paperwork. The system shows the name, date, and category of the paperwork with added files.
* Failed to add paperwork: System shows error that the actor needs to fill in all the required fields.

Paperwork could not be submitted:  The system displays an error message prompting the actor to resubmit the invoice.

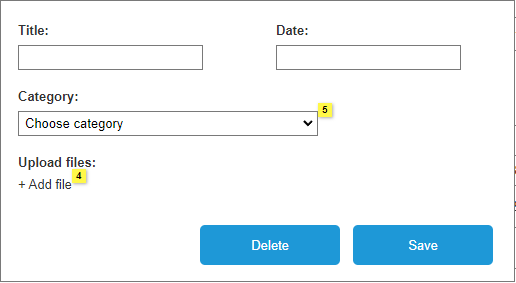
**Screens:**

7\_Paperwork\_Archive

****

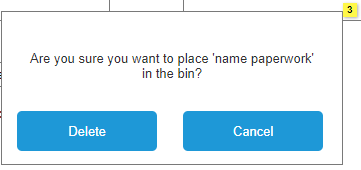
|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | This should be a file, if you click it, you open that file. |
| 2 | The paperwork is ordered by date |
| 6 | When paperwork is removed it should be archived/placed in a bin that is accessible to the coordinators. I would keep it in the category it is in now and add an 'archived' category so searching in the bin is easier. |
| 7 | Clicking this button prompts an already filled in screen: “7\_Paperwork\_Archive(upload prompt)” |
| 8 | Clicking this button should show a table with all removed paperwork. I would keep the categories for that table as well, if that's doable. |
| ADD | To screen “7\_Paperwork\_Archive(upload prompt)” |
| EDIT | To screen “7\_Paperwork\_Archive(upload prompt)” |
| New Category | To screen “7\_Paperwork\_Archive(New category)” |
| Finances/House/Insurance | Displays list of paperwork of that category |

7\_Paperwork\_Archive(Upload prompt)

****

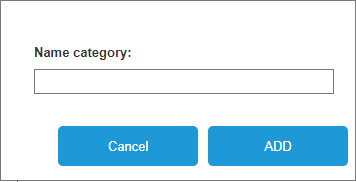
|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 4 | This should be a link that allows you to upload files. |
| 5 | Categories should stay up to date with the categories that can be created. |
| Save | To screen “7\_Paperwork\_Archive” and saves changes |
| Delete | To screen “7\_Paperwork\_Archive(Delete paperwork)” |

7\_Paperwork\_Archive(Delete paperwork)

****

|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 3 | 'name paperwork' should be replaced with the name of the paperwork the user is trying to delete. |
| Delete | To screen “7\_Paperwork\_Archive” and deletes selected paperwork |
| Cancel | To screen “7\_Paperwork\_Archive(upload prompt)” |

7\_Paperwork\_Archive(New category)

****

|  |  |
| --- | --- |
| **Action** | **Explanation** |
| Cancel | To screen “7\_Paperwork\_Archive” |
| ADD | To screen “7\_Paperwork\_Archive” and adds given category |

#### **Manage invoices:**

Functionality: As a coordinator / personnel, I can manage invoices.

Precondition: coordinator/personnel is logged in.

Normal flow: The system shows the list of the invoices. The system displays options for the actor (delete, archive, submit). The actor archives an invoice. The system shows the updated list of invoices.

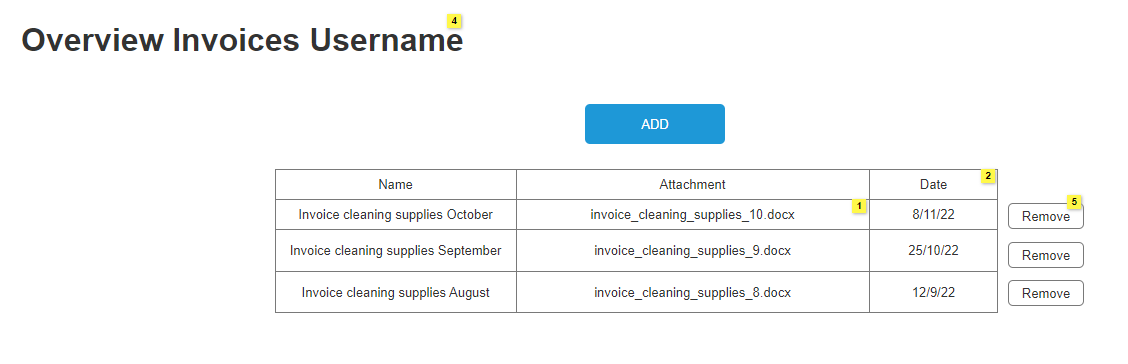
Alternatives:

* Delete: The actor chooses an invoice to delete. The actor deletes invoices. The system displays a confirmation message. The actor confirms. The system displays the updated list of invoices.
* Invoice could not be submitted:  The system displays an error message prompting the actor to resubmit the invoice.

Additional remarks: As a personnel, I can only submit invoices.

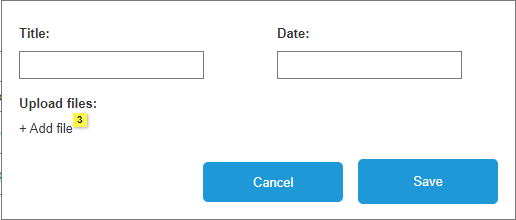
**Screens:**

8\_Manage\_Invoices



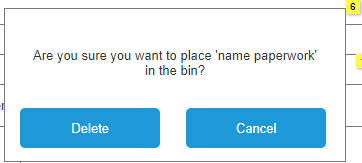
|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | This should be a file, if you click it, you open that file. |
| 2 | The paperwork is ordered by date |
| 4 | It would be nice if the username was the name of the user logged in so it is clear they only see their own invoices. |
| 5 | When the invoices are removed they should be archived/placed in a bin that is accessible to the coordinators. The removed paperwork should disappear from this table. |
| ADD | To screen: “8\_Manage\_Invoices(Add invoice)” |
| Remove | To screen: “8\_Manage\_Invoices(Remove invoice)” |

8\_Manage\_Invoices(Add invoice)



|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 3 | This should be a link that allows you to upload files. |
| Cancel | To screen “8\_Manage\_Invoices” |
| Save | To screen “8\_Manage\_Invoices” |

8\_Manage\_Invoices(Remove invoice)



|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 6 | 'name paperwork' should be replaced with the name of the paperwork the user is trying to delete. |
| Delete | To screen “8\_Manage\_Invoices” and deletes the selected invoice |
| Cancel | To screen “8\_Manage\_Invoices” |

#### **Create user:**

Functionality: As a coordinator, I can add users.

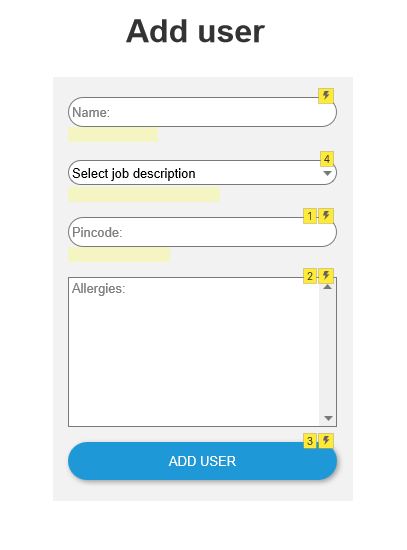
Precondition: Coordinator is logged in.

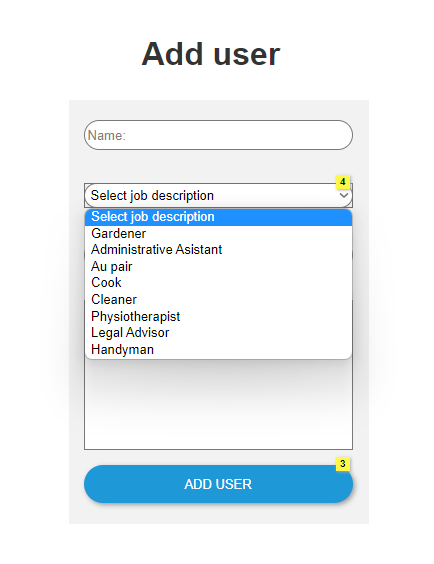
Normal flow: The system displays a blank form. The actor will fill in the necessary details (name, pincode and allergies) and select the role of the new user from a dropdown list. The actor clicks “Add User”. The system saves the new user.

Alternatives:

* Information was incorrect: The actor is able to edit the information of every user and saves it again.

**Screens:**





|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | 4 digits only |
| 2 | Leave empty when no allergies |
| 3 | User’s name will be added to the 'Edit profile' page. |
| 4 | Default selection is "select job description" and should be grayed out like the placeholder for name and pincode |
| Add User | User details will be added to database |

#### **Manage meals:**

Functionality: As a coordinator/ cook , I can plan meals.

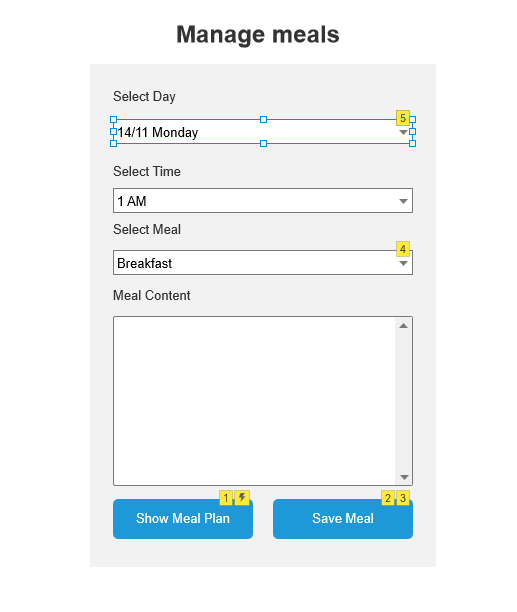
Precondition: Coordinator/cook is logged in.

Normal flow: The system will display a blank form. The actor will enter the necessary details for the meal (Selects the day, selects the time, selects the meal, enters the content of the meal). The actor saves the meal. The system adds the meal to the meal plan and displays it on the calendar.

Alternatives:

* Details for meal weren’t added: The system will display a message indicating that the fields weren’t filled in

**Screens:**



|  |  |
| --- | --- |
| **Action** | **Explanation** |
| 1 | When clicked, redirect to the homepage where the calendar is. |
| 2 | when clicked, save the meal on the calendar in specified time slot |
| 3 | when clicked, save the meal on the calendar in specified time slot |
| 4 | Selected Meal should show up as a label on the calendar |
| 5 | Display current day and the next 6 days on this dropdown list |

## Non-functional requirements

Besides the above described functional requirements, the application also needs to fulfill some non-functional requirements. The most important non-functional requirements involve the graphic user interface. It is to be assumed that the application will be used primarily by people who have little to no knowledge of computers. It is important that the design of the application is simple and straightforward so that anybody can intuitively find their way around. On top of this the application will be used by elderly people. For this reason there should be an alternative version of the application with larger font sizes and clear colour contrast (take into account that older people often struggle with yellows, blues, and greens and often need to use low brightness).

While designing the graphic user interface the platform should also be taken into account. For now, the application will run on a desktop, but in the future it would be ideal if it could be expanded to mobile devices. This should be considered for the design, for instance by avoiding a lot of details and working with columns for the layout.

This app is meant to be used by both people situated in the UK and in Europe. This means it will be used from (at least) two different time zones. This is important to consider while designing the calendar. There should never be any conflict due to the difference in time zones for the different users.

In the app people will be storing details of their daily schedules. This is sensitive information and should be protected as such. Schedules may only be accessible to people when they are logged in, and they should only be able to see the schedules that are relevant to them. Account details should only be accessible to the admin and the user themself. [General European guidelines](https://www.enisa.europa.eu/topics/incident-reporting/for-telcos/guidelines/technical-guideline-on-minimum-security-measures) should also be considered.

In conclusion, during the development of this application attention should be spent on accessible design. On top of that, different time zones and basic security guidelines should be considered.

# Priority per functionality

Introductory information for this section of the report

Indicate for all functionalities / use cases which MoSCoW-priority should be assigned.

## Navigation bar:

* Logged in as …: Must have
* Navigation to other pages: Must have
* Log out button: Must have
* In/out England button (for Laurien only): Must have

## Log in:

* Name text box: Must have
* Pincode text box: Must have
* Forgot pincode: Must have
* Remember me: Should have

## Create User:

* Name text box: Must have
* Job description dropdown: Must have
* Pin code: Must have
* Allergy text box: Should have

## Edit Profile:

* Droplist with current profiles: Must have
* Name text box: Must have
* Pincode text box: Must have
* Job description dropdown: Must have
* Allergy text box: Should have
* Delete profile button: Must have
* Save changes button: Must have

## Archive paperwork:

* Categories: Should have
* Table with paperwork: Must have
* Edit function: Must have
* Bin/delete: Must have
* Add function: Must have
* Attachments: Should have

## 

## Manage invoices:

* Table with all invoices: Must have
* Name: Must have
* Attachment: Must have
* Date: Must have
* Remove: Must have
* Add function: Must have

## Manage Schedule:

* Monthly calendar: Could have
* Checkboxes to turn on/off in weekly schedule: Should have
* to-do list checks: Should have
* Weekly schedule: Must have
* Meal planning: Must have

**Manage to-do list:**

* New task button: Must have
* Show more information: Must have
* List/dropbox with to-do’s: Must have
* Task name: Must have
* Created on: Should have
* Created by: Should have
* Responsible: Must have
* Time: Should have
* Recurring: Must have
* Description: Could have

## Manage Meals:

* Day selection dropdown: Must have
* Time selection dropdown: Must have
* Meal selection dropdown: Must have
* Meal content information box: Must have
* Save meal button: Must have
* Show meal plan button: Should have