HARMONY Model Suite Manual

Detailed instruction manual for the running of various simulations

HARMONY H2020 PROJECT | 2023

Index

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement Nº815269. HARMONY is a project under the CIVITAS Initiative, an EU-funded programme working to make sustainable and smart mobility a reality for all.

HARMONY Model Suite Manual

This is the Online Manual of the HARMONY Model Suite (MS for short). It explains the main concepts of the platform and how an end user can use the platform. It also includes an Appendix that describes (i) how the platform can be extended with new components and (ii) how one can install the platform and create new users (useful for the maintainers of the platform).

1. Terminology and Overview

The MS targets two end users - Transport Modelers (**Modelers** for short) and Transport Planners (**Planners** for short). It also assumes two supporting roles, that of Component Developers (**Developers** for short) and of platform **Maintainers**.

There are five important concepts in the MS:

1. **Modeling Components**
2. **Modeling Templates**
3. **Projects**
4. **Scenarios**
5. **Dashboard**

The building blocks of the **HARMONY MS** are the **Modelling Components**, i.e. the stand- alone models created by the **developers**. Selecting one modelling component or combining two or more, the **modellers** can create a **Template** i.e., a modelling application composed by one or more stand-alone models. Once the template is created, a specific **Project** can be defined by uploading the required input files.

In this phase, also the **Planner** can use the HARMONY MS to create **projects** and **scenarios**. By default, the first set of input files is used automatically by the platform to create the Default Scenario.

Within the same project, it is possible to create different **scenarios** by uploading different input files or defining different parameters. Finally, it is possible to navigate among the project KPIs, to analyse the impacts and compare the results of two or
more scenarios through the HARMONY MS **Dashboard**.

In the figure below, a graphical representation of the HARMONY MS structure is shown.

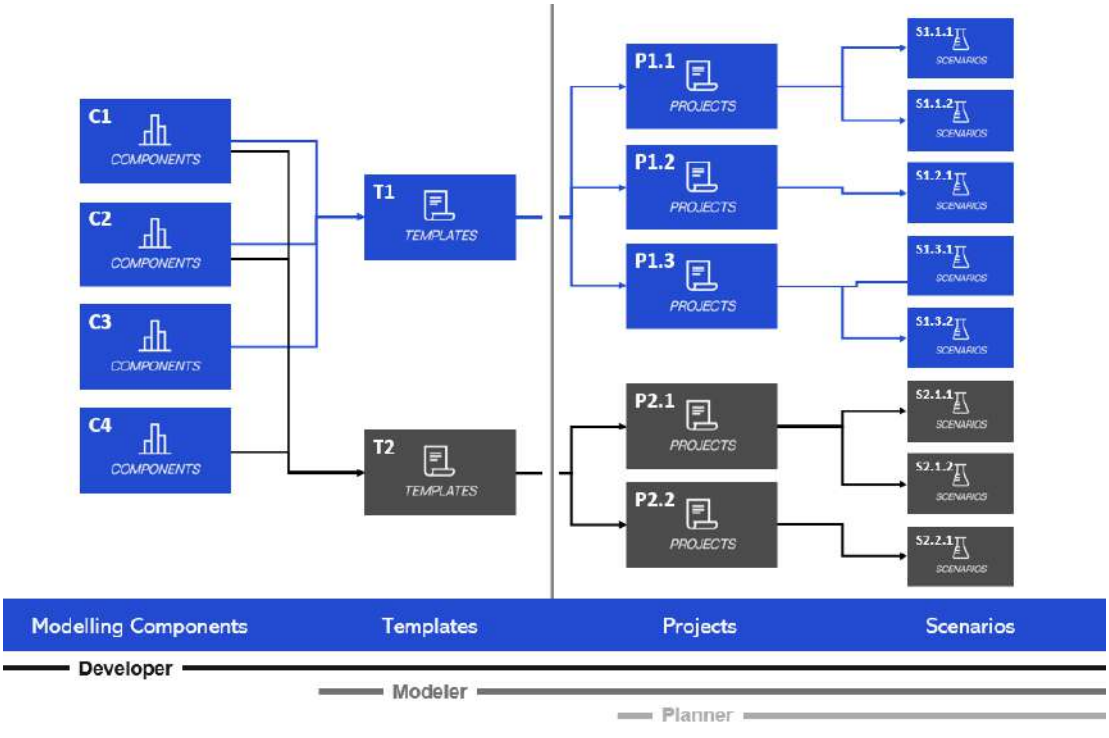

Fig.1 - HARMONY MS Structure

A more detailed definition of the above roles and main concepts is available in the *MS Glossary.*

2. Walkthrough of Main Functionalities

A walkthrough of the main functionalities (and corresponding screens) of the MS is described in the sections below. If you want to try out the examples, the mentioned filescan be found

here.

The first step as an end user (Modeler or Planner) is to **login** to the platform (Fig. 2). For this, the user needs to navigate to the URL of the platform and use the username and password provided to them by the platform maintainers. The User Type option needs tobe "User" (preselected) and not "Admin".

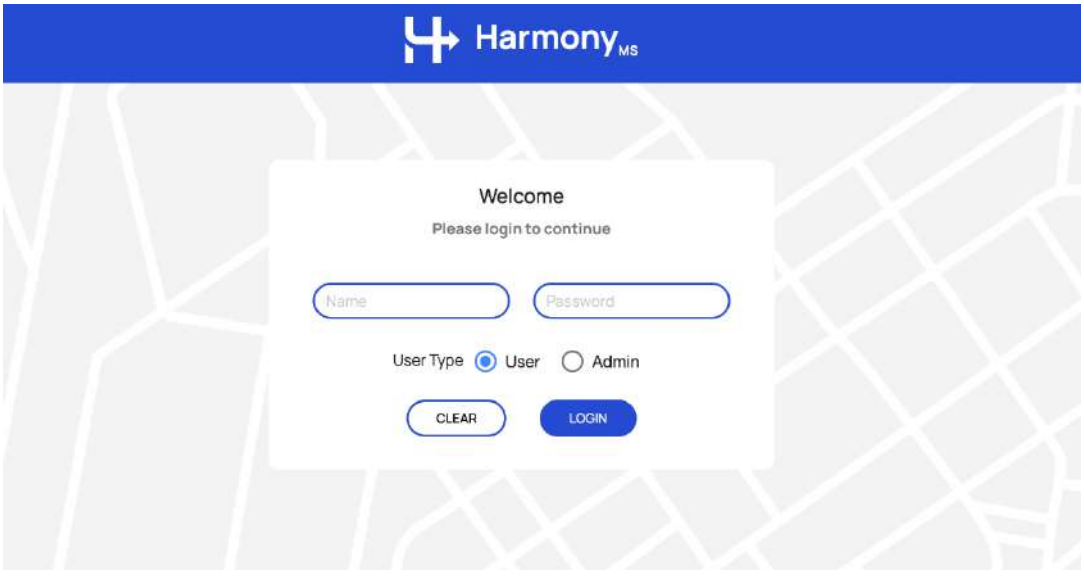

Fig.2 - MS login page

When a user successfully logs in to the platform, they see the MS home page (Fig. 3). This includes shortcuts to the four main screens of the MS: Modeling Components, Modeling Templates, Projects, Scenarios. They can be accessed by clicking on the corresponding name or small figure above the name. Clicking on the "Run Scenario" or "View Scenarios" button brings the user to the Scenarios page.

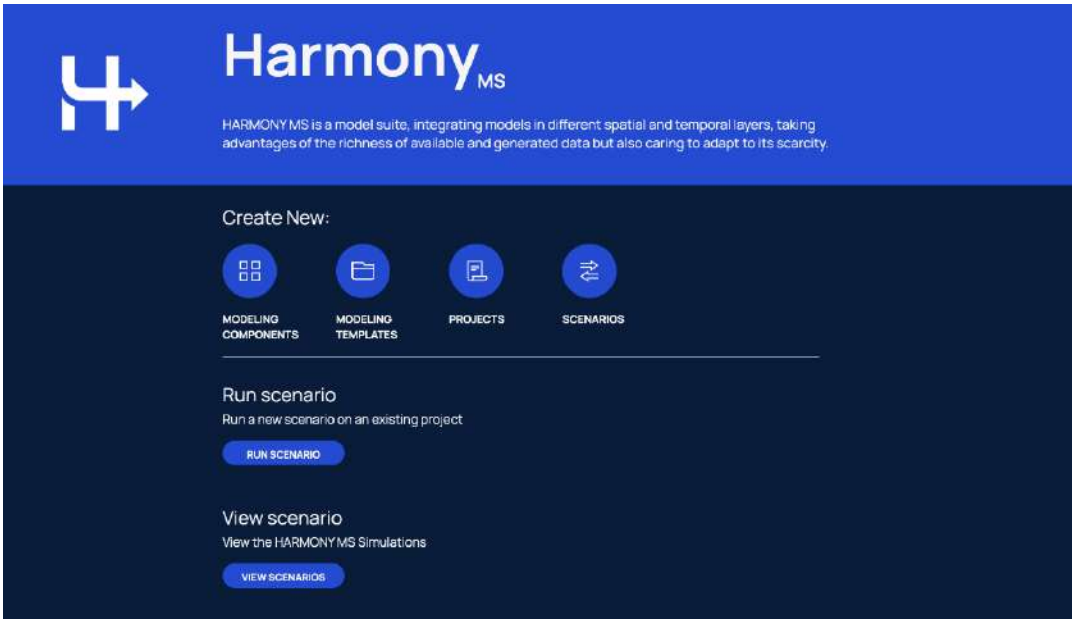

Fig.3 - MS home page

For instance, if the user clicks on the "Modeling Components" in the home page, they are redirected to the corresponding page, which will look like the one of Fig. 4. Note that depending on the components that are so far created or uploaded, different components would appear in the list. However, each of the main screens (Modeling Components, Modeling Templates, Projects, Scenarios) has the following in common:

1. The **logo** of the MS appears at the top left of the screen - [1] in Fig. 4.
2. The **active screen** is highlighted in blue background in the left side-menu - [2].
3. The **name** of the logged-in user appears at the bottom left - [3].
4. Links to the *MS Glossary* and *MS Online Manual* appear directly below the user name at the bottom left - [4].
5. **Copyright** information appears at the bottom of the screen - [5].
6. **Links** to the MS **About** page and its **Terms and Conditions** page appear at the bottom right, along with a **log out** button - [6]. Clicking on the log out button terminates the current user session and brings the user to the MS login screen.

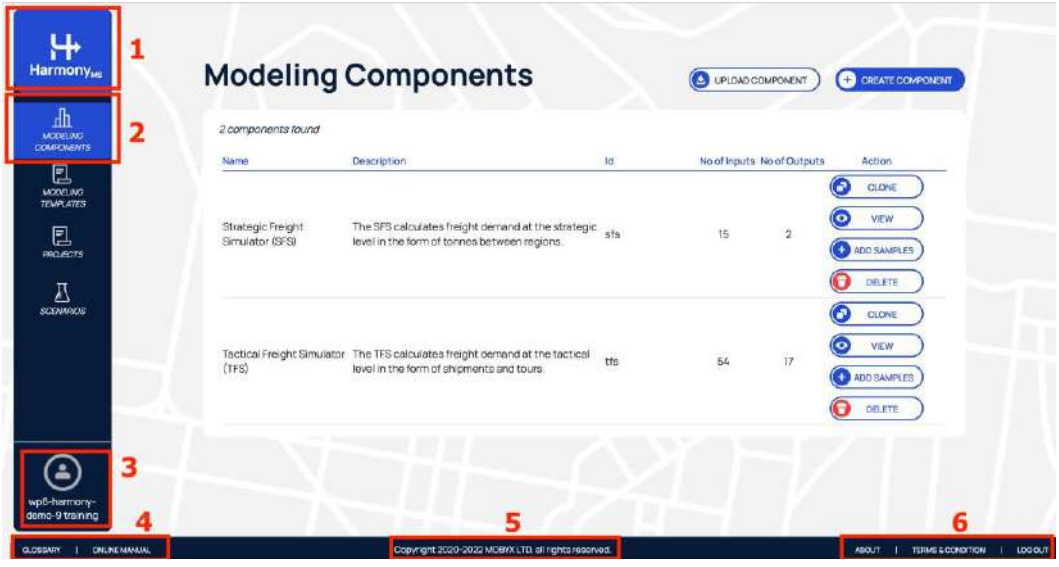

Fig. 4 - Example of a Modeling Components page

The following sections A, B, C, D, and E explain in turn the main functionalities of the MS,which are:

A. *Include modeling components* (done by Modelers)

B. *Create modeling templates* (done by Modelers)

C. *Create projects* (done by Modelers and Planners)

D. *Run scenarios* (done by Modelers and Planners)

E. *View and compare scenario results* (done by Modelers and Planners using the MSDashboard)

A. Include a Modelling Component in the MS

A **Modelling Component** is a stand-alone model, developed exogenously with respect tothe HARMONY MS and requiring a set of inputs in order to run and produce outputs indicators. To include a Modelling Component in the MS, one has to upload it and specify which are the inputs and which are the outputs files.

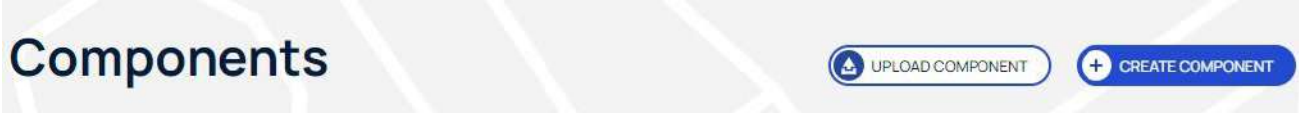

Fig.5 - Modelling Components Upload and Create options

There are two ways to do it:

- **Using a specification file**. The **HARMONY MS** is able to upload the Modelling Components specifications reading from a `json` file. The developers have already created the specification files for the models developed in the HARMONY project. Therefore, moving to the tab *Components* of the HARMONY MS, one can click on the button *Upload Component* and select the related `json` file. It is a common practice to name these files as `modelname_components.json`.

- **Manually specifying inputs and outputs files**. In the tab ***Components***, one can choose to use the button ***Create Component***. This button leads to the *Create Component* page, in which it is possible to add name, description and model ID of the Modelling Component that is under creation. The user also needs to specify theinputs and outputs files of the Modelling Component.

Fig.6 - Fields to be filled in when creating the component

In both cases, in the ***Components*** tab, beside the component name, it is possible to **view**the component' description page by clicking on the button *view*. The aim of the component page is to show the component structure in terms of input and output, bothas a list of files and in a schematic visual representation.

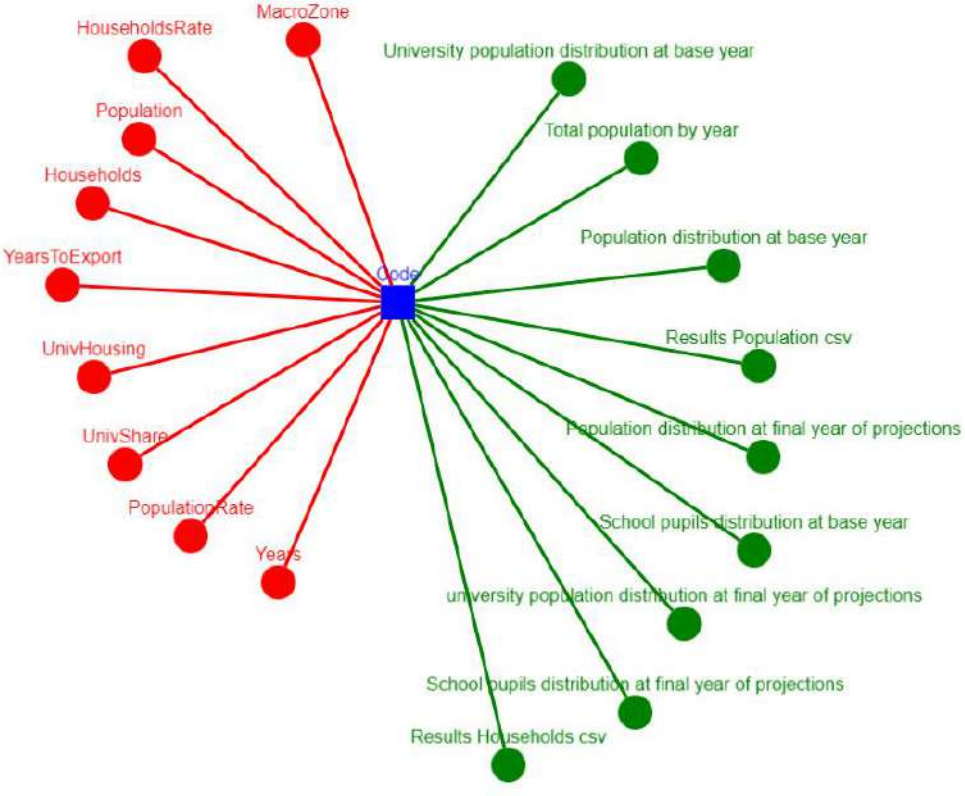

Fig.7 - Visual representation of a Modelling Component (Inputs highlighted in red, Outputs highlighted in green)

B. Build a Modelling Template

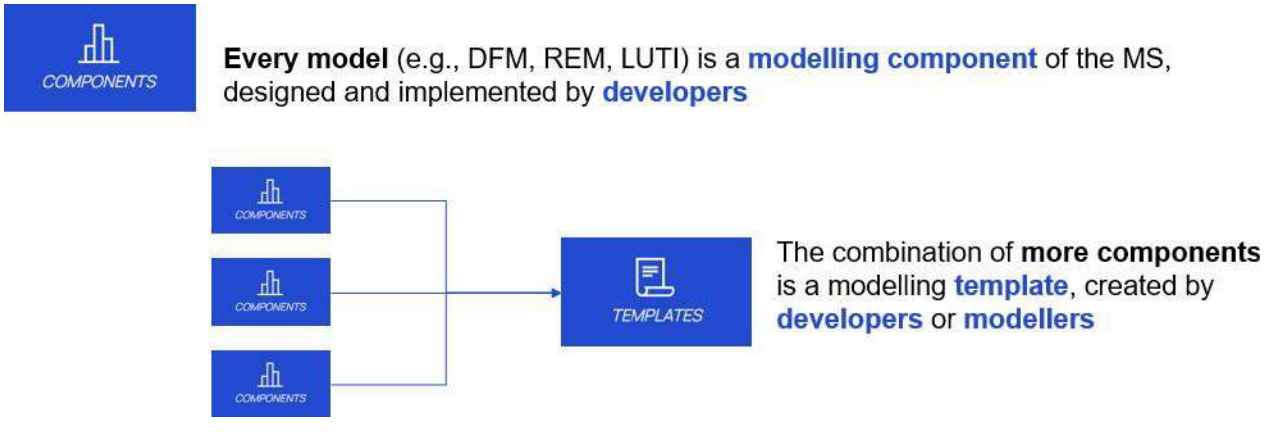

Fig.8 - Template created as combination of one or more Modelling

Once one or more Modelling Components are available in the MS, it is possible to combine them to create a **Modelling Template**. In case of a single component, the template is simply replicating its structure. Otherwise, the aim of this process is to specify the linkages between two or more modelling components, e.g. the *Demographic Forecasting Model* feeding the *Regional Economy Model* with the variable `Total Population by Year` which is calculated in a specific output file and read in a specific inputfile.

Fig.9 - Templates Upload and Create options

There are two ways to do it:

- **Using a specification file**. The HARMONY MS is able to upload the Template specifications reading from a `json` file. The developers have already created the specification files for a list of templates implemented in the HARMONY project, in order to avoid errors and to speed up the procedure. Thus, moving to the tab ***Templates*** and clicking on the button ***Upload Template***, one can select
  the json file of the related template and upload it in the **HARMONY MS.**
- **Manually specifying the structure of the template**. By clicking on the button ***Create Template***, one can manually add the Modelling Component(s) to use and specify the linkages between each other through the input and output variables (and files). First of all, it is required to specify the name of the **Template**, the description and the Template ID. Then, one can specify the Modelling Component(s) to be included in the Template: it could be a single Modelling Component or two or more. This is defined by specifying the number of tasks and using the `add(+)` button to include the related Modelling

components. As an example, if we consider a **Template** with the DFM (Demographic Forecasting Model) that feeds the REM (Regional Economy Model), thefollowing steps are needed:

- click on the `add(+)` button to specify the order of the Modelling components: i.e., Task 1 is the DFM model (by selecting from the menu) and the next step is represented by Task (Modelling component) 2.
- click again on the `add(+)` button, to specify that Task 2 is the REM model (byselecting from the menu)
- within the same window, add a ***generated input***, by clicking on thebutton `add(+)` beside the label *generated input*
  - to specify which is the feeder Modelling Component, i.e. the generator task(task 1, which is the DFM in this example) from the menu
  - to specify the name of the output variable of the first modelling component(`PopYearTot`)
  - to specify the name of the input variable (receiving input) for the current modelling component, which is the `Total Population by Year` in this example (asdefined in the REM)
  - to specify that the DFM is the feeder model.

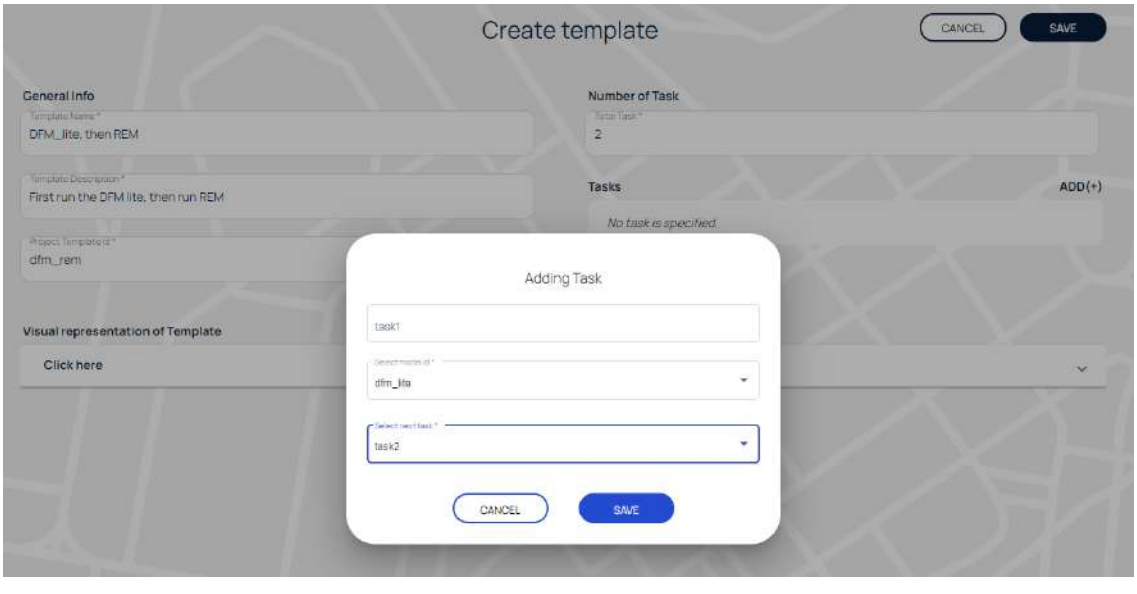

Fig.10 - Creating a Template using the "Create Template function"

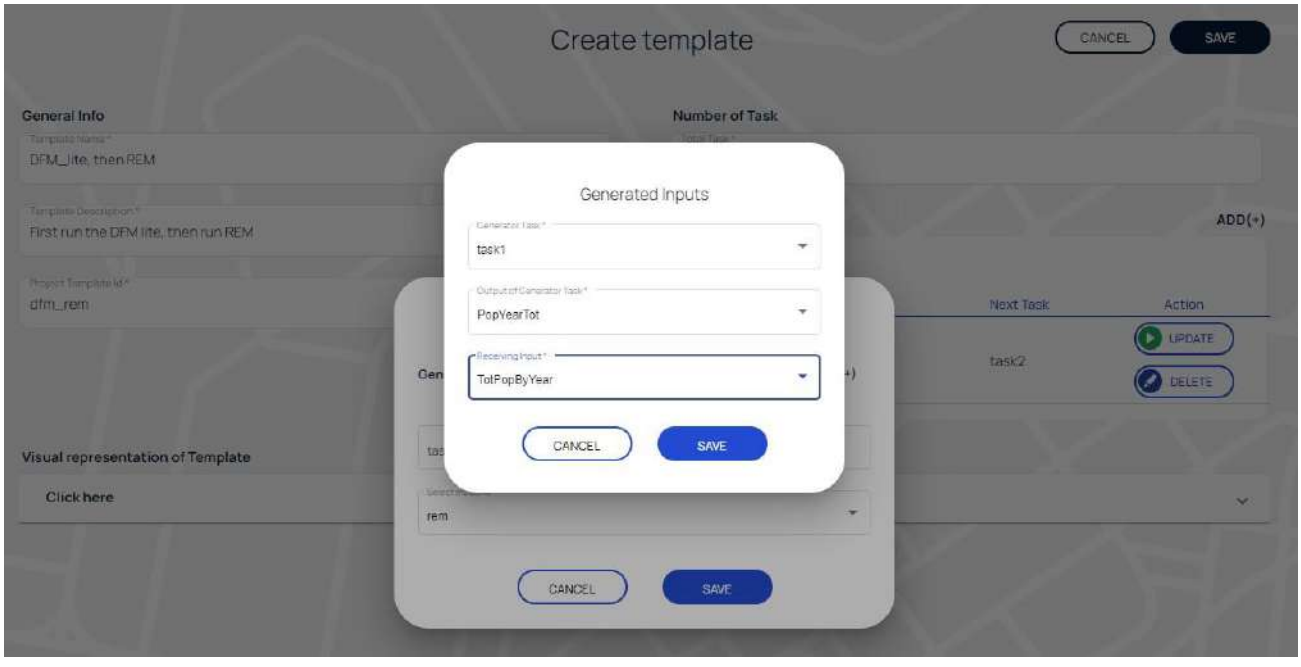

Fig.11 - Adding Generated Inputs to a Modelling Component fed by another one

In both cases, in the ***Templates*** tab, beside the template name, it is possible to click on ***view*** to check the template structure, i.e., the Modelling components included, with their inputs, outputs and linkages, and see the visual representation of the Template.

With respect to the latest, in addition to the visual representation of the single modelling component, also the linkages between the models used for the Template are representedwith purple arrow(s).

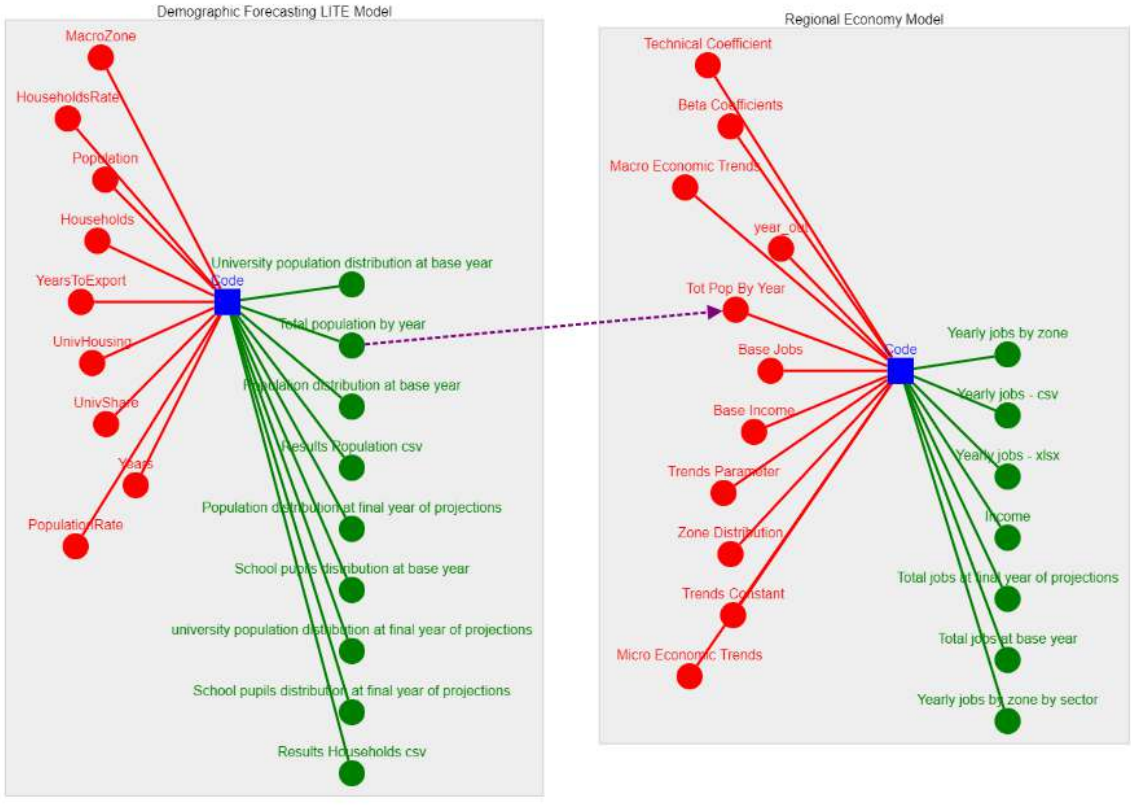

Fig.12 - Visual representation of a Template, with the Modelling Components and their linkages

C. Use input files and templates to create a Project

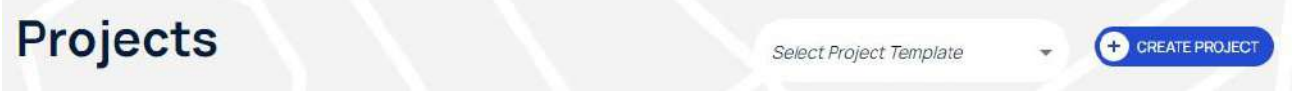

Fig.13 - Select a Template to create a Project with it

Once the modelling **template** has been created, it can be used to upload the input files representing the metropolitan area of interest and initialize the related **Project**. By uploading input files related to a specific context, the Modelling Components and the Templates are applied to the case studies within the **Project**. With this step, not only developers and modellers can use the HARMONY MS, but also planners can play with it.

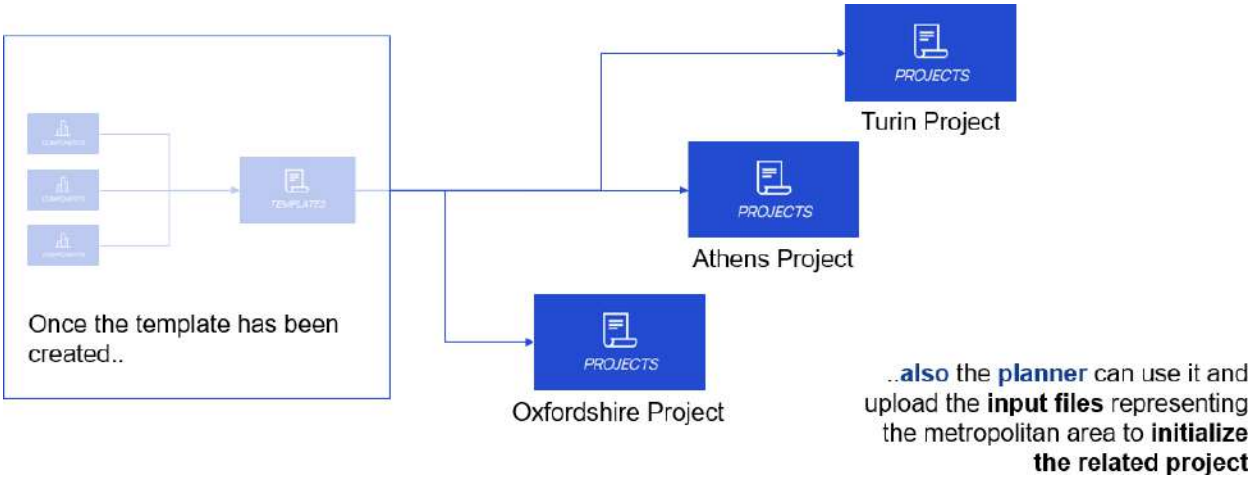

Fig.14 - From a Template, different Projects can be created

To create a Project, one can navigate to the ***Projects*** tab, choose the Template to use, and click on *Create Project*. In the ***Create Project*** tab, it is possible to move from a modelling component to another one, uploading the required input files. The inputs needed are listed under the label *File key and description*: if an input is generated by another modelling component, it is excluded from the list on the right, and included in the list of the *Generated Inputs*.

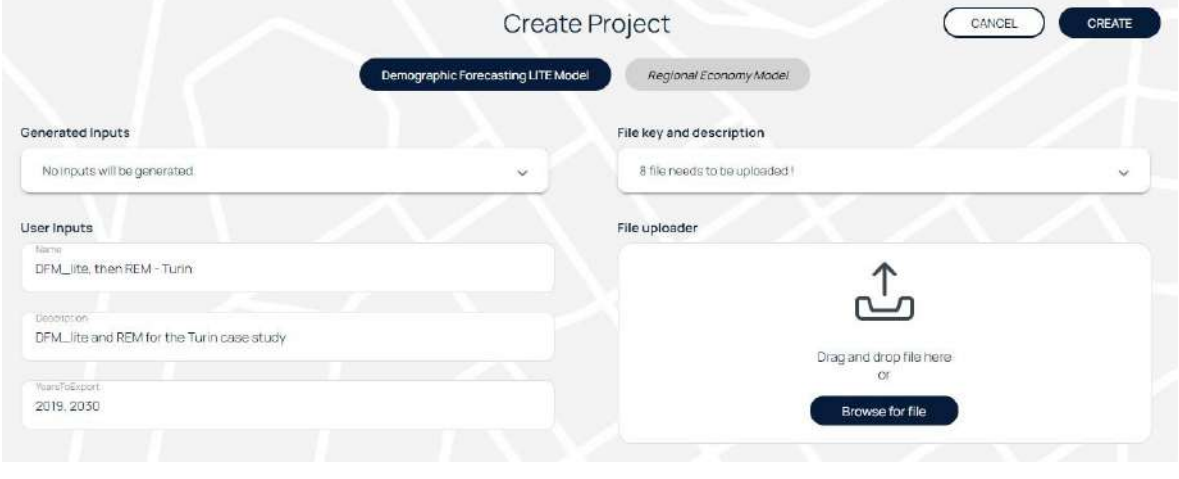

Fig.15 - Creating a Project

The **HARMONY MS** can automatically recognize the input files, if the names are matching those described in the component specifications. Otherwise, one can simply click on the input file not recognized and select manually the corresponding input file of the modelling component in the related folder. Once all the input files of all the modelling components of the template are correctly uploaded, one can click on ***Create*** button, and come back to the Project tab. For each project, a default scenario is automatically generated.

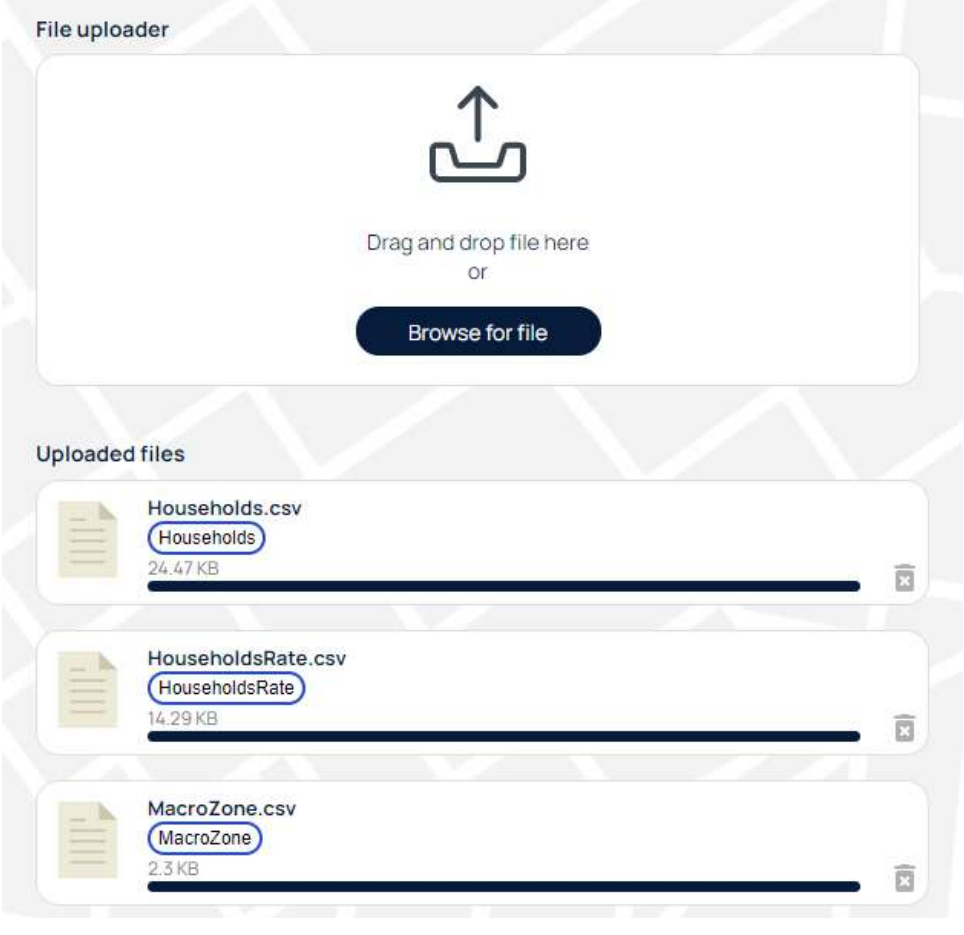

Fig.16 - Uploading files when creating the Projects.

TIP: the *clone* functionality

Since the procedure described above it is time-consuming in case of many input files, it is possible to re-use some steps of the work already done.

For example, if one would like to create a template slightly different from another one already existing in the HARMONY MS, it is possible to navigate the tab ***Templates***, choose the template to clone and click on ***Clone***. This will lead to another template creation page, with all the features of the previous template, but editable.

In the same way, it is possible to create different projects, cloning an already existing project, and changing only a few input files.

D. Run scenarios

i) Run the default scenario

Moving to the tab *Scenarios* of the HARMONY MS, the *default scenario* automatically generated with the creation of the Project is available. Here, by clicking on `Start`, the HARMONY MS will run the modelling project with the features and input files already specified for the related context. A percentage diagram is real-time updated, to show the progress of the model run. Once 100% is reached, it is possible to click on *`View`* and access to the `HARMONY MS Dashboard` to explore the results.

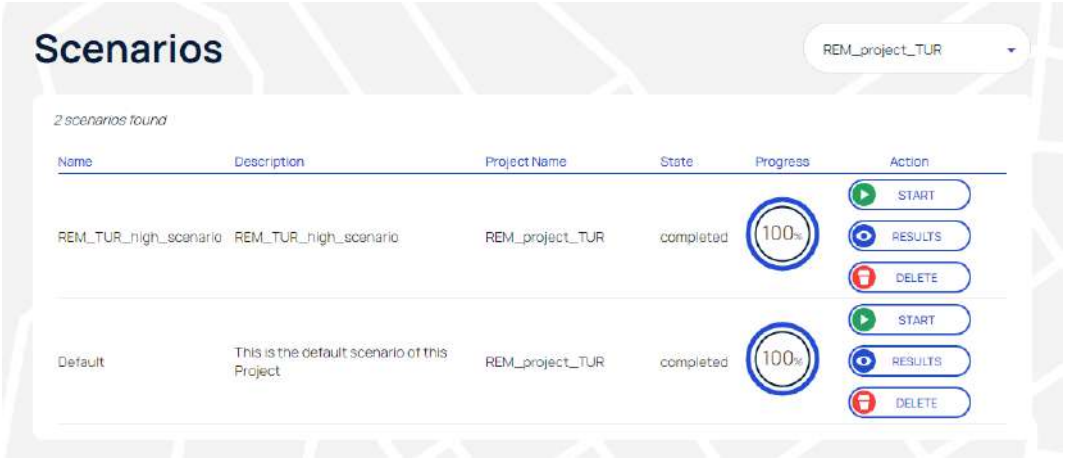

Fig.17 - Scenarios created and run for the Project specified in the top right corner. On the right the options for each scenario

ii) Create different Scenarios

One of the main goals of the HARMONY MS is to compare different scenarios, through an easy and customizable dashboard. Thus, with the same Modelling Components, Template and Project, one can simulate different scenarios by navigating to the ***Project*** tab and clicking on the ***Add Scenario*** button. Following the same procedure described above for uploading the input files, an additional scenario will be created and made visible in the ***Scenarios*** tab. By clicking on `Start`, it is possible to run the scenario in the HARMONY MS and check the progress until 100% is reached.

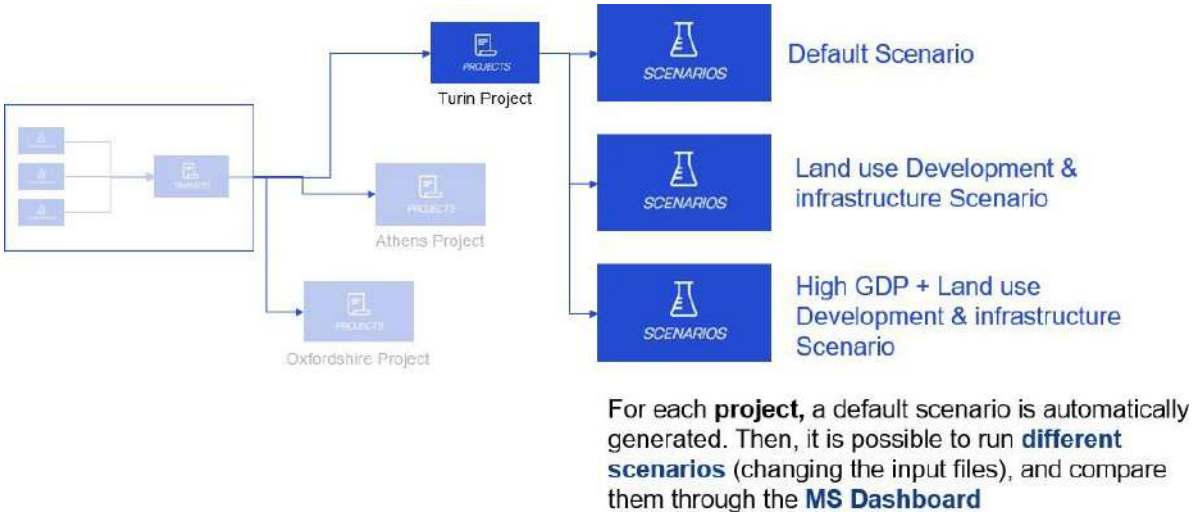

Fig.18 - Different Scenarios can be created for the same Project.

E. Compare different Scenarios in the HARMONY MS Dashboard

Once at least a Scenario has been run, it is possible to click on ***Results*** and open the **HARMONY MS Dashboard**. Here, on the left on top the Project to which the scenario belongs is specified. Below, the list of all scenarios belonging to this project (created and run), is shown. By clicking on a scenario, it is possible to visualize its results, through theselected project KPIs. The list of KPIs is available from a menu at the top of the page: it is possible to add a KPI to the dashboard by choosing it and clicking on the button `ADD KPI`.

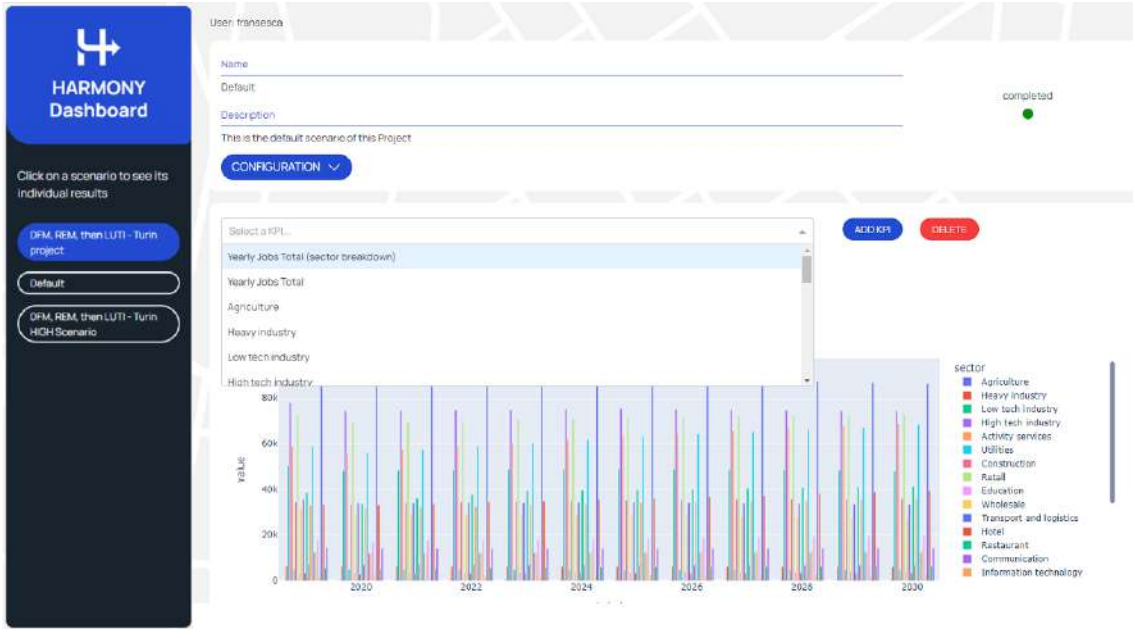

Fig.19 - Selecting a KPI in the HARMONY MS Dashboard.

Once a KPI is added, using the buttons at the bottom of the KPI (Fig. 20), it is possible to(going through the buttons from left to right):

1. delete it.
2. enlarge it (its width is set to the width of the dashboard - see Fig. 21).
3. shrink it (its width is set to one third of the width of the dashboard, so that three smallsized KPIs can be shown next to each other - see Fig. 20).
4. download its data (in CSV format).
5. select it (for a batch action - so far deletion is supported).

Several KPIs can be deleted by first selecting them and then clicking on thered `DELETE` button (next to the `ADD KPI` button at the top).

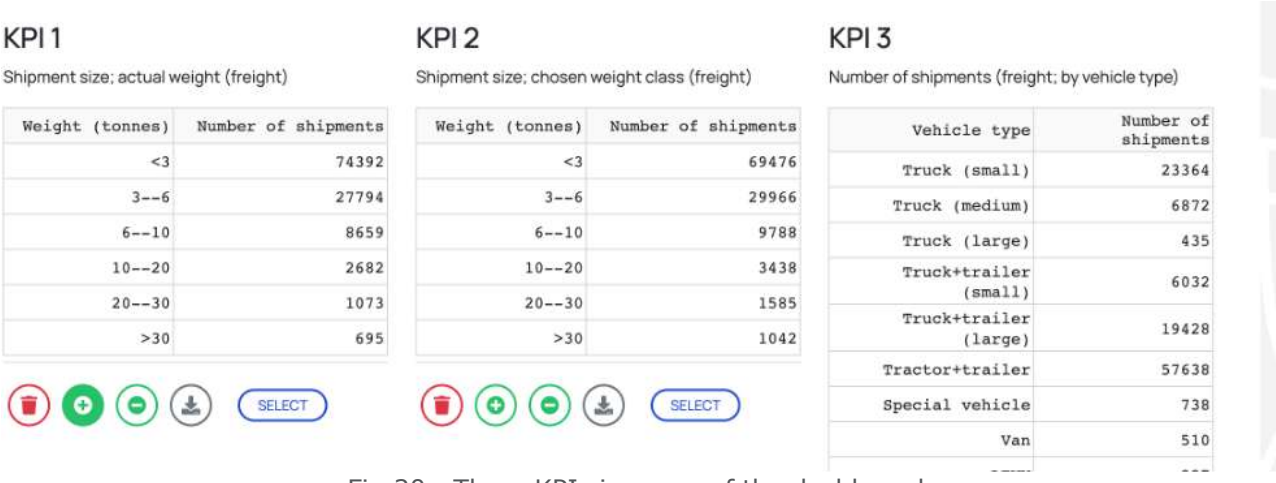

Fig.20 - Three KPIs in a row of the dashboard.

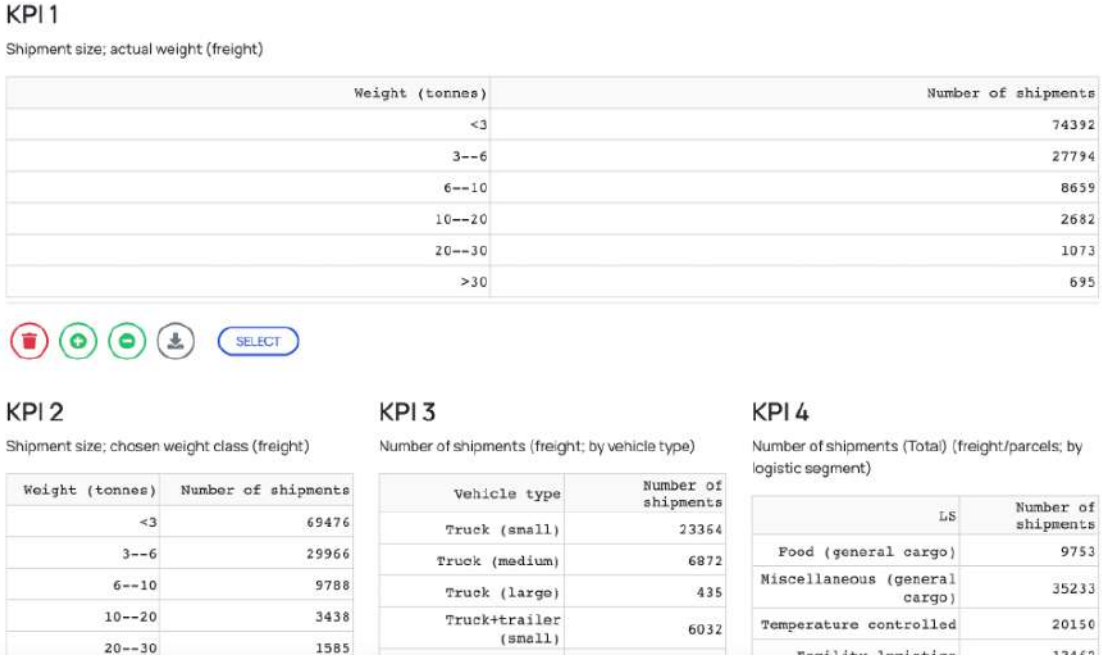

Fig.21 - KPI 1 is enlarged (compare to Fig. 20).

In addition, the following functionality is available by clicking at the `Configuration` buttonat the top:

- by expanding the *Configuration* menu, it is possible to see the visual representation of the **Template**, with all the inputs, outputs, and linkages (in case of more than one modelling component).

- by clicking on each input / output, it is possible to see how it is structured, or directlydownload it. To do it, one should click on the symbol to `download` or `view` it.

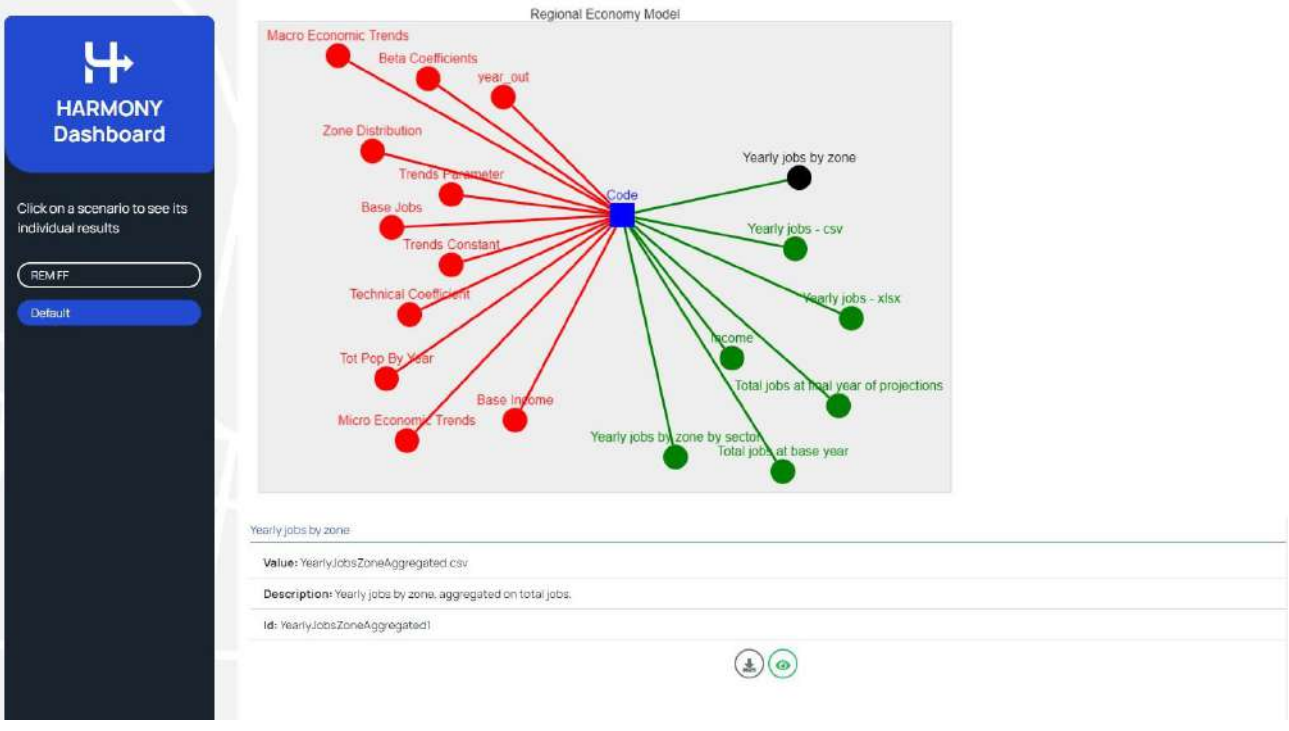

Fig.22 - View or download input and output files in the HARMONY MS Dashboard.

When more than one scenario is simulated, it is possible to compare them through the **HARMONY MS Dashboard**. By clicking on the name of the Project on the left side, on top of the page it is possible to choose:

- from the first menu the *Scenarios* to be compared
- from the second menu the *KPI* to be used for comparison, clicking on thebutton `ADD KPI` as explained above

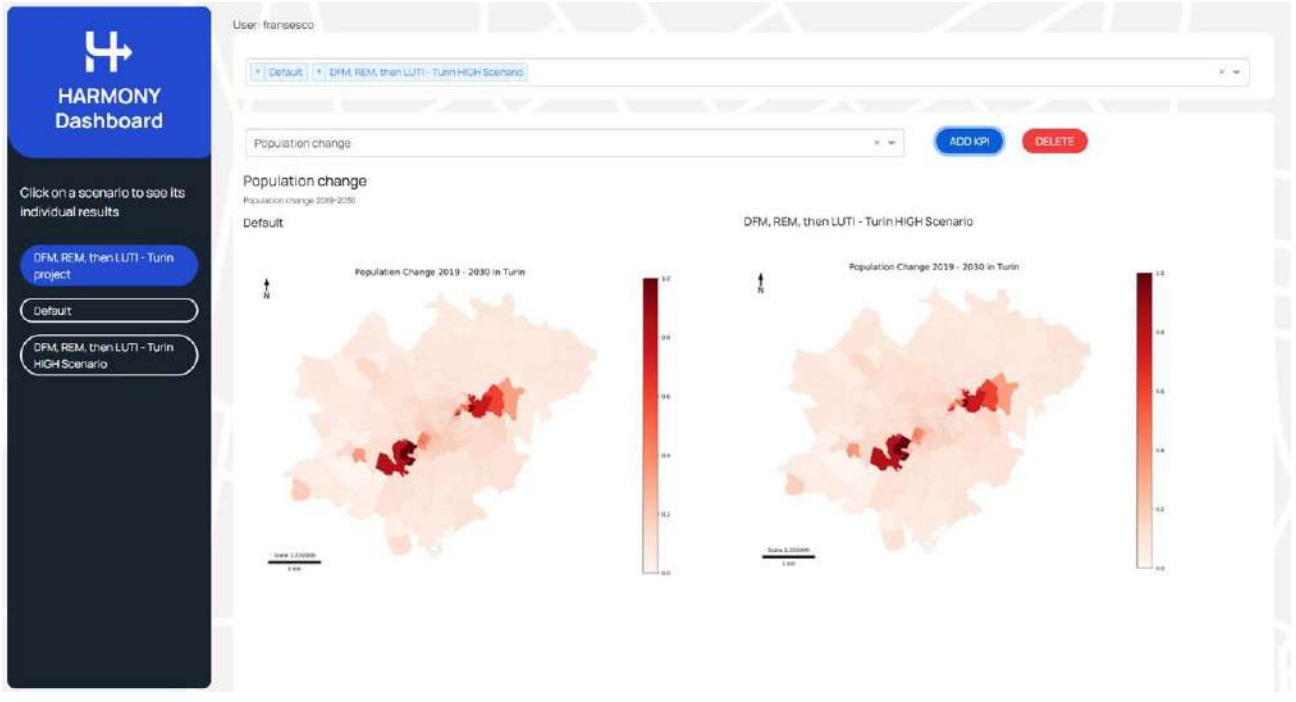

Fig.23 - Comparing a KPI for different scenarios in the HARMONY MS Dashboard.

3. Extending the MS

The MS can be extended by developing and integrating new modeling components. Such components can be written in any programming language as long as they communicate via Kafka and are deployed using Docker. The integration of a modelling component to the HMS is rather complex process in which both Component Developers and MS Maintainers are involved. We describe each role's tasks in turn.

A. Tasks of Component Developer

The first task of the Component Developer is to provide the specification of the modelling component. This comprises:

1. The name of the component.

2. A short description of its functionality.
3. Its ID used to uniquely identify the component in the code of the platform.
4. A description of its inputs – the parameters that need to be set so that the modellingcomponent can be executed.
5. A description of its outputs – the files that are generated once a component isexecuted.

The above are provided in a single, JSON-formatted file, called "component specification". An example of such file can be found *here.*

When preparing such a specification, one needs to consider the following:

- The `name` of the component (line 2) is the name that will be shown in the user interface of the platform (e.g., in the overview of components of the "Modelling Components" page)
- The `description` of the component (line 3) is the description that will be shown in the user interface of the platform (e.g., in the overview of components of the "Modelling Components" page)

- The `modelId` of the component (line 4) is the ID of this modelling component. This isnot shown at the user interface of the platform but is important for uniquely identifying components and Kafka topics as will be described later.

- The specification of each input and output needs to include the keys `fieldType`, `key`, `label`, and `description`.

- The `label` is the name of the parameter what will be shown in the user interface of theMS.

- The `description` will be shown in the user interface of the MS as well (currently this isshown when a user puts the mouse over the name of a parameter in the "Create Project" and "Create Scenario" pages).

- The `key` is used for uniquely identifying the parameter in the code of the platform.

- The `fieldType` **of an input** can be `textbox`, `dropdown`, or `file`. `Textbox` corresponds to an input that is a value (a string or a number), `dropdown` corresponds to a choice between predefined options, and file corresponds to a `file` that needs to be provided as input.

  - In case of `textbox`, then
    - a `scalarType` needs to be specified which can be `string`, `int` or `float`.
    - a `value` needs to be specified which will be shown in the user interface of theMS as the default value for this parameter.
  - In case of `dropdown`, then
    - a dictionary with name `options` needs to be specified, to provide the options tochoose from. Each option is specified by a key (what is shown in the user interface of the MS) and value (the representation of the option in the code of the platform).
    - a `value` needs to be specified which will determine the default (initial) option when this dropdown is shown in the user interface.

  - In case of `file`, no additional information needs to be provided.
- The `fieldType` **of an output** can only be `file`.
- The values of all keys need to follow the mixedCase naming style: initial lowercase character, capital initial letter for any other word.

> **Note**
>
> Such a JSON-formatted specification file can be uploaded directly by the user interface of the MS to create a new modelling component that modelers can use to create Modelling Templates.

The next step of the Component Developer is to provide the source code of the modelling component. Within HARMONY, a private Github repository has been used to collect the source code of the different modelling components from the different HARMONY partners. A partner contributing a modelling

component needs to create a new branch off the master branch of the repository, commit and push their code there, and notify the HMS maintainers, i.e., the team from MOBY responsible for the development and maintenance of the platform during the project. The source code of a modelling component needs to satisfy two important requirements:

1. Wherever the code reads an input file or write to an output file, the file paths are read by the runtime structure (e.g., in Python, this is a dictionary). This is a prerequisite for the following integration steps, where MS maintainers modify the source code so that this runtime structure is updated with the paths to the input and output files managed by the platform (and communicated to the modeling component via Kafka messages, as explained later).
2. All the dependencies (e.g., external libraries, modules) need to be resolved at runtime, so that the MS maintainers can successfully run the code. The way the code is run, as well as any steps to be done before running the code (e.g., downloading libraries, setting up local environment), need to be documented in a file called README.md that is shipped with the code. **The preferred way of packaging the code so that the dependencies can always be resolved is for Component Developers to provide a Docker container image.**

Note

An MS modelling component can be written in any programming language. However, most of the modelling components (more than 90%) are written in Python. To simplify and speedup the development in such a case, we have provided a template project in a private Github repository that a Component Developer can use as a starting point for structuring their components,

B. Tasks of MS Mainteiner

Given a new modelling component that needs to be integrated to the platform, the MS Maintainer (a role that during the

HARMONY project is undertaken by Moby) has the following tasks:

1. Create the specification of the Kafka messages that will be exchanged between the platform and the modelling component.
2. Ensure that the code of the modelling component (i) can receive the relevant Kafka messages, (ii) is packaged as a Docker container.
3. Copy the code of the modelling components to the code base of the MS.
4. Perform the necessary changes in the code base of the MS for the integration to be successful.

The **first task** of the MS Maintainer is to create the specification of the Kafka messages that will be exchanged between the platform and the modelling component. The communication between the workflow engine of the platform backend and each modelling component is done via **Kafka**. Two types of messages exist in this interaction:

1. `start` messages that are sent by the workflow to the modelling components. Once a modelling component receives such a message, it must start its execution. These messages are sent to a Kafka topic named after the ID of the modelling component (to which the code of the component subscribes to).

2. `progress_and_output` messages that are sent by the modelling components to the workflow. These contain a number representing the completion percentage of the execution of the component and, optionally, a dictionary of generated outputs. When a message with completion percentage of 100 is received by the workflow, the workflow assumes that the execution of that component is completed. These messages are sent to a Kafka topic with name `<component_id>` + `"_progress_output"`.

As an example, we can take the *start message specification for the TFS component*. The specification is written in a language called **Protocol Buffers**. At runtime, the **Schema Registry** of the HMS checks whether the messages exchanged in a Kafka topic conform to its specification. If they do not (e.g., an input is missing, or an input is in the wrong format), then an error is thrown. This helps the troubleshooting in the platform since

communication errors can be seen and resolved early.
Hence, the first task of the HMS Maintainer is effectively to map the JSON-based specification to the Protocol Buffers-based specification of start message for this component. In this mapping, it is important to note that:

- The name of an input or output in the Protocol Buffers-based specification corresponds to the key of the corresponding input of output in the JSON-based specification.

- Inputs with fieldType `textbox` and scalarType `string` are mapped to string inputs.

- Inputs with fieldType `textbox` and scalarType `int` are mapped to int32 inputs.

- Inputs with fieldType `textbox` and scalarType `float` are mapped to float inputs.

- Inputs with fieldType `dropdown` are mapped to `string` inputs.

- Inputs and outputs with fieldType `file` are mapped to `string` inputs.

When it comes to a file input or output, only its path is sent over Kaftka (as a string), not the actual file. This is an important decision taken in MS: files are not sent "over the wire" from the platform backend to the modelling components or vice versa. Instead, both the platform backend to the modelling components have access to a common file system (which is implemented as a Docker Volume in the current version of the platform) and are able to write to it and read from it. This decision reduces the amount of data that needed to be communicated between components – since it is much easier and faster to send the path of a file than the actual file (that can be of several MBs). On the downside, this decision means that the modelling components and the platform backend need to be either be deployed on the same machine (our current setup on AWS) or have access to the same network drive.

We note also here that the specification of `progress_and_output` messages is shared among all modelling components – hence no action is needed on this end by the HMS Maintainers. Such specification can be found *here*.

Such a message can contain one or more (`repeated` keyword) generated outputs. An output is comprised by a `key` (which needs to match one of the keys of the outputs in the start message) and a `value`, which is the path to the file that was generated by the modelling component.

Once the Protocol Buffer specification of the start message of the modelling components is created, the MS Maintainer needs to add it under the protos ~~folder of~~ a publicly available Github project called *harmony-interface*. The next time the platform will be built, it will then include the newly created specification.

The **second task** of the HMS Maintainer is to ensure that the code of the modelling component:

1. Can receive start Kafka messages and send `progress_and_output` Kafka messages. The code needs to be adapted to start the execution of the modelling component upon receiving a start message, and to send several progress_and_output messages at different points

in its execution. This step is specialized and clearly depends on the implementation language used for the development of the modelling component. We have documented the necessary steps in full detail for the most common case – that of Python – in a separate Wiki page focusing on the Integration process.

2. Is packaged as a Docker container. This might already be the case for certain provided components in which case no further action is needed. If a modelling component will not be run as a Docker container (contrary to this preferred option), then the necessary setup scripts (depending on the operating system where the platform is deployed) need to be prepared.

The **next tasks** of the HMS Maintainer are summarized below:

1. Update `docker-compose.yml` to add a new service with the name of the new component. This is only needed in the (preferred) case where the modelling component is packaged as a Docker container. Docker-compose is a utility used by the platform to handle the lifecycle of all the Docker containers in the platform, i.e., to start and stop their execution. When a docker container of a modelling component starts, it should simply wait for `start` messages in its Kafka topic.
2. Make the following changes to the source code of the `harmony-interface` project so that the new modelling component is correctly recognized by the platform, in particular:
   - Update the `check_for_start_messages` method in `kafka_message_receiver.py` to include the ProtobufDeserializer corresponding to the proto file of the new component.
   - Add a new method `send_start_X` in `kafka_message_sender.py` where X is the ID of the new modelling component.
3. Make two following changes to the source code of the platform's backend: In file `manager/scenario_runner.py`:
   - Add a new `KafkaMessageSender` for this component at the beginning of the file.
   - Update the method `start_scenario` to be able to start the component when requested by the users in the platform's user

interface.

- In file `manager/kafka_admin.py`, add the new topic name to the list of topics thatare created upon startup of the server.

4. Build the server without cache (to reload the interfaces). This can be done, e.g., by issuing `sudo docker build --no-cache server`.

As already mentioned, this integration process is also described in this separate Wiki page.

4. Installation and initial configuration of the MS

A. System requirements

Make sure following software/tools are installed in the target machine:

1. Python 3.6
2. Docker-compose
3. Kafka-Zookeeper
4. Mongodb
5. Makefile
6. Node (version 12), npm, nvm

B. Installation

1. Please clone the Harmony platform from the following Link: https://github.com/MobyX-HARMONY/harmony_interface/blob/main/harmony_interface/protos/common/progress_outputs.proto
2. To use the platform please open the terminal, go to `HARMONY-Platform/src`
3. To start the platform issue: `make build` and then `make start`
4. To stop the platform issue `make down`
5. To see the logs, please open the browser, go to the following

link http://localhost/9999and you should be able see all the containers and logs.

C. Running the server

To run the server, please make sure you have the correct credential file in json format. Open the terminal and please go inside server directory (`cd HARMONY- Platform/src/server`).

Put the `credentials.json` file inside the server directory. Server will read this file,load the required configuration and should run properly.

D. Running the Database

Since the MS database is password protected, it requires creating a database user upfront to run it properly. To run the database properly, please make sure you have the following file and put them inside the directory `HARMONY-Platform/src`. The files are:

1. `mongo-init.sh`
2. `Init_harmony_mongodb.js`

Inside the `src` directory please run the command `./mongo-init.sh`. This should run our database properly.

E. Running the client app

To run the client app please go inside the client app directory and run the following commands

```
cd HARMONY-Platform/src/client-app
npm install
ng serve
```

Please, open the browser and go to the link `http://localhost:4200/login`. If everything goes alright, the below screen should appear and that ensures the app successfully installed.

F. Creating first user

The platform allows us to create a user to use the platform. To create a user, please log in as Admin. The login credentials can be found inside the `credentials.json` file.
A successful login as Admin will represent the following screen. Click the add user button which will show a user registration pop. Please fill up relevant information to create the user.

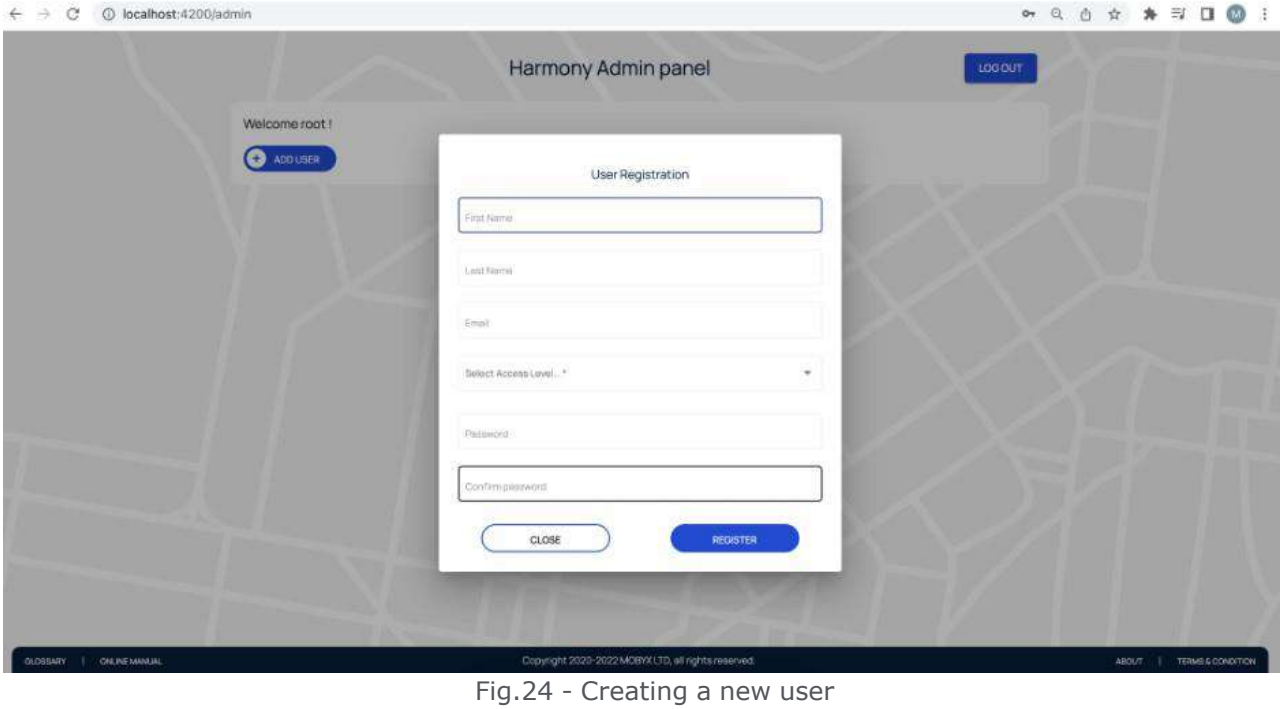

Fig.24 - Creating a new user

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement Nº815269. HARMONY is a project under the CIVITAS Initiative, an EU-funded programme working to make sustainable and smart mobility a reality for all. Read more – civitas.eu.

Website
https://harmony-h2020.eu/

Email
info@harmony-h2020.com

Linkedin

Twitter

Youtube