ICT Berufsbildung
Formation professionnelle
Formazione professionale

ICTskills 2023 | Skill 09

Assignment

Version 1.0

Version: 1.0
Authors: Thomas Kehl, Ueli Niederer

ICT-Berufsbildung Schweiz
Waisenhausplatz 14, CH-3011 Bern
Telefon +41 58 360 55 50
info@ict-berufsbildung.ch
www.ict-berufsbildung.ch

Table of contents

Assignment

ICT Berufsbildung
Formation professionnelle
Formazione professionale

1 Project overview

HappyHeal Medicals Ltd. is an outfitter for medical institutions, selling various medical products (tools, devices, consumables, etc.). Complete tracking of goods and their history/lifecycle is key to guarantee the supreme quality expected. To optimize their product tracking mechanisms, the company wants to digitalize the tracking by harnessing a dedicated software. This software shall allow for:

1. Registering and enforcing different steps in the Lifecycle of a product.
2. Triggering actions dependent on the transition from one status to another.
3. Querying the status and the history of a product.

Your task is to create such an application software according to the description outlined in following chapters. The assignment is split into two separate parts. The second part may not be started, before the results for the first part have been handed in.

2 Application overview

Note:	Besides the description in this document, you are provided with a set of data to be imported (see chapter 3). Looking at the data provided may further help clarifying and understanding the descriptions in this document.

The application shall allow for tracking the status of different instances of products sold by the company. The product instance's status represents its current position within its lifecycle.

Example:	HappyHeal Medicals Ltd. sells blood pressure measurement devices (product). For traceability, the company wants to keep track of every single instance of these devices (product instance). Hence, they record the status of every device instance along with its serial number.

The application is split into two major functionalities:

1. **Designing** workflows and assigning them to product groups and/or products.
2. **Update and review** the status of an actual product instance according to the assigned workflow.

This assignment is split along these two functionalities in two corresponding sessions.

2.1 Definition of a workflow

A workflow may be perceived as sort of a state machine, describing the various states a product instance could be in, and the possible transitions to future states.

2.1.1 Representation of a workflow

The representation of a workflow is basically a directed graph consisting of nodes (states) and edges (transitions), just like an UML state diagram:

- **States** are depicted as rectangles styled as follows:
  - Rectangles may or may not be drawn with rounded corners.
  - The shape displays the state's name in its centre.
  - Each state can be filled in a, user defined colour.
- **Transitions** indicate changes between states, allowed by the workflow.
  - Transitions are drawn as arrows pointing from the departing state to the target state.
  - Each transition may be configured with an action that is executed, whenever the transition is executed (see section 6.6 for details).

In addition to the ordinary states, one special (pseudo) state exists: The initial state. This state serves as a marker for the initial state a product instance is set to. Whenever a workflow is started, the product instance 's state will automatically set to the state, the transition departing from the initial state is pointing to. Only one initial state can be set. The initial state is styled as circle with black fill colour.

2.1.2 Maintaining and versioning of workflows

Each workflow has assigned an integer, increasing version number where the largest version number marks the most recent version of the workflow. A workflow version can be modified if not a single product instance yet referenced the workflow. As soon as a workflow version is being used by a product instance, the workflow version becomes immutable (i.e., cannot be modified anymore).

Changes to the workflow result in an updated version. Such an updated version is created by effectively copying the existing workflow, increasing its version number, and finally modifying the copy created. After saving this updated version, it will only be applied to new product instances, while existing product instances still follow the version they started with.

A similar logic applies to deleted workflows: While they are not assigned to new product instances, existing product instances follow the workflow version they started with.

2.2 Products, product groups and Workflows

To organize the multitude of products managed by the application, products are hierarchically grouped in product groups. These product groups may be further combined with other products and product groups to form a meaningful hierarchy as illustrated in Figure 1. There may be multiple root elements within the product groups. However, each element may only have one single parent element.

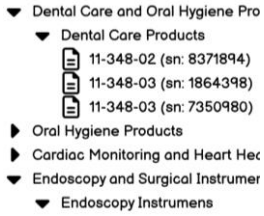

Figure 1: Example of products and product instances arranged in product groups

Workflows are assigned to either a product or one of the product groups. As this might lead to multiple workflows being (in-)directly assigned to a given product, a decision must be made, which workflow to apply. The following outlines the ruleset for the decision logic:

1. If a workflow was already applied to the product instance, this workflow is used.
2. If the product instance does not follow a workflow yet (i.e., the product instance is new), the workflow assigned to the closest node in the hierarchy is applied (i.e., the tree is searched from leaf to root for the first workflow assigned)

2.3 Data storage

The application is supposed to store its data in a relational database (i.e., either a MySQL or a MS SQL) backend. The application is expected to create the necessary structure within the database to operate properly. The concept of the data storage shall protect sensible information (i.e., passwords) from being retrieved even if the data storage would be openly accessible.

2.4 User management and login Screen

To enable full traceability, users must login through an appropriate login window, prior using the software. Such a window is illustrated in Figure 2. This allows for attaching the user's identity to changes/logs where necessary.

On the login screen, the user also decides whether the application starts in designer view (according to chapter 5) or in user mode (according to chapter 6).

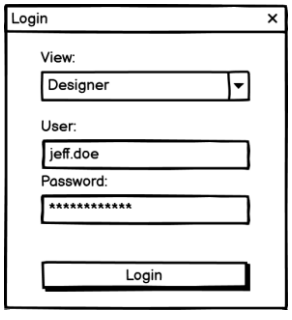

Figure 2: Mock-up of a Login dialog

A management of the user data (i.e., adding, removing, disabling users) apart from the import outlined in chapter 3 does not have to be implemented. The login dialog shall be shown upon application startup or after the application is locked. The application locks itself after a certain time of inactivity. If the application locks, the user interface is not operatable

anymore and the login dialog is displayed again with the username fixed to the user logged in. The timeout shall be configurable in the application configuration (see section 2.5) and may be set from as low as a few seconds to up to a several minutes.

The login screen shall fulfil the following requirements:

- For privacy, the password entered shall be masked, to prevent:
  - Identification of typed characters
  - Number of characters in the password (e.g., by displaying a different number of masking characters than actually typed).
- Upon pressing **`Login`**, the credentials entered shall be checked against the user data in the backend.
- If credentials are valid, the dialog should close, and the main window shall be displayed.
- If the credentials turn out to be invalid, the following error message shall be displayed:

  `Unable to login, please check credentials.`

- In case the user closes the login dialog, the application is automatically shut down.

2.5 Application configuration

Any configuration expected to be setup with the application is to be stored in a configuration file named **`setup.json`**. This file is automatically loaded if found in the working directory. The absence of the configuration file shall trigger the use of meaningful (judgement by the creator of the application) default values and the creation of a “template configuration file” representing the values used at the expected location.

3 Import of data

The application shall provide an import mechanism, allowing to load the database with data provided through a set of JSON-files. This mechanism is triggered whenever JSON-files are present in a folder named import within the working directory. After finishing the import, the source files in the import folder shall be deleted automatically.

Upon import, the current content of the database is deleted, and the database is filled with the import data only. There is no need to merge potentially existing data.

The import data is provided through the file **`data.json`** on the competition's USB stick. The data is serialized as JSON document composed of three main sections, marking the three major information types processed within the application.

Note: The data to be imported does not necessarily reflect all attributes that are necessary to cover all use cases outlined in this assignment (e.g., there is no created by-information available in the **`data.json`** file, while this is supposed to be recorded by the application). It is subject to the creator of the application to implement an appropriate strategy to deal with that missing information.

3.1 ProductGroupTree

The product group tree contains the products along with their instances, embedded within one or multiple product groups, effectively forming a composite structure. Each product instance is provided with a serial number and a contact mail address (**`CustomerMail`**).

3.2 Workflows

An array of workflow objects. Each workflow is represented as a basic named state machine consisting of states and transitions between the states. While the states are described through attributes such as colour or title, the transitions always have a starting point (**`from`**) and a target (**`to`**). The transitions also carry the setup for actions supposed to be triggered upon execution of the transition.

3.3 Users

An array of user credentials and meta information for users eligible to work with the application.

4 Corporate Design

This chapter describes the visual identity for HappyHeal Medicals Ltd. To blend in with the other appearances of HappyHeal Medicals Ltd., the corporate design must be respected during application development.

4.1 Logo

HappyHeal Medical Ltd.'s logo features an interpretation of the rod of Asclepius. For maximum flexibility in use, the company's logo is provided as one SVG file named **`happyheal.svg`**:

Figure 3: HappyHeal Medical Ltd. Logo [1]

4.2 Colours

While the background of the application can be left in the framework's default colour, the application is expected to use the palette defined in Table 1.

Table 1: HappyHeal Medical Ltd. colour palette

Sample	Colour Name	RGB	Primary	Secondary
	HappyHeal Medical Orange	#CD9323	X	
	Accent 1	#81932D	X	
	Accent 2	#3D874E	X	
	Tropical Rain Forest	#007567		X
	Blue Stone	#005F6B		X
	Pickled Bluewood	#2F4858		X

4.3 Iconography

In minimum the icons shown in Table 2 shall be used to depict functionalities on the user interface. These icons are provided on the competition's USB stick.

Table 2: HappyHeal Medical Ltd. icon palette

Icon	Icon Name	Filename	Original source
	HappyHeal Medical Ltd. Logo	happyheal.svg	[1]
	Add	add.svg	[2]
	REST API	internet.svg	[3]
	Mail	mail.svg	[4]
	Message Box	message.svg	[5]
	Save to File	save.svg	[6]

5 Session 1: Designer View

Goal of the first session is to create the tools to import and edit workflows. Manipulating the states for the imported product instances and implementing actions on the transitions are explicit non-goals and left for session 2.

5.1 The main window

Upon entering the designer view after login or if no workflow is currently opened (after a close action), the user is confronted with the designer's main window, displaying a list of all currently available (active) workflows as illustrated in Figure 4.

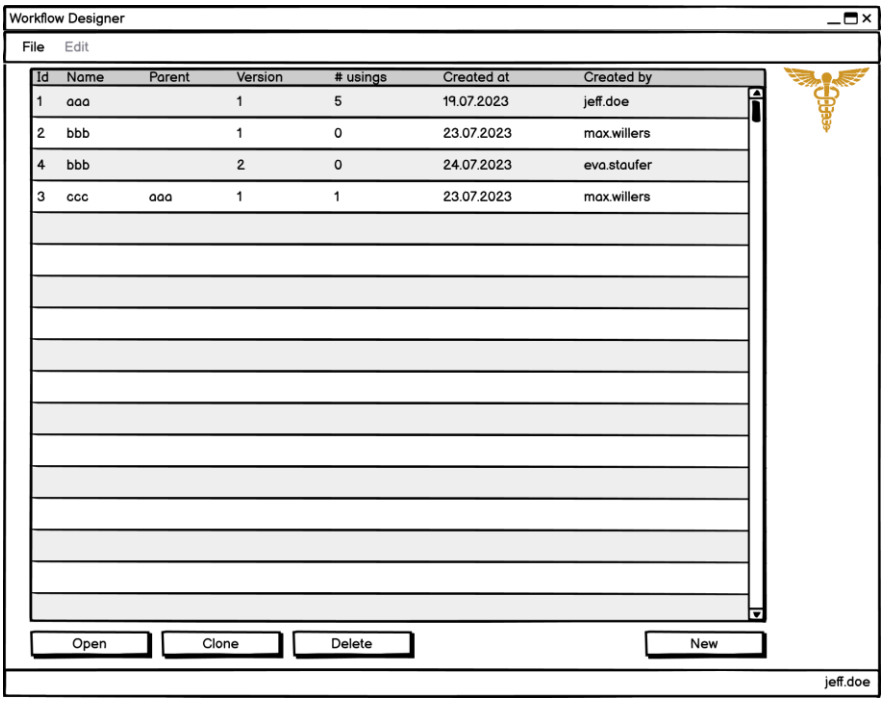

Figure 4: Designer view's main window

In this window, the user can decide either to:

- Create a new (empty) workflow (see section 5.1.1)
- Select an existing workflow from the list and
  - open (see section 5.1.2)
  - clone (see section 5.1.3)
  - delete (see section 5.1.4)

  it by clicking on the appropriate button.

The list of workflows available shall reveal the following workflow properties:

- Name of the workflow
- Name of the parent workflow (i.e., the workflows preceding version, nothing if the workflow is the first of its kind)
- Version of the workflow (integer, the number of preceding workflows + 1)
- Number of product instances currently running in the current workflow version
- Date of the workflow's creation
- Name of the creator of the workflow (i.e., the person last saving the workflow)

Each workflow presented shall be expandable to reveal preceding versions and the respective attributes.

5.1.1 Creating a new (empty) workflow

If an empty workflow is created, the workflow designer outlined in section 5.2 is opened, only with the initial state already available and a default name.

The default name is composed from the prefix **`New Workflow`**, followed by the next larger integer, making the name unique.

Example: If a workflow named **`New Workflow 1`** exists, the generated name is expected to be **`New Workflow 2`**. In case two workflows exist, one by the name of **`New Workflow 1`**, another with the name **`New Workflow 3`**, the name generated is expected to be **`New Workflow`** 4 even though there may be no existing **`New Workflow 2`**.

5.1.2 Opening a workflow

If a workflow is opened, the workflow designer outlined in section 5.2 is opened, displaying the workflow selected for opening.

5.1.3 Cloning a workflow

If a workflow is cloned, the selected workflow is copied and stored as new workflow consisting of the same (cloned) states as the selected workflow. This new workflow is opened in the workflow designer outlined in section 5.2.

The cloned workflow uses the same name as the original workflow, appended with **`(clone n)`**, where n follows the same rules as for the numbering of new workflows outlined in section 5.1.1.

5.1.4 Deleting a workflow

Once created, a workflow can never be deleted from the system. However, it shall be marked as deleted upon deletion to prevent usage of it.

5.2 Workflow Designer

The workflow designer allows for displaying and manipulating workflows. The screen as shown in Figure 5 displays the Workflow Designer. Here a workflow can be viewed and/or modified.

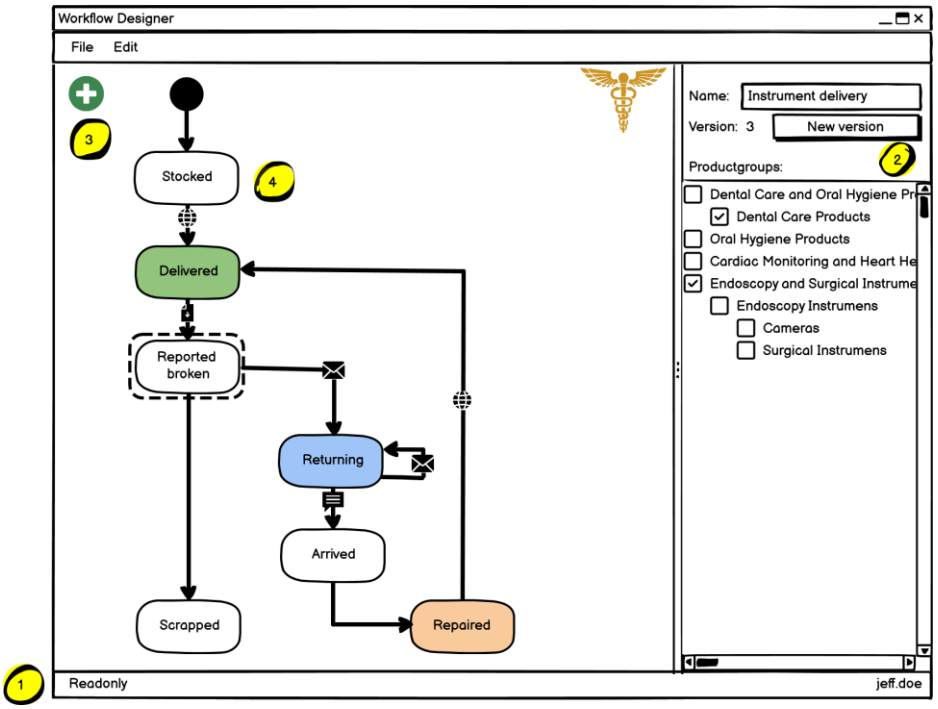

Figure 5: Workflow designer main screen

A change in the workflow shall only be stored in the database upon explicit request by the user (i.e., **`Save Workflow`**, **`Delete Workflow`**). If the user closes the application without saving, a notification occurs, asking whether to save or discard the changes made.

5.2.1 General window layout

The window consists of the following four sections (numbers refer to the callouts in Figure 5):

- Title bar displaying the application name.
- The main menu bar.
- The workflow editing area composed of
  - It is graphical representation (left pane)
  - It is metadata (right pane)
- Status bar (1) displaying:
  - **The username of the user currently logged in.**
  - **`Read-only`** if the workflow has already been initiated for a product instance. If a read only workflow needs to be changed, an updated version must be created (2). → See also section 2.1.2.

5.2.2 The main menu bar

A menu bar shall give quick access to all necessary commands in the application. The menu bar consists of two main menu items as shown in Figure 6. The commands can be triggered either through menu or – where indicated – through the respective shortcut:

- File Menu for operations revolving around managing the workflow such as:
  - **`New Workflow`**
    Closes the currently open workflow and creates a new empty workflow.
  - **`Clone Workflow`**
    Creates a new workflow by copying the currently open workflow.
  - **`Open Workflow`**
    Closes the currently open workflow and opens an existing workflow according to the user's selection.
  - **`Save Workflow`**
    Saves the currently open workflow (inactive and greyed out if workflow is unmodified).
  - **`Delete Workflow`**
    Deletes the currently open workflow (inactive and greyed out if workflow was never saved).
  - **`Exit`**
    Closes the application.
- Edit Menu for operations editing the workflow:
  - **`New Version`**
    Creates an updated version of the workflow (see section 2.1.2)
  - **`New State`**
    Inserts a new state (see section 5.2.3).

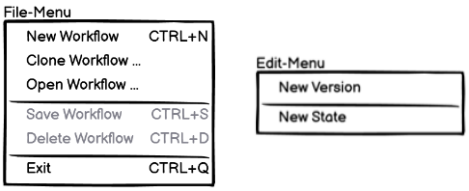

Figure 6: Workflow designer: Main menu items

5.2.3 Creating states & transitions

By clicking the Plus-symbol (3) a new state (4) is created in the workflow, which can be positioned within the window through drag & drop. Each state features a context menu as shown in Figure 7, allowing to create transitions between two states. Alternatively, a transition creation can be triggered by selecting a state through left click and pressing **`Ctrl + T`** afterwards. A selected state is decorated with a dashed outline.

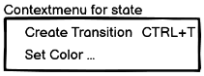

Figure 7: State context menu

After triggering the **`Create Transition`** command, the user shall select the target state by a left click. As soon as the user selected the target, the transition is created and a transition setup dialog (see section 5.2.4) appears. This process shall be guided by appropriate messages in the status bar.

The user can cancel creating a new transition by hitting the Escape-Key. Having multiple concurrent transitions pointing from A to B shall be prevented by the application.

In case the user selects **`Set Colour`**, a colour chooser is presented to set the state's fill colour.

5.2.4 Setting up transitions

Whenever a new transition is created or an existing transition is opened for edit by double clicking the transition in the Workflow Designer, a setup dialog to configure the transition is displayed. The options on the right-hand side vary depending on the selected action, as shown in Figure 8.

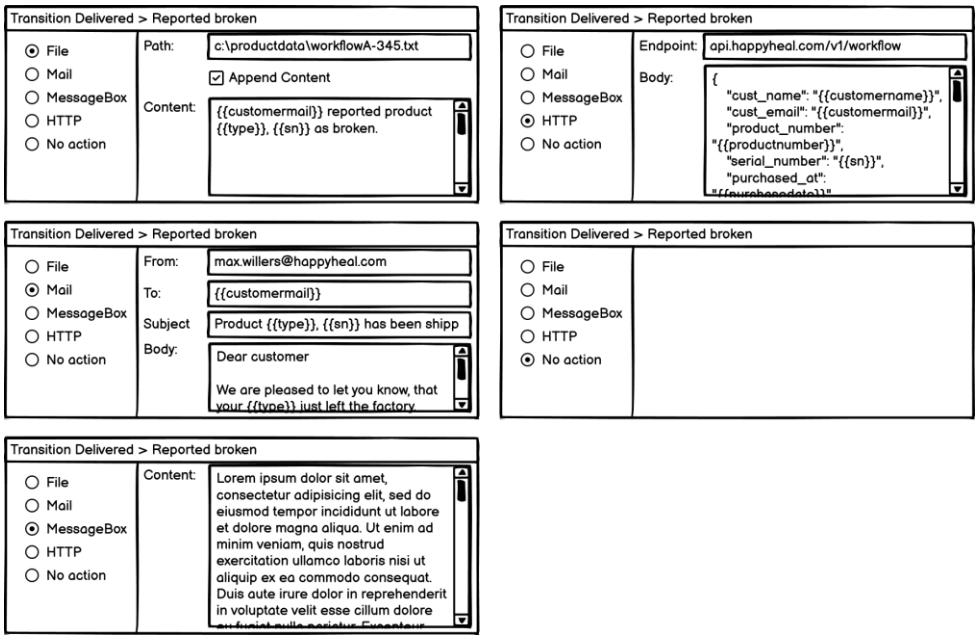

Figure 8: Transition setup dialog variations

These actions are executed whenever a product instance follows the state change along the given transition and allow for

- logging data to a file
- sending automated mails
- displaying a message box with instructions to the user triggering the transition
- posting something to a remote REST API
- doing nothing

The action configured for a given transition is presented to the user graphically by adding the respective icon (see chapter 4) to the transition arrow. If no action is configured, the icon is omitted.

For this session, the focus lies on acquiring and storing the configuration for the transitions. Implementing the functionalities is left for the follow-up session (see section 6.6).

5.2.5 Assigning workflows to product groups

A workflow can be assigned to one or more product groups by ticking the appropriate groups in the metadata (8) of the workflow as shown in Figure 5. Only one workflow can be assigned to a product group at the time. If the ticked product group is already assigned a workflow, the user shall be presented a dialog asking whether the current assignment shall be replaced with this new workflow. The question shall be:

The product group {product group name} is already assigned a workflow:
{Current workflow name + version}
shall this assignment be replaced with
{New workflow name + version}?

The data model must represent the evolvement of workflow assignments over time. I.e., the data allows to evaluate for which periods in time a certain workflow has been assigned to which product group.

5.3 End of session 1

Please hand in your results from session 1 to get access to the necessary files of session 2. While changes to the results from this session can be made during session 2, the verdict for session 1 will **not be reevaluated**.

You'll receive the password for pages 17-22 after handing in your solution for session 1

7 Glossary

Term	Definition
Product	A product *type* managed by HappyHeal Medical. (e.g., blood pressure measuring device)
Product category	Hierarchical element, grouping products and product categories together (e.g., cardiac devices)
Product instance	An actual incarnation of a product (e.g., the blood pressure measuring device with serial number ABC123DEF is an instance of a blood pressure measuring device)
Workflow	See section 2.1.

8 List of figures

9 List of tables

10 List of references

[1] Raymond, “medical,” 09 05 2017. [Online]. Available: https://thenounproject.com/icon/medical-1053629/. [Accessed 28 07 2023].

[2] Xinh Studio, “add,” 30 09 2017. [Online]. Available: https://thenounproject.com/icon/add-1298671/. [Accessed 28 07 2023].

[3] Karl, “Internet,” 11 01 2023. [Online]. Available: https://thenounproject.com/icon/internet-5500060/. [Accessed 28 07 2023].

[4] Y. R. Pebriyanto, “Mail,” 27 06 2023. [Online]. Available: https://thenounproject.com/icon/mail-5886525/. [Accessed 28 07 2023].

[5] AmruID, “Text Box Message,” 25 07 2019. [Online]. Available: https://thenounproject.com/icon/text-box-message-2895410/. [Accessed 28 07 2023].

[6] Y. Pujiyono, “Save File,” 27 04 2019. [Online]. Available: https://thenounproject.com/icon/save-file-2587173/. [Accessed 28 07 2023].

[7] C. Wanstrath, “{{ mustache }},” 19 01 2021. [Online]. Available: https://mustache.github.io/. [Accessed 19 07 2023].

[8] T. Kehl and U. Niederer, mustache manual excerpt, 1.0 ed., ICTskills 2023 | Skill 09, 2023.