

# ADIN

(ADAPTIVE INTERFACES)

Make adaptability your main power

User Manual



## Contents

<b>Introduction.....</b>	<b>2</b>
<b>Component Access .....</b>	<b>3</b>
<b>Software requirements .....</b>	<b>3</b>
<b>Set up .....</b>	<b>3</b>
<b>Application Sections .....</b>	<b>6</b>
<b>How to structure the database .....</b>	<b>21</b>

## Introduction

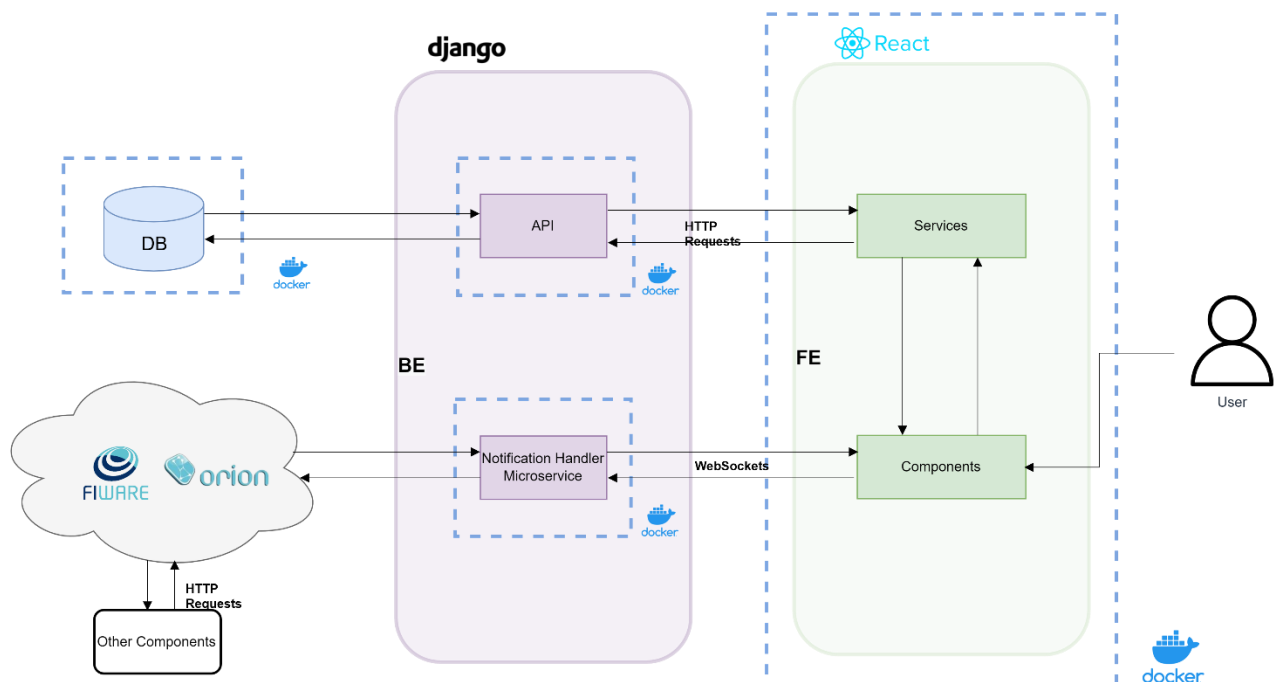
The purpose of this document is explain the user what is tof the application, how to set it up and show how to use its main features.

### ADIN - Architecture

ADIN is a web-based application capable of assisting employees in the manufacturing sector during their working time. ADIN helps avoid human error and reduce stress by providing the most relevant and specific information needed by the user, for example by guiding workers to perform tasks.

The application is comprised of four main blocks:

- Frontend (FE): Generates the interface with which the user interacts;
- Backend API (BE-API): It is in charge of defining the endpoints that provide data to the FE or receive information from it. Additionally, it defines the data models of the entities used by the application and shape the database.
- Backend Notification Handler (BE-NH): It handles the bidirectiona communication between ADIN and with other components via FIWARE.
- Database (PostgreSQL): It is where is stored all the necessary data that the application needs for its correct operation.



## Component Access

ADIN component can be found at SHOP4CF repository of RAMP Docker Registry.

The component is divided into several artifacts. In order to use ADIN, it is necessary to download the following artifacts:

- *adin-api*
- *adin-fe*
- *adin-nh*

## Software requirements

To use ADIN component, it is necessary to have Docker installed and a device able to run a web application. Additionally, it is advisable to run the application in a browser such as Google Chrome or similar.

## Set up

Once docker is installed, it is required to download the images of the artifacts, *adin-api*, *adin-fe*, *adin-nh* (*Note: download the latest version of them*) from RAMP. In the next section is explained how to run the application using docker compose.

### Docker Compose approach

Docker-compose requires having download all the images of the artifacts. Create a file named *docker-compose*, write in the document the following commands and and save it with the extension *.yml* (this can be done in a notepad program as NotePad++):

```
version: "3"

services:

  adin-api:

    image: docker.ramp.eu/shop4cf/adin-api:5.0.0

    container_name: adin-api

    command: bash -c "python manage.py runserver 0.0.0.0:8000"

    environment:

      - POSTGRES_USER= YOUR_USER
      - POSTGRES_PASSWORD=YOUR_PASSWORD
      - POSTGRES_DB=YOUR_DB
      - DB_HOST=adin-db
      - DB_PORT= 5432

    ports:
```

- "8000:8000"

depends\_on:

- adin-db

adin-db:

image: postgres:14

environment:

- POSTGRES\_USER=YOUR\_USER
- POSTGRES\_PASSWORD=YOUR\_PASSWORD
- POSTGRES\_DB=local\_adin\_db\_2

volumes:

- pgdata:/var/lib/postgresql/data
- ./db\_data/backups/adin\_db\_backup\_v6.sql:/docker-entrypoint-initdb.d/init.sql:ro

ports:

- "5432:5432"

adin-fe:

image: docker.ramp.eu/shop4cf/adin-fe:1.4.1

container\_name: adin-fe

command: npm start

depends\_on:

- adin-api
- adin-nh

ports:

- "3000:3000"

expose:

- "3000"

adin-nh:

image: docker.ramp.eu/shop4cf/adin-nh:1.3.0

container\_name: adin-nh

command: bash -c "python bridge.py runserver"

depends\_on:

```
    - orion

ports:
  - "54100:54100"
  - "8080:8080"

expose:
  - "54100"
  - "8080"

mongo:
  image: mongo:3.2
  command: --nojournal
  container_name: mongo

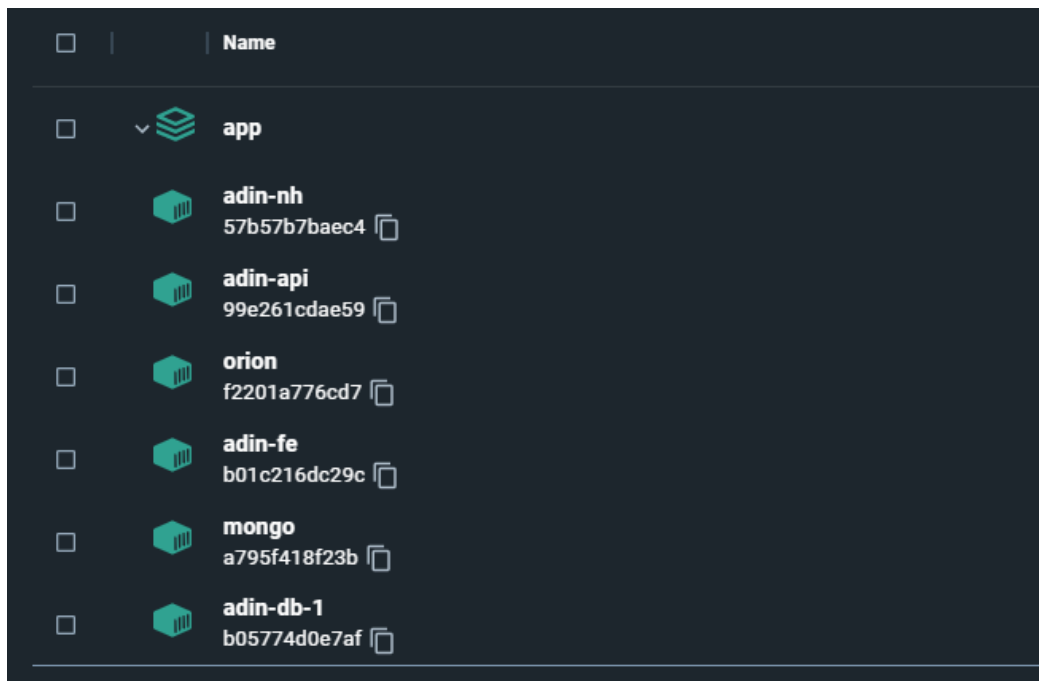
orion:
  image: fiware/orion-ld:latest
  links:
    - mongo
  ports:
    - "1026:1026"
  command: -dbhost mongo -logLevel DEBUG -logForHumans
  container_name: orion
  expose:
    - "1026"
  depends_on:
    - mongo

volumes:
  pgdata:
```

First of all, to run the application, docker should be initialized. Once docker is running, navigate in the shell (e.g.: cmd) to the path where *docker-compose.yml* is located and execute the command: ***docker-compose up***.

*Note: The command **docker-compose up -d** will execute the app in the background (detached).*

To check if the application has been deployed correctly, check at Docker Desktop the *Containers/App* section:



To stop the application, use the command: ***docker-compose down***.

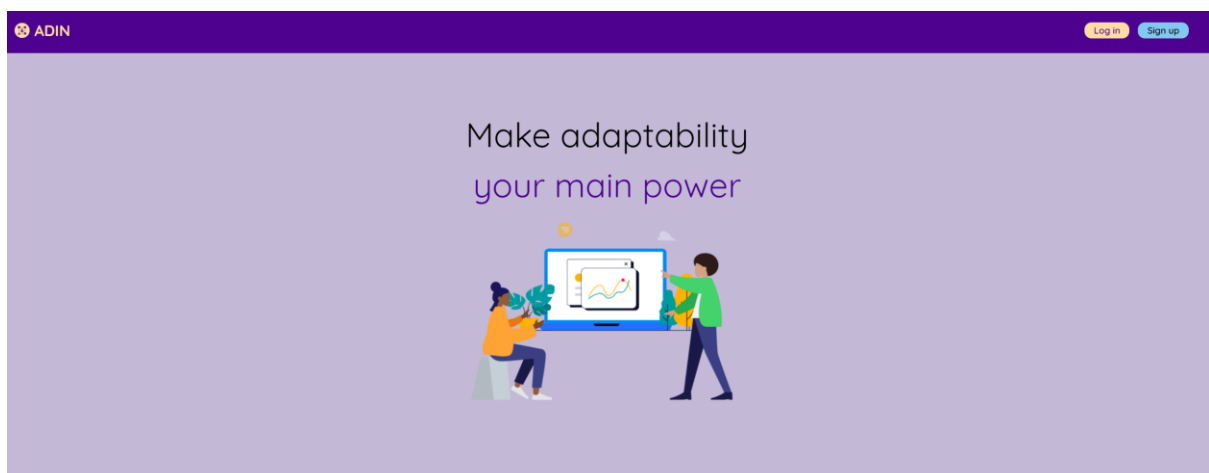
## Application Sections

The application is divided into two main sections: *Home Page* and *Dashboard*.

### Home Page

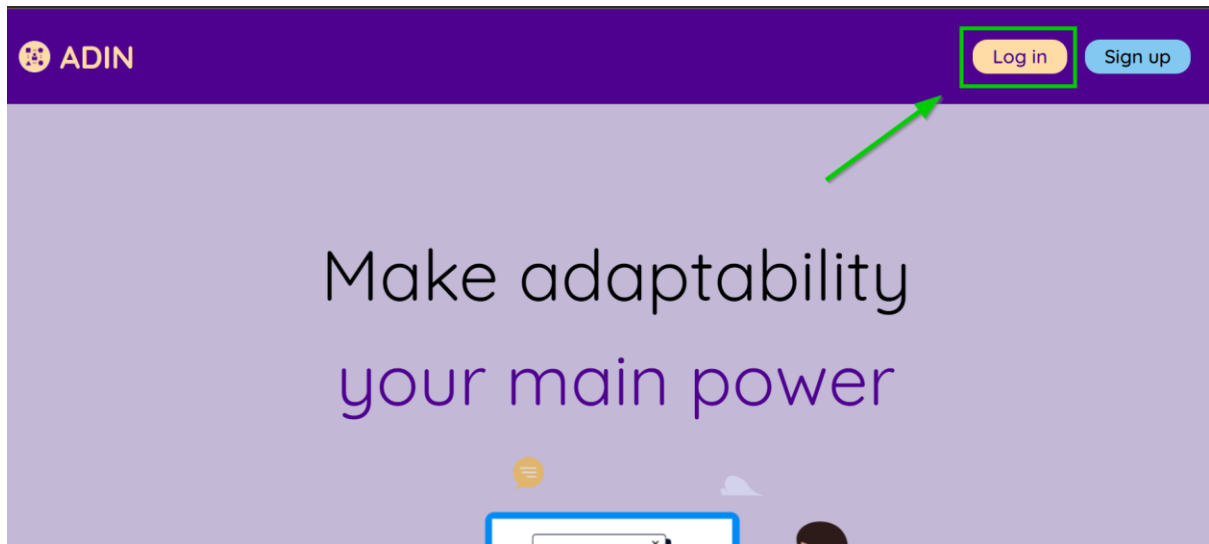
This is the first page of the application where a user can perform to main actions:

- Log in: If the user has already an account, can log in and be redirected to the dashboard.
- Sign up: If the user doesn't have an account, can fill in the necessary data to register into the application.

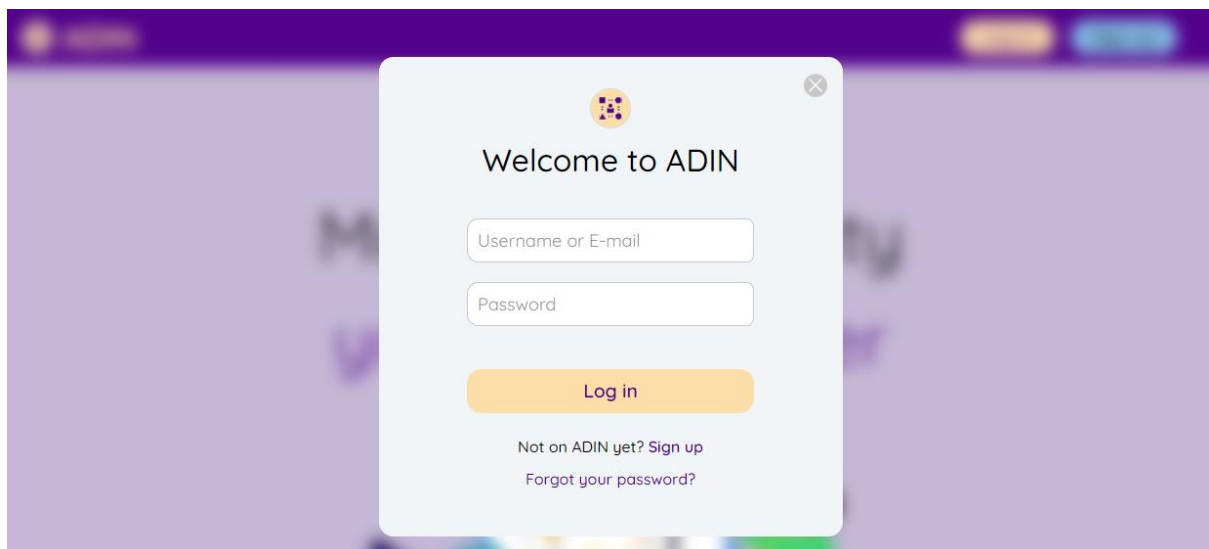


### Log in

To log in to the application the user should press the *log in* button which is located in the upper right section of the window.

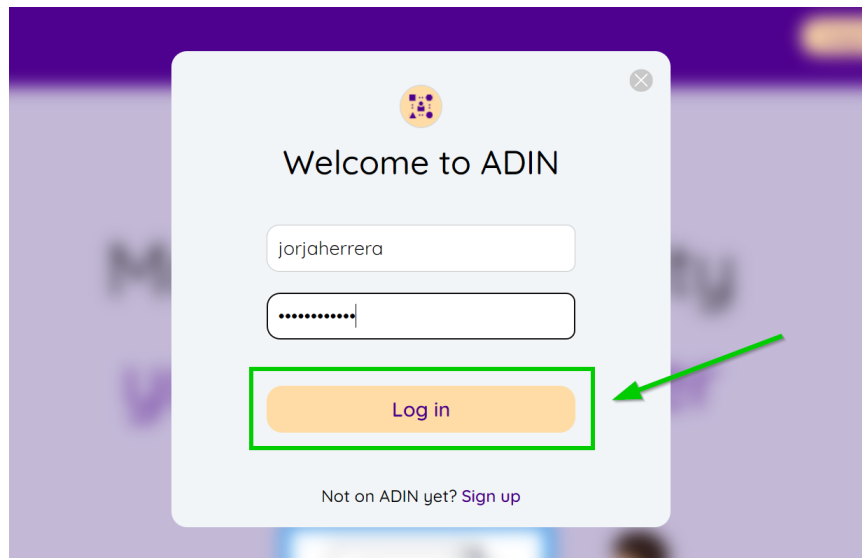


After pressing the button, it will appear a popup window in which the user should fill two fields, username and password.



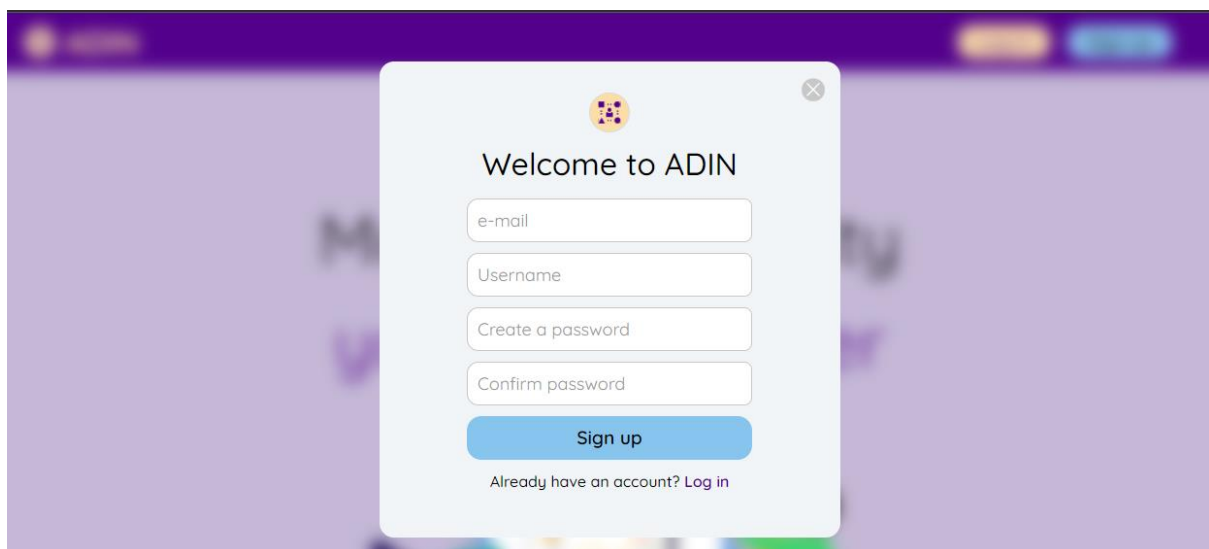
Once the username/e-mail and password have been introduced, by pressing the *Log in* button of the popup window, the user will be redirected to the Dashboard.

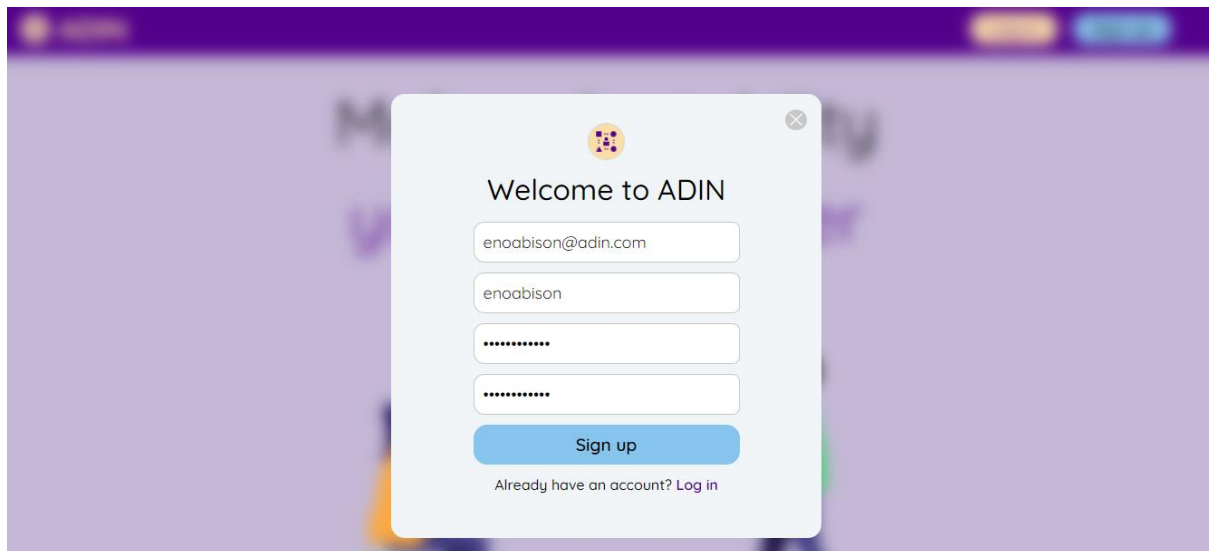




### Sign up

Similar to the login process, in order to sign up in the application, the user should press the *sign up* button located in the top right corner of the screen. A popup window will emerge asking the user to introduce data to generate a profile.



A screenshot of a web application showing a sign-up modal. The modal is titled "Welcome to ADIN" and contains four input fields: email, username, password, and confirm password. The email field is pre-filled with "enoabison@adin.com" and the username field with "enoabison". Below the fields is a blue "Sign up" button and a link "Already have an account? Log in".

Welcome to ADIN

enoabison@adin.com

enoabison

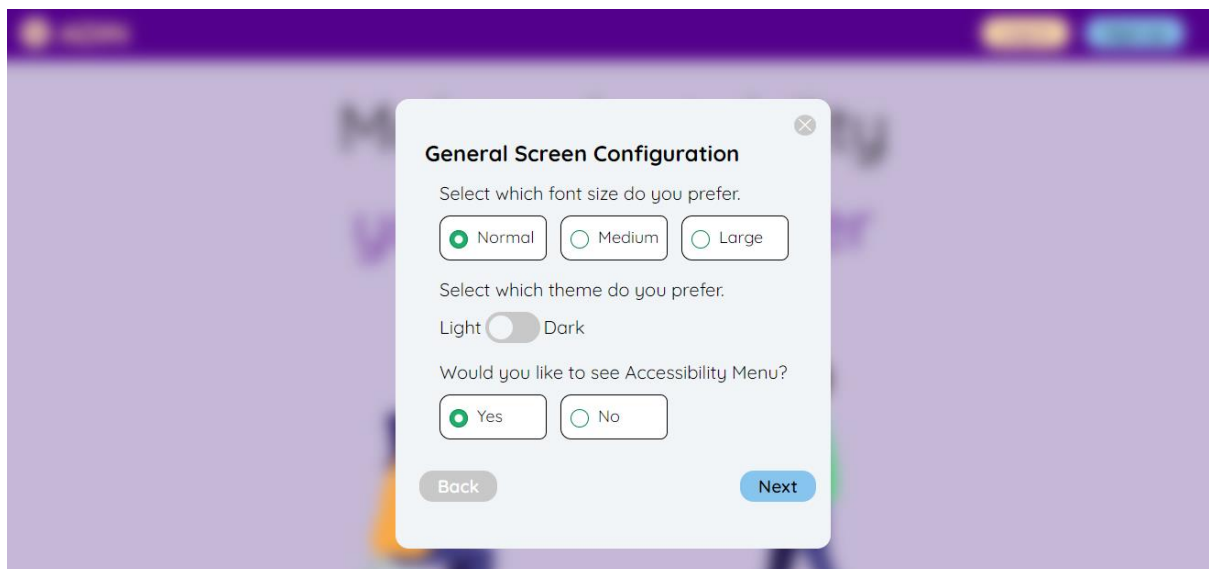
.....

.....

Sign up

Already have an account? [Log in](#)

Once the fields are filled, the user should select the general screen configuration that prefers.

A screenshot of a web application showing a "General Screen Configuration" modal. The modal contains three sections: font size selection (Normal, Medium, Large), theme selection (Light/Dark toggle), and accessibility menu selection (Yes/No). At the bottom are "Back" and "Next" buttons.

General Screen Configuration

Select which font size do you prefer.

☒ Normal ☐ Medium ☐ Large

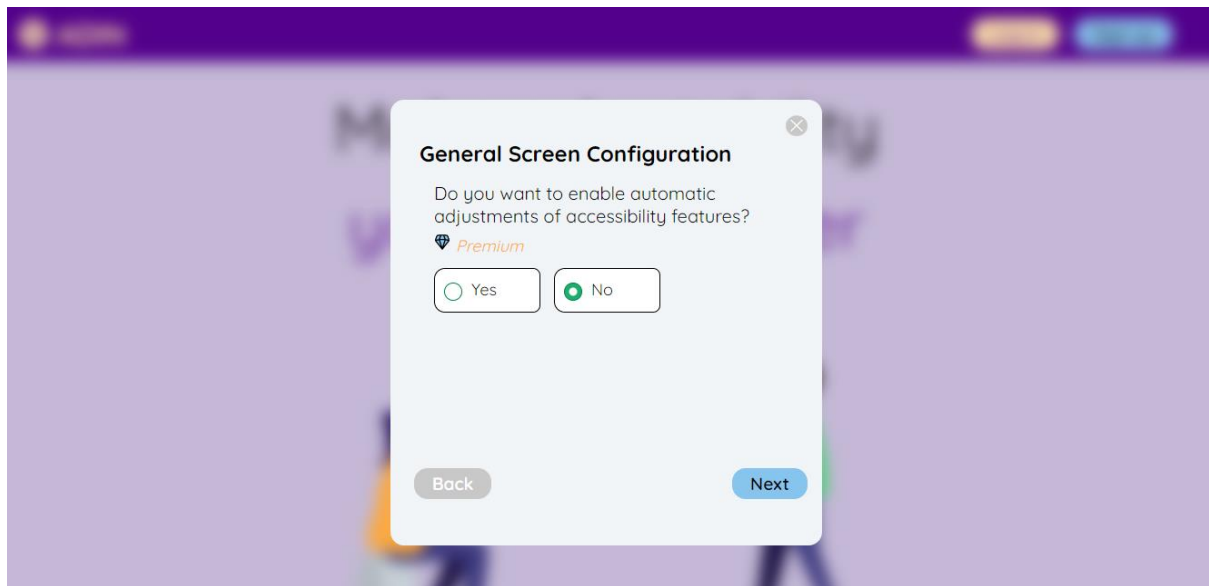
Select which theme do you prefer.

Light ☐ Dark

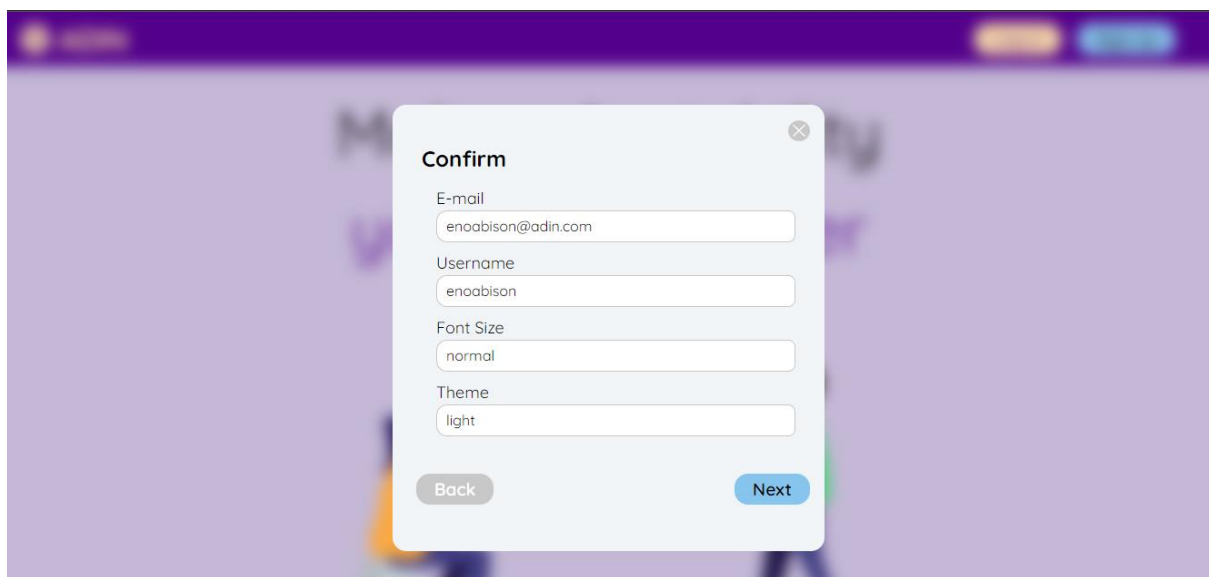
Would you like to see Accessibility Menu?

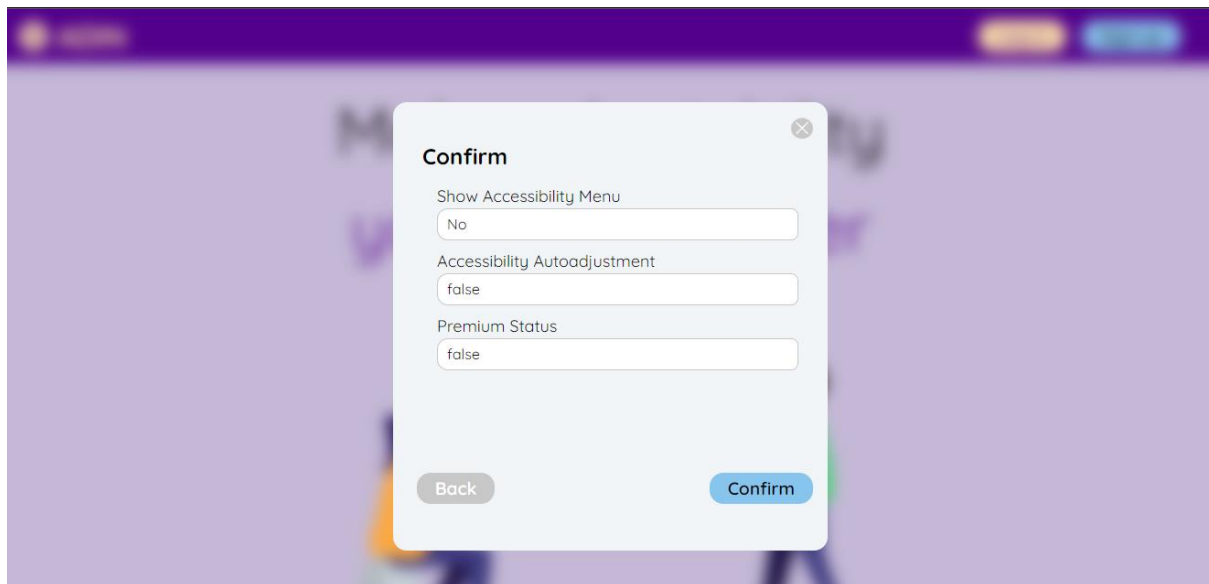
☒ Yes ☐ No

Back Next

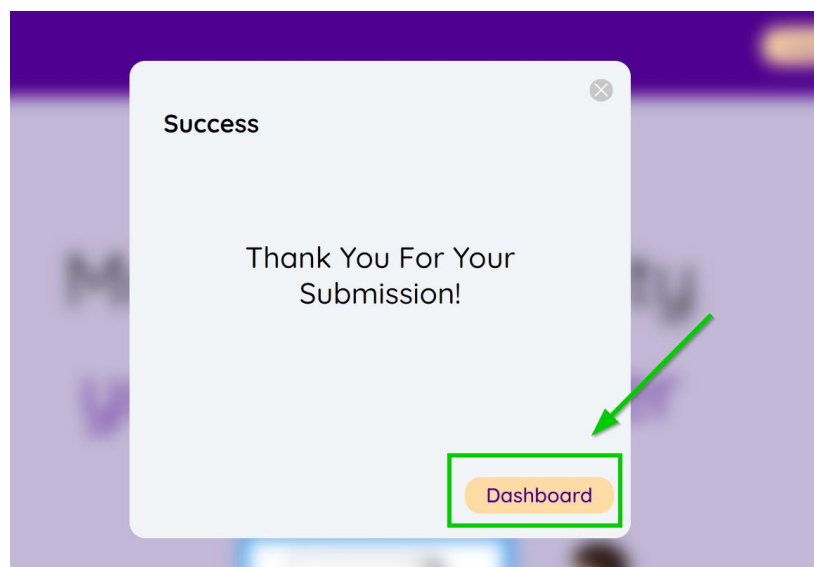


Then the use has can confirm the introduced data. If everything is correct, by pressing the *confirm* button the user will see a message confirming that the registration has been succeeded and a Dashboard button.



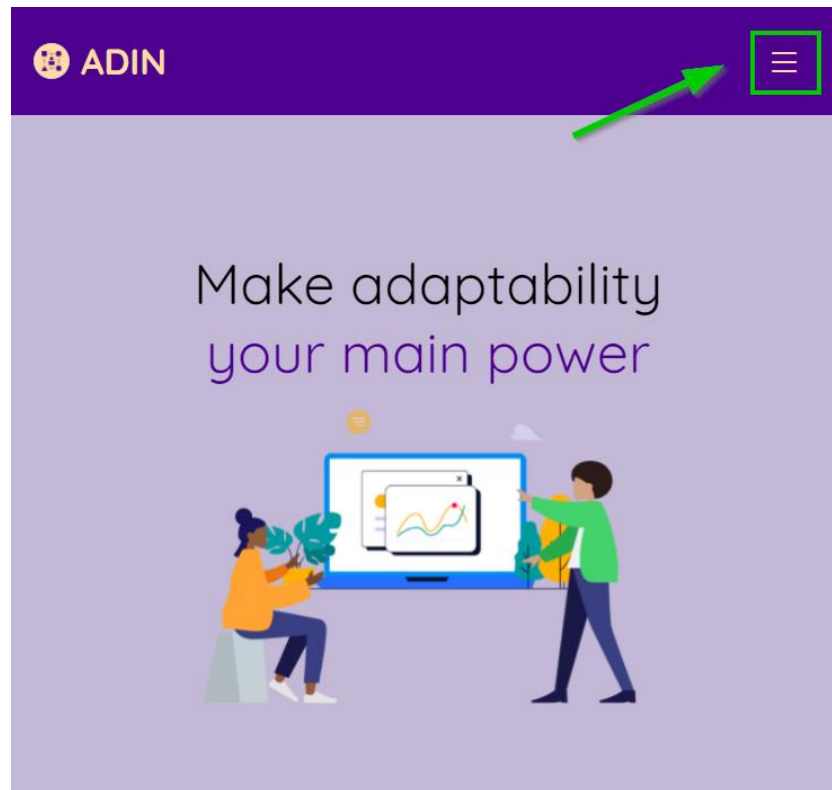


Pressing the *Dashboard* button will lead the user to the Dashboard.

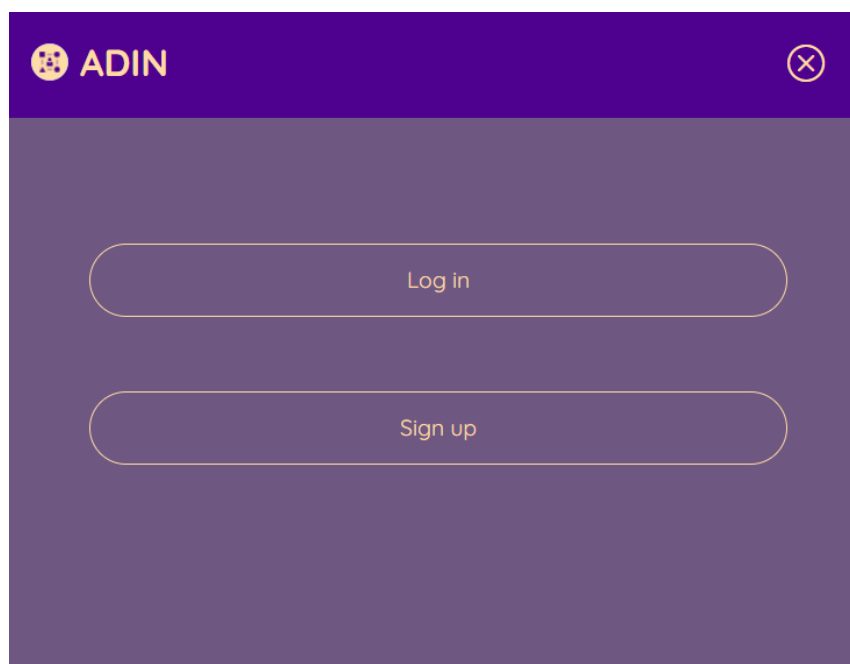


### Small screen

If the screen where the application is being displayed is reduced to a certain size, the items displayed on the navigation bar situated on top of the screen will disappear, being displayed in its place a hamburger menu.



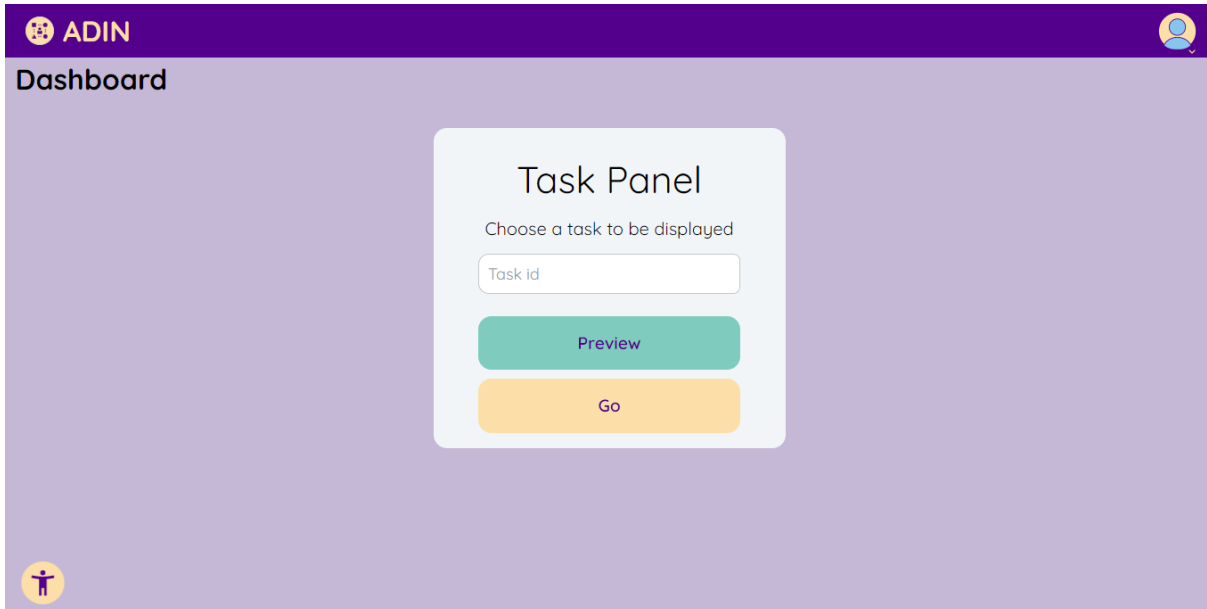
By clicking the hamburger icon, a menu will be displayed on the screen with the same buttons that were contained in the navigation bar. The user can make the same procedures of login and sign up described before.



## Dashboard

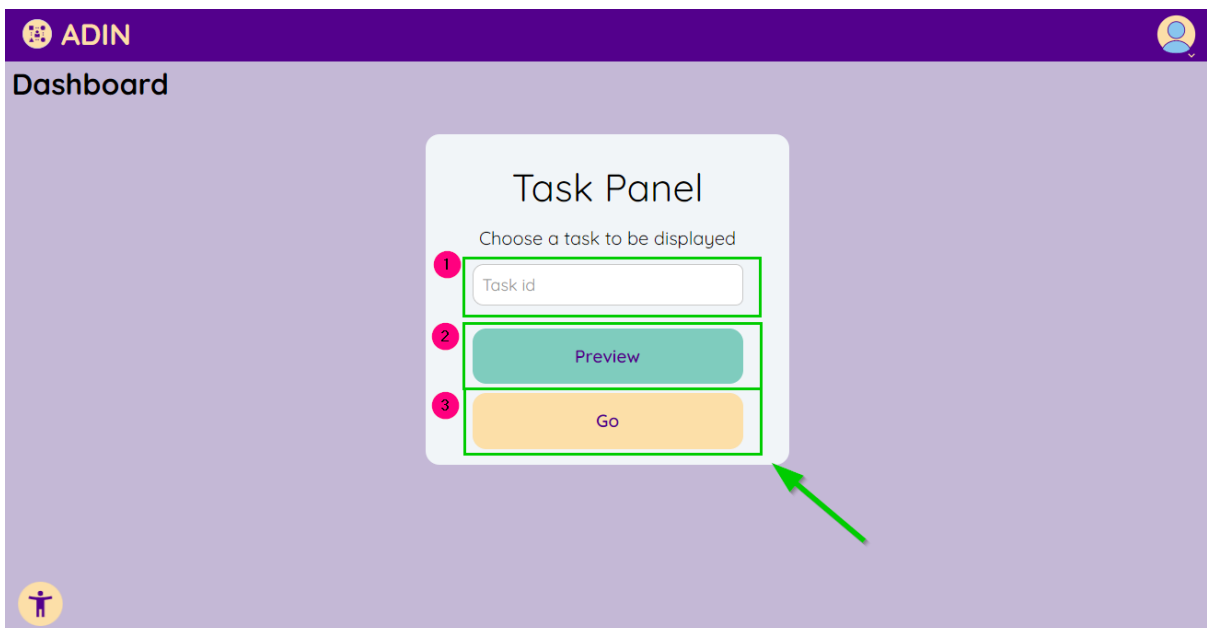
On this page, it is represented the *Task Panel*. The *Task Panel* is meant to assist the user through the development of a task by showing the required steps and information to fulfil without any risk the selected task.

The first element that the user will see is a panel to choose the id of the task to complete.



The screenshot shows the ADIN Dashboard interface. At the top, there is a purple header bar with the ADIN logo on the left and a user profile icon on the right. Below the header, the word "Dashboard" is displayed. In the center of the dashboard is a white "Task Panel" box. Inside this box, the text "Choose a task to be displayed" is followed by a text input field labeled "Task id". Below the input field are two buttons: a teal "Preview" button and an orange "Go" button. A small user profile icon is visible in the bottom left corner of the dashboard area.

By introducing the id number of the task and pressing the *Go* button the application shows the *Task Panel* with the information of the selected task. Additionally, the user can preview the task that has selected by pressing the *Preview* button.



This screenshot is similar to the previous one, but it includes numbered steps and a green arrow. The "Task Panel" box contains the same elements: the text "Choose a task to be displayed", the "Task id" input field, the "Preview" button, and the "Go" button. To the left of the input field is a pink circle with the number "1". To the left of the "Preview" button is a pink circle with the number "2". To the left of the "Go" button is a pink circle with the number "3". A green rectangular box highlights the input field, the "Preview" button, and the "Go" button. A green arrow points from the bottom right towards the "Go" button.

## Dashboard

### Preview Task

Task ID: 1

**Description:** In this task the user will have to assemble a wooden box by fasten all the bolts with the assistance of a cobot. Additionally, the user will wear an EEG headset in order to share their emotional data with the cobot.

**Number of Steps:** 7

**List of Components:** Bolts

**List of Tools:** Screwdriver

**List of Safety Tools:** Gloves, Helmet

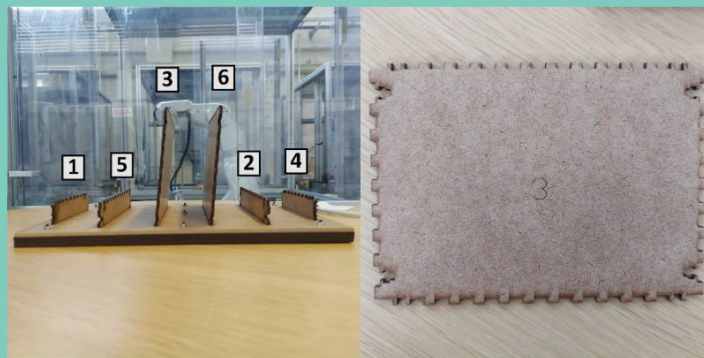
Next



## Dashboard

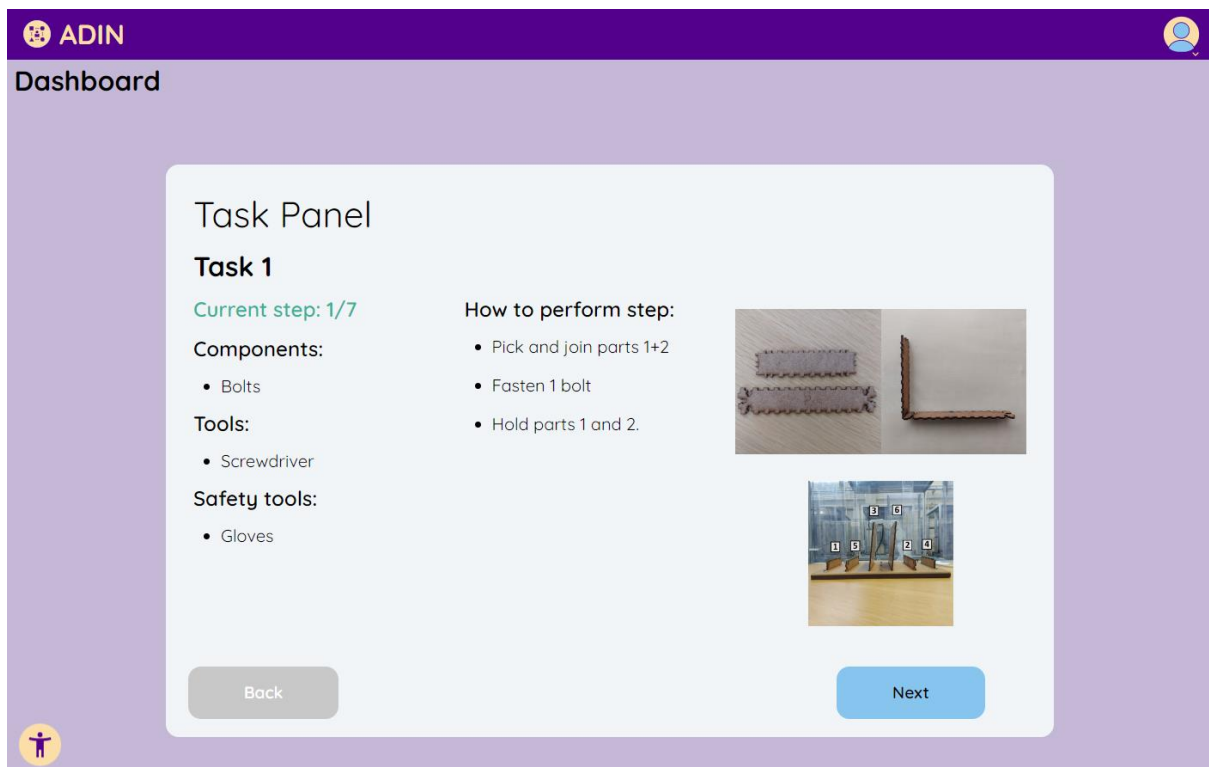
### Preview Task

Visual Information:

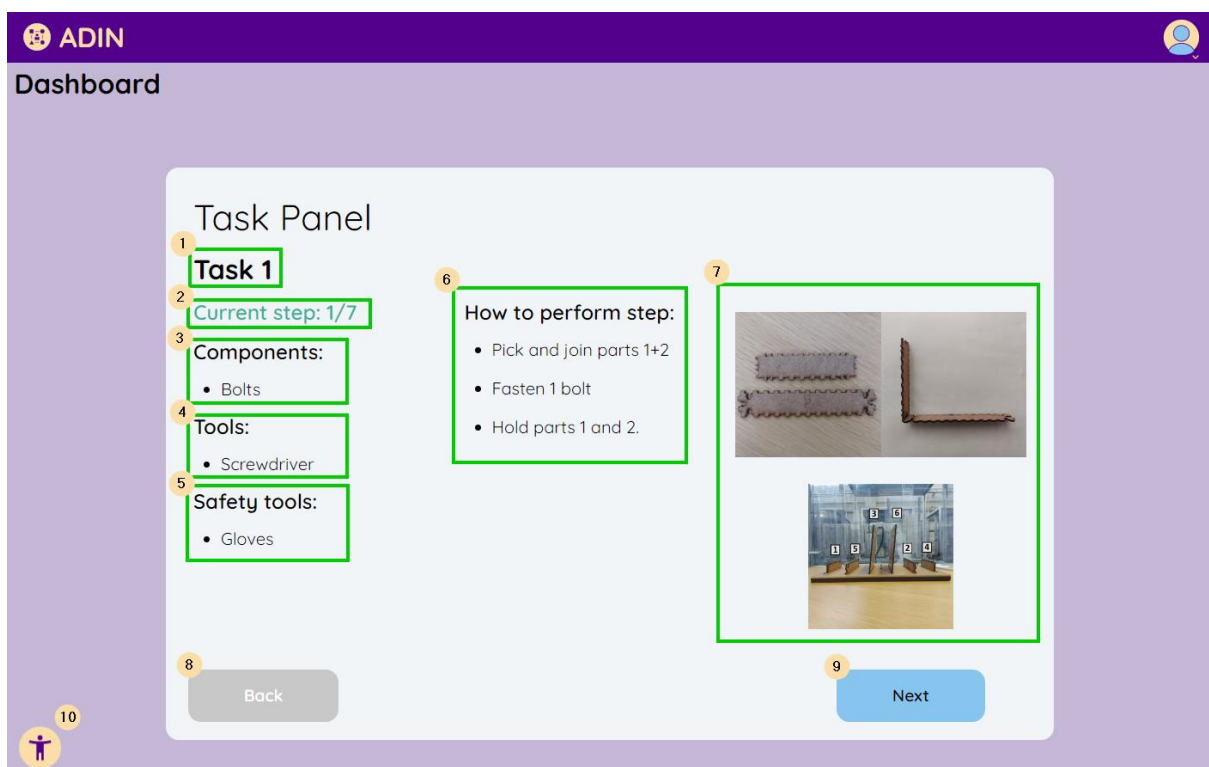


Back





In the following picture, it can be seen all the elements that form the Task Panel.



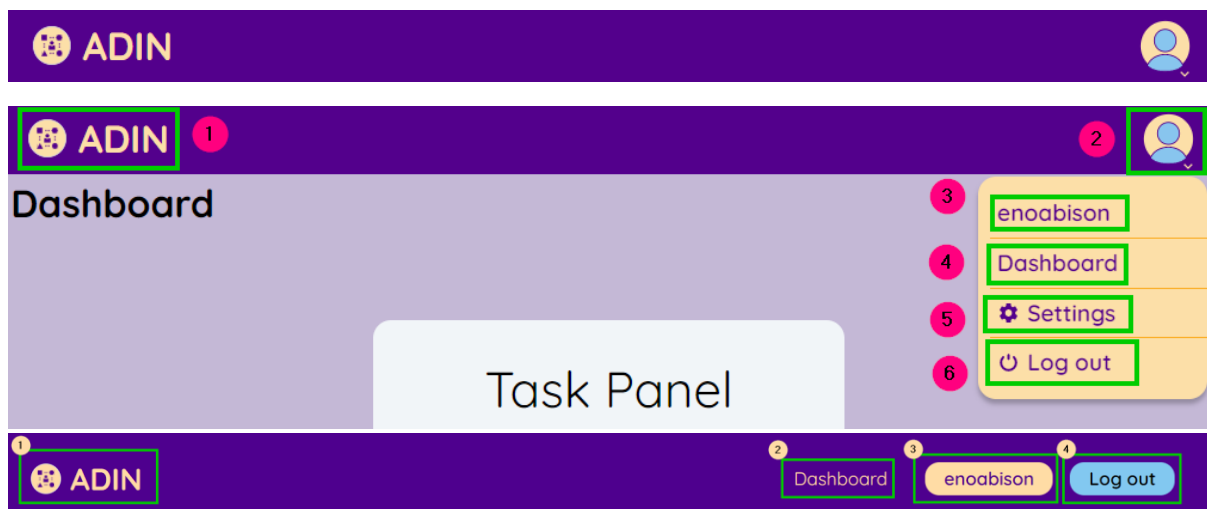
1. Name of the selected task.
2. Shows the current step that the user is visualizing and how many steps are.



3. Necessary components to perform the step.
4. Needed tools to complete the step.
5. Required safety tools to perform the step.
6. Instructions on how to perform the step.
7. Space reserved for pictures related to the step if needed.
8. Button to go see the previous step.
9. Button to proceed to the next step.
10. Accessibility Menu

Once the user is in the last step of the task, by pressing the next button the user is redirected to the first window to select a new task to display.

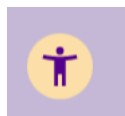
After logging in, the navigation bar situated on top of the screen shows some different items than before.



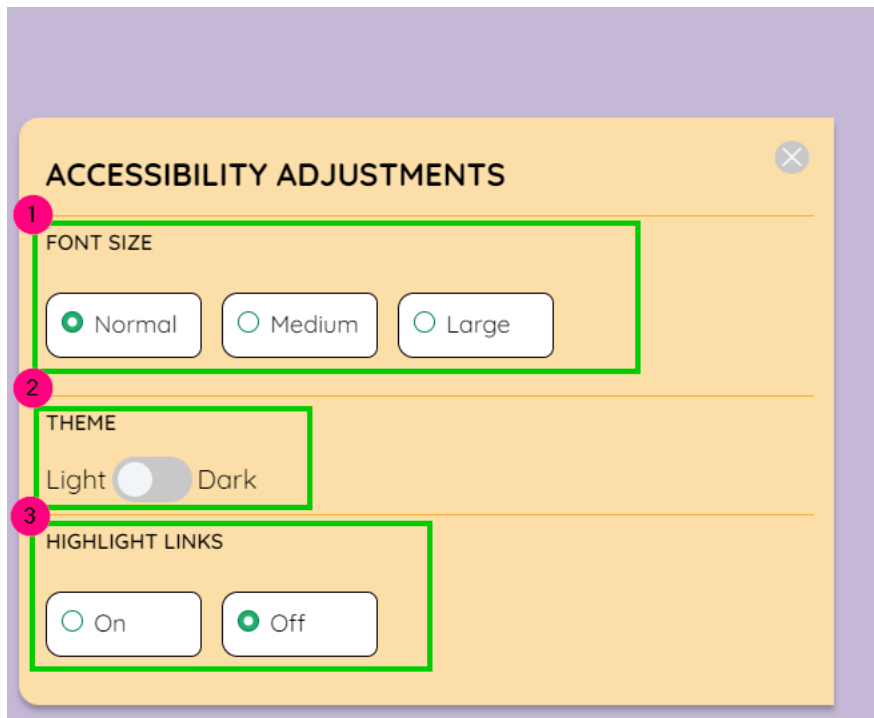
1. ADIN Logo: By pressing the logo the user will be redirected to the home page.
2. User Logged Menu: By pressing the icon, it opens a menu with several options.

### Accessibility Menu

The *Accessibility Menu* is opened by pressing the accessibility icon placed in the bottom left corner.

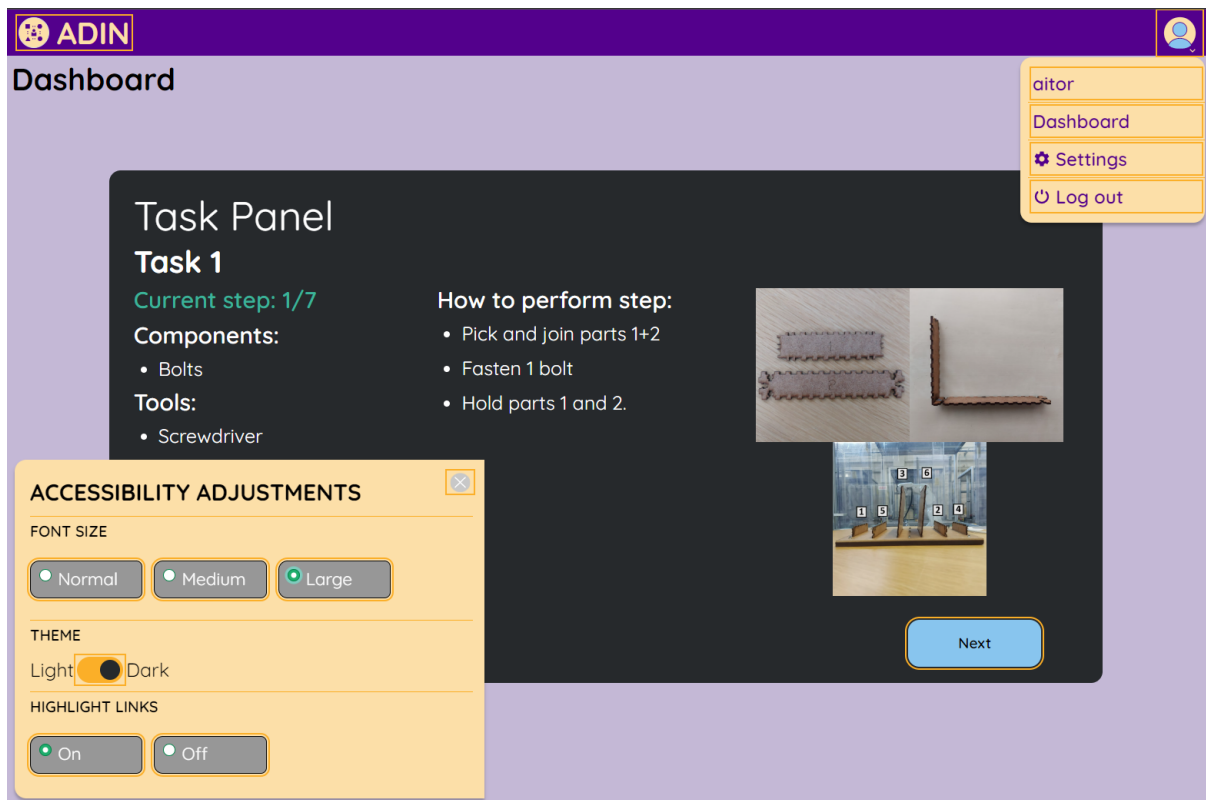


When the users clicks the button, it can be seen several options that the user can modify in order to adapt the application to their personal preference, making it more comfortable to use it whenever they want.



1. Font size: The user can increase or decrease the font size of the application.
2. Theme: It changes the theme of the application. There are two options, light or dark theme.
3. Highlight links: By activating this options, all the components of the application that are clickable will have a square surrounding them in order to facilitate the understanding of the elements.

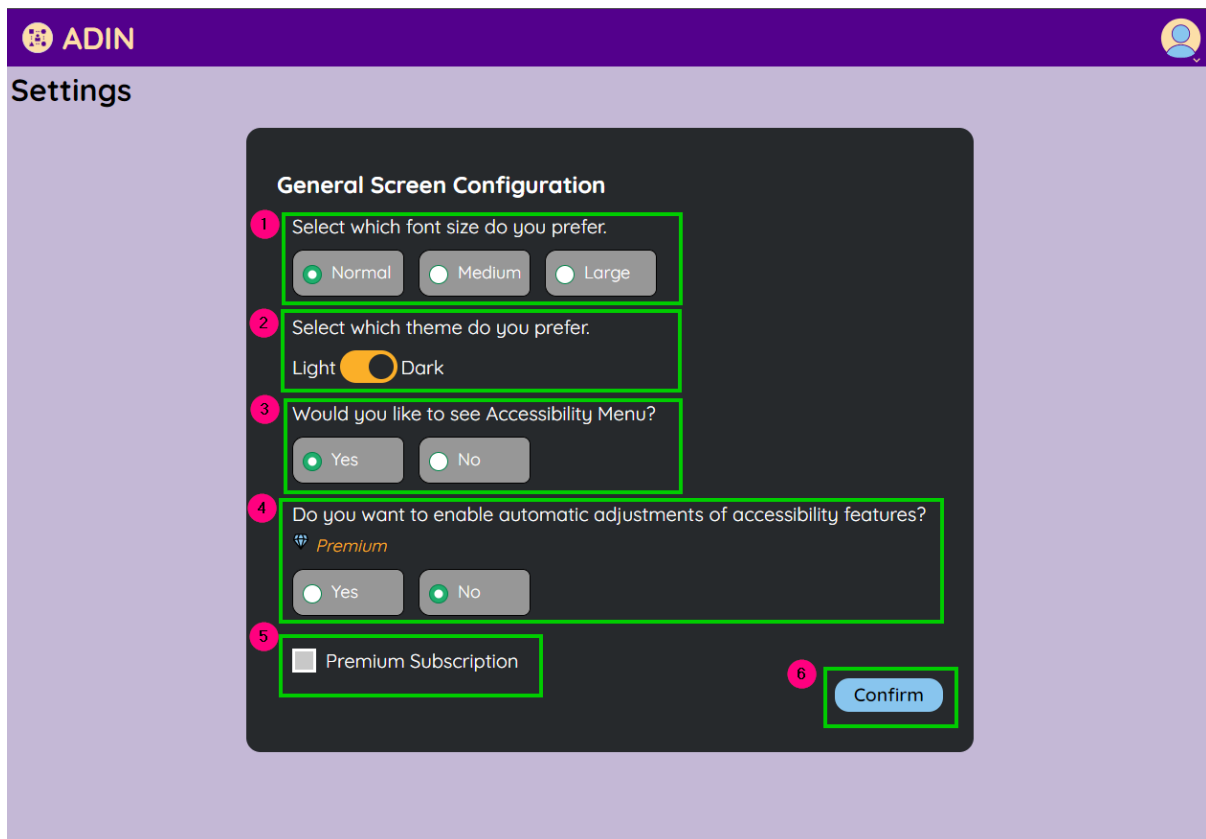
An example of the previous options selected is shown in the following picture. The selected options are *Large* font size, *Dark* theme and *Highlight links* on.



3. Username: This represents the user that has initiated a session in the ADIN application.
4. Dashboard: If the user press this link the application will load the Dashboard page. If the user is already at the Dashboard, it doesn't have any action.
5. Settings: This button opens the Settings page were the use can update the configuration of the application.
6. Log out: Pressing this button will close the session of the user and be automatically redirected to the homepage.

## Settings

In the settings panel, the user can update the configuration of the application and save it in order to have it ready for the next time that logs in.



**ADIN**

### Settings

**General Screen Configuration**

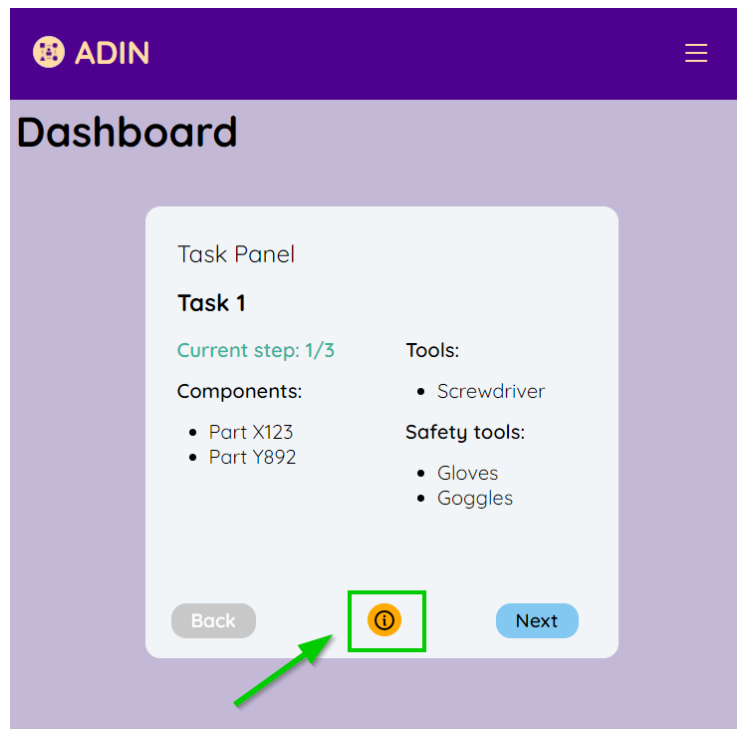
1. Select which font size do you prefer.  
☒ Normal ☐ Medium ☐ Large
2. Select which theme do you prefer.  
 Light ☒ Dark
3. Would you like to see Accessibility Menu?  
☒ Yes ☐ No
4. Do you want to enable automatic adjustments of accessibility features?  
 Premium  
☐ Yes ☒ No
5. ☐ Premium Subscription
6.

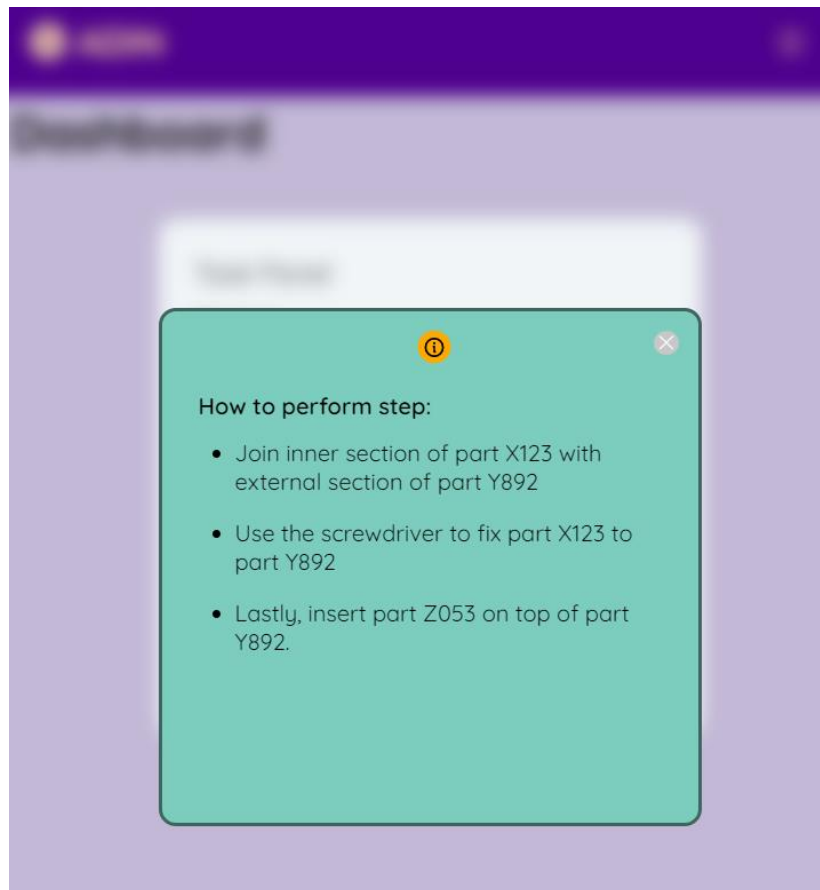
1. Font size: The user can increase or decrease the font size of the application.
2. Theme: It changes the theme of the application. There are two options, light or dark theme.
3. Accessibility Menu Visibility: It enables or disables the visibility of the Accesibility Menu.
4. Auto adjustments of Accessibility Features: It enables the automatic adjustment of the accessibility features. This feature can only be activated if the user is premium.
5. Premium Subscription: It subscribes the user as premium or not. Allowing or denying the possibility of having premium features.
6. Confirm: Button to save the selected configuration.

### Small screen

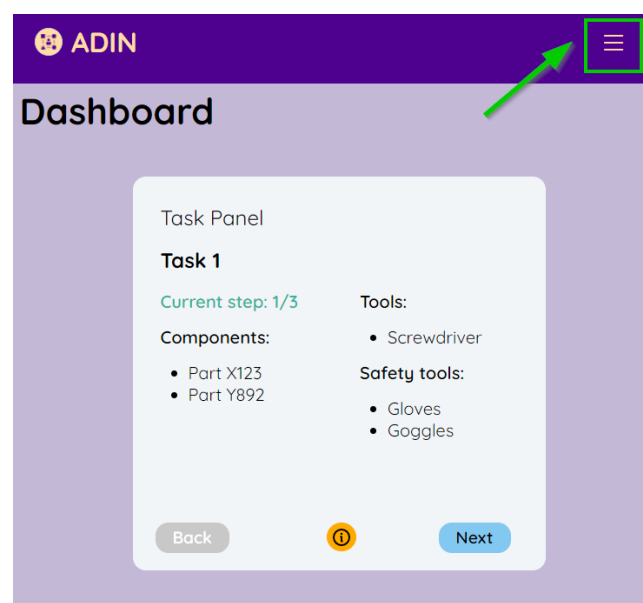
If the screen is reduced, the *Task Panel* will be displayed in another way to fit better the size of the window where the application is being rendered. In the next picture is possible to see the reduced version of the *Task Panel*.

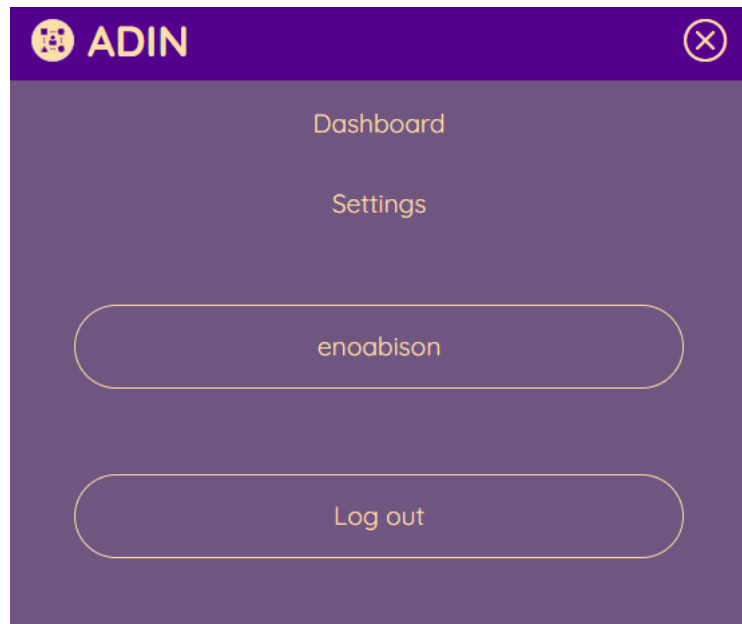
The information shown in the panel is the same but without the additional images. In order to see the information about how to perform the steps, the user should click the info button and a new window will pop up with the information.





Also, by clicking the hamburger button on the top right corner of the screen, the user would be able to see the items that were displayed before on the navigation bar. The buttons have the same actions that were described before.





## How to structure the database

ADIN shows the information on several tasks in order to assist a user to fulfil them properly. That data is stored in a structured database. This database contains information related to the users, tasks, steps, components, tools and safety tools.

Regarding the data of the user, it is automatically saved after registration. On the other hand, data of the tasks should be introduced manually in the database. It is very important to follow the proper structure of the data so it can be displayed in the application as is meant to be.

Focusing on the creation of tasks is necessary to explain the models, relations among models and tables.

### Models

Task:

- task\_number: Number of the task. Should be the same as its id.
- description: Task description.

Step:

- task: ForeignKey related to the correspondent task of the step.
- step\_number: Number of the step
- instructions: Instructions on how to perform the step.
- components: List of components to use in the step of the assembly.
- tools: List of tools to use in the step of the assembly,
- safety\_tools: List of safety tools to use in the step of the assembly.
- images: List of images to show in the step of the assembly.

Component:

- component\_name: Name of the component.

Tool:

- tool\_name: Name of the tool.

SafetyTool:

- safety\_tool\_name: Name of the safety tool.

GeneralScreenConfig:

- user: ForeignKey related to the user that has this configuration.
- font\_size: Defines the size of the font to be displayed.
- accessibility: Flag marking if user wants to display accessibility features.
- auto\_adjust: Flag marking if user wants auto adjustment of accessibility features.
- is\_premium: Flag marking if a user is premium or not.

## **Relations**

Task - Step: One to Many

Step - Components: Many to Many

Step - Tools: Many to Many

Step - Safety Tools: Many to Many

Step – Images: Many to Many

User – GeneralScreenConfiguration: One to One

## **Tables**

api\_task:

- id: integer (Primary Key(PK))
- task\_number: integer
- description: varchar(1000)

api\_step:

- id: integer
- instructions: varchar(1000)
- step\_number: integer
- task\_id: bigint (Foreign Key(FK))

api\_component:

- id: integer (Primary Key(PK))
- component\_name: varchar(50)

api\_tool:

- id: integer (Primary Key(PK))
- tool\_name: varchar(50)

api\_safety\_tool:

- id: integer (Primary Key(PK))
- safety\_tool\_name: varchar(50)

api\_step\_components:

- id: integer (PK)
- step\_id: bigint (FK)
- component\_id (FK)

api\_step\_tools:

- id: integer (PK)
- step\_id: bigint (FK)
- safetytool\_id (FK)

api\_step\_safety\_tools:

- id: integer (PK)
- step\_id: bigint (FK)
- tool\_id (FK)

api\_step\_images:

- id: integer (PK)
- step\_id: bigint (FK)
- image\_id (FK)

### Creation of steps

The tasks and steps information should be as brief and direct as possible. A great number of instructions, components, tools or steps within a task could lead to confusion of the users. As consequence, a reduction of the difficulty of the steps is advisable to have a clear view of the information.

Additionally, for a good representation of the instructions on how to perform the task, these should be separated by points. Example:

*Join inner section of part X123 with external section of part Y892. Use the screwdriver to fix part X123 to part Y892. Lastly, insert part Z053 on top of part Y892.*

id	instructions	step_number	task_id
1	1 Join inner section of part X123 with external section of part Y892. Use the screwdriver to fix part X123 to part ...	1	1

The representation will be as follows:

#### How to perform step:

- Join inner section of part X123 with external section of part Y892
- Use the screwdriver to fix part X123 to part Y892
- Lastly, insert part Z053 on top of part Y892.