DEEPHEALTH

D2.5 EDDLL Toolkit front-end

Project ref. no.	**H2020-ICT-11-2018-2019 GA No. 825111**
Project title	Deep-Learning and HPC to Boost Biomedical Applications for Health
Duration of the project	1-01-2019 – 31-12-2021 (36 months)
WP/Task:	WP2/ T2.5, WP3/T3.4
Dissemination level:	PUBLIC
Document due Date:	31/03/2020 (M15)
Actual date of delivery	06/04/2020 (M15)
Leader of this deliverable	SIMAVI
Author (s)	Dana Oniga, Elisa Ionascu, Iulia Stefan
Version	V1.08

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511

Document history

Version	Date	Document history/approvals
1	30/12/2019	First draft contents
1.1	28/02/2020	Revised Content
1.2	11/03/2020	Content reviewed by Jon Ander Gomez
1.3	18/03/2020	Content reviewed by Marius Jianu, Iulian Mocanita
1.4	24/03/2020	Peer review by ProDesign (Heiko Mauersberger)
1.5	25/03/2020	Content reviewed by Dana Oniga
1.6	30/03/2020	Content reviewed by Monica Caballero Galeote
1.7	03/04/2020	Content reviewed by Marius Jianu, Iulian Mocanita, Gabriela Balasoiu
1.8	03/04/2020	Content reviewed by Monica Caballero Galeote, Jon Ander Gomez

Table of contents

Table of Figures

1 Executive summary

Deliverable D2.5 “EDDLL Toolkit front-end” presents the result of T2.5: *Design and Implementation of a front-end for training models*, within WP2 - *Design and Development of the EDDLL library* and part of the work done in T3.4 *Implementation of a front-end for image manipulation*, within WP3 - Design and Development of the ECVL.

The DeepHealth toolkit includes both libraries, ECVL and EDDLL, plus the front-end exposed to expert users for an efficient usage of all the functionalities of these libraries. This toolkit will be free and open-source software.

The ECVL library is devoted to image processing functionalities useful for pre- and post- processing, data loading, augmentation and image format management, while the EDDL library provides the tools for defining Deep Learning pipelines and executing them efficiently on hybrid hardware resources.

A web-based couple front-end and back-end completes the DeepHealth infrastructure and makes the toolkit a stand-alone solution.

It is extremely important to underline that ECVL and EDDL libraries strictly cooperate within the DeepHealth toolkit, every time that a training or inference process takes place in EDDL a previous images manipulation (i.e., dataset loading and splitting, image augmentation, etc.) occurred. For this reason, the development solution adopted unifies ECVL and EDDL front-ends, instead of implementing an EDDLL specific front-end (and one for ECVL). The same applies for the back-ends.

Therefore, despite the initial title of this deliverable, this document aims at reporting the rationale behind the DeepHealth toolkit front-end functionalities for both EDDLL and ECVL, providing a general description of this part of the toolkit, its interaction with the back-end, and a detailed documentation manual. Withal, the back-end counterpart could be found detailed in report D3.5 “ECVL Toolkit Front end”.

2 Introduction

The DeepHealth Toolkit (DHt) is composed by three main components, the two libraries ECVL and EDDLL plus the so called front-end (see Figure 1). However, and in order to avoid confusion, the front-end of the DHt is a software module that is divided into two parts, one visible to the user (a.k.a. graphical user inferface–GUI) through the navigator (Firefox, Chrome, Edge, …) and one invisible part that performs all the actions indicated by the user using the functionalities provided by both libraries. The GUI is also known as front-end. The invisible part is commonly known as back-end, which is described in D3.5 – ECVL Front-end Toolkit. In order to clarify, when we use the concept of front-end in this document (D2.5) we refer to the GUI. When we use the concept of front-end as a component of the DHt, we refer to the module that includes both the GUI and the backend.

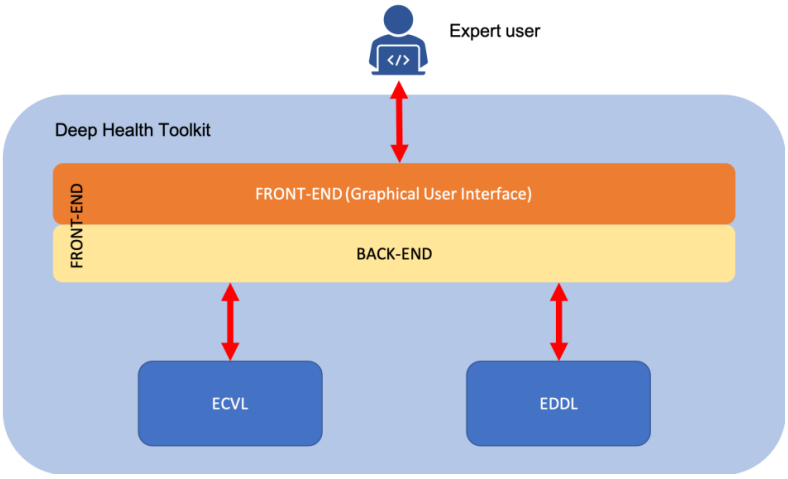

Figure 1: Scheme of the DeepHealth toolkit

Therefore, this deliverable provides the descriptions of the brown block titled as “GUI (FRONT-END –Graphical User Interface)” in Figure 1.

The design of DeepHealth toolkit follows several core principles to support a set of key requirements:

1. support a wide variety of application types in medical image analysis and computer-assisted diagnosis;
2. be simple to be used in common use cases, but flexible enough for complex use cases;
3. support parallel processing, model distribution and adaptation;
4. support best practices as data augmentation and correct data set partitioning;
5. user-friendly visualization.

The web-based graphical user interface (GUI) is the tip of the iceberg that allows the entire toolkit to follow the aforementioned core principles.

The Graphical User Interface then allows user to access, in a graphic way, to all functionalities provided by the libraries (ECVL and EDDL) and provides the graphical tools to exploit these functionalities in a user-friendly way. To achieve this goal, the front-end needs to be coupled with a back-end that manages low level libraries interaction, driving the dataset loading, image manipulation and the training/inference processes.

This architecture provides several benefits:

1. Expert/end users do not have to manage individual software, installing and updating them at every library update;
2. Similarly, the web browser interface delegates the pairing with the hardware infrastructure to the software maintainer, meaning that any customer can easily maximize the use of the toolkit on any combination of hardware and operating system;

3. Expert-users will no longer have to worry about compatibility between the graphical user interface (GUI) and the server. The graphical user interface is intuitive, attractive, and easy to use and to learn.

The back-end includes the datasets management tool, and allows handling data/image manipulation pipelines. The former allows loading dataset images to be manipulated and that will feed the neural network model for training or testing.

Another important result related to the front-end/back-end infrastructure is that the latter will be a key element in the cloud-based version of the toolkit. In fact, the backend will serve as an entry point for the requests and then distribute the workloads to computational facilities through the network.

It is important to stress that ECVL and EDDLL libraries strictly cooperate within the DeepHealth toolkit. Considering a typical scenario in which expert users interact with the toolkit, they will want to provide their data (for example an image or a dataset) and then choose a neural network model that performs the desired activity. In this situation ECVL read, augment and expose the images to EDDL, which is in charge of setting up the neural network model for training or inference.

Therefore, despite the initial proposal, in the adopted architectural software design, instead of implementing an ECVL specific front-end (and a specific one for EDDL), we tightly coupled the two libraries in a single front-end. The same applies for the back-end.

Besides the initial title of the deliverable, this document aims at reporting the GUI as part of the DeepHealth toolkit front-end functionalities for both ECVL and EDDL, providing a general description of this part of the toolkit, its interaction with the back-end and a detailed documentation manual. The back-end counterpart for both ECVL and EDDL is reported in Deliverable 3.5.

The rest of the document is structured as follows: Section 3 gives the details about the Graphical User Interface as part of the DeepHealth Toolkit front-end. Section 4, Functionalities, offers more information about the technologies used in the development process and about the main functionalities developed. Section 5, Main Components, describes the main features displayed by the graphical user interface of the front-end application. Finally, Section 6 gives more details about the relation between the Graphical User Interface and the back-end of the DeepHealth toolkit front-end.

3 Description of the Graphical User Interface

The DeepHealth toolkit Front-end is a web-based graphical, intuitive and attractive user interface that allows the user to access all of the functionalities provided by both libraries.

The EDDLL library provides the tools for defining Deep Learning pipelines and executing them efficiently on hybrid hardware resources. The ECVL library is devoted to image processing functionalities useful for pre- and post- processing, data loading, augmentation and image format management.

The interface helps and guides users to do the most common tasks, based on the compatibility between the graphical user interface (GUI) and the server (back-end). The GUI includes an option to update itself as new versions become available.

The features provided by the interface are organized in three main groups:

- datasets management
- pipelines and algorithms management
- results

The application loads image files as DICOM and NIFTY formats and CSV files. Similarly, compatible datasets can be combined and index files are generated in order to use images from two or more sources. After that, there are defined pipelines for image transformations and extracting aggregated values from complementary data. Transformed images and aggregated values are used to train and test deep neural networks. The results of evaluating the trained models are displayed in a graphical way, allowing the user to make comparative studies. The user will be automatically notified when a training or inference process is finished.

Datasets management include pre-processing images using ECVL functionalities. Pipelines to train models include on-the-fly transformations to images done by the ECVL. During the training process of a DNN the batches of samples can be prepared on-the-fly due to memory limitations, “prepared” means images are loaded from disk and transformed if required, usually for data augmentation, so such transformations are done by the ECVL, not by the EDDLL.

The DeepHealth toolkit front-end was developed based on the requirements highlighted in D1.4 Definition and specification of the DeepHealth Toolkit, Chapter 6.4: Component details: front-end.

4 Functionalities

4.1 Technologies used

For the development of the front-end there have been used common programming tools, such as HTML5 with CSS3, and Angular 7.3.9. Angular is used to write scripts in the *Typescript* scripting language (https://www.typescriptlang.org/) that run the different tasks offered as options in the user interface. Scripts execute the runtime programs provided by both libraries.

Angular is a web development framework, made by Google team as an open source platform that uses mainly *Typescript*. This framework offers the possibility to create a Single Page Application which includes more components. These components are structured like a tree with parents who have one or more child components connected.

We needed an interactive interface to be compatible with most browsers, actively maintained and to offer the simple communication with the back-end. Angular provides components that update dynamically and a logical structure to keep the code separated; it is easy to style the html elements using its own Angular Material Design components which implement common interaction patterns according to the Material Design specifications.

Angular provide feature for compressing and encryption merge source file for production and also provide feature for easy debug the front-end code in browser or in the IDE.

Also, the advantage of the *Typescript* is the structure of the object-oriented programming language, the declaration and initialization of type safe variables and auto completion and refactoring in development processes.

We have used Angular Material which is a library that contains user interface components such as buttons, radio buttons, toggles, modals.

Material design was developed by Google as a design language meant to use more animations and transitions and style elements in order to improve the existent digital materials.

4.2 Main functionalities

The main functionalities provided by GUI are mentioned below:

- Creation of new projects
- Uploading of new datasets
- Selection of models and datasets
- Selection of training and/or inference process
- Selection of classification or segmentation
- Selection of input type

The user has the option of loading from back-end sets of models and datasets in order to start the training or inference process.

1. *Train*: the user wants to see how well a model can perform on his data, by training it from scratch, or from an existing set of weights -a pretrained model. This will create a new set of weights.
2. *Inference*: the user wants to see how well a model can perform on his data, by only running the data through a model with an existing set of weights. This does not create new weights for the selected model.

The user has the possibility of selecting some specific properties for training processes or inference processes because not all of them are useful for both processes. This is visible on the configuration page of the project page (see Figure 5).

The Graphical User Interface will be updated whenever the two libraries will be updated too. In this context, the version of this solution can be considered final only when the libraries are at the final version.

Based on the experience accumulated during the development of this project we would like to bring improvements to this solution until the time of the final version. These improvements are:

1. The training method, meaning that if the user fills the *Weight* field, the back-end performs a retrain of the model; otherwise it launches a training from scratch;
2. Also, it is necessary to deploy the front-end application on a public external server to be tested by the partners;
3. The front-end should prevent the case when the browser window is accidentally closed per session and for that it will be implemented the functionality that returns user-preset data for a train/inference process.

5 Main components

The main components of the front-end application are the user page and the project pages.

5.1 User page component

Below, you will find the first page of the mock-up used (Figure 2) while developing the application and a screenshot of the user page (Figure 3) version available at the time of writing this document.

The application includes a menu panel on the left side of the page, which contains all the models and datasets available for training. On the right side of the page there is an area for all the projects created previously by the user. The user has the option to open an existing project or to create a new one, by clicking the “Create new Project” button.

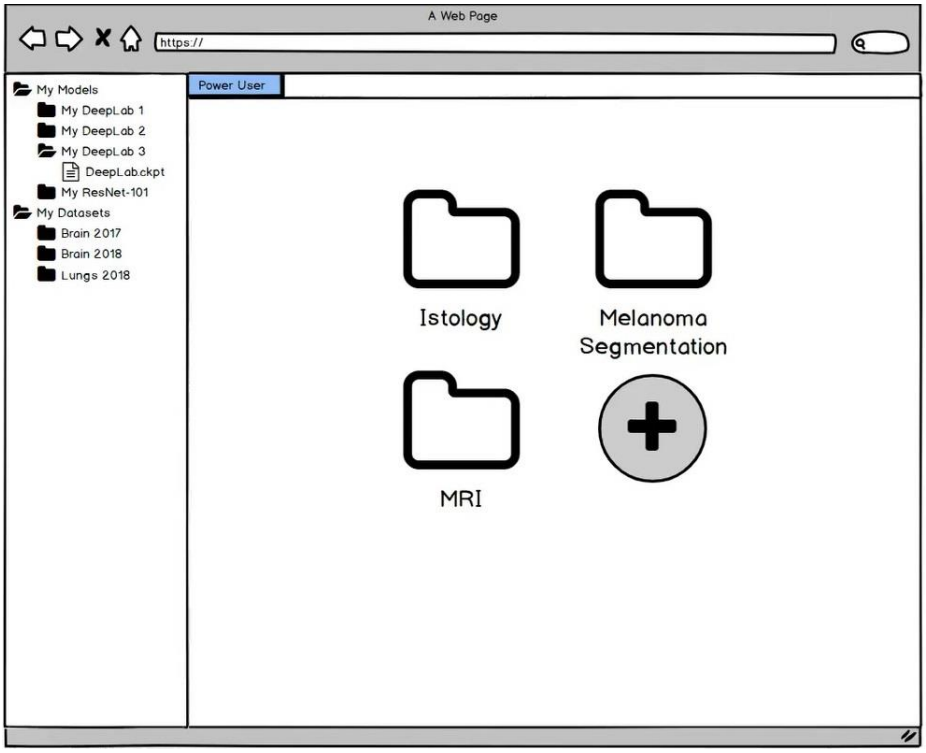

Figure 2: First page of mock-up - User page

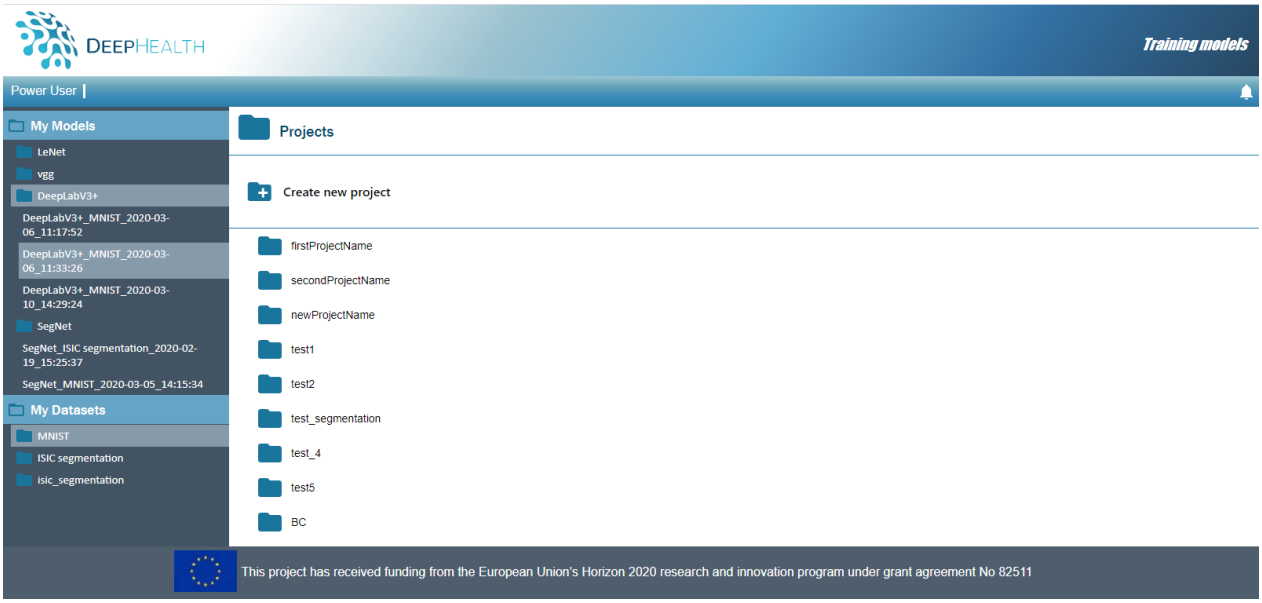

Figure 3: User page front-end

5.2 Project page component

Figures below illustrate the mock-up (Figure 4) used for the project configuration page while developing the application and screenshots of the project page version available at the time of writing this document (Figure 5).

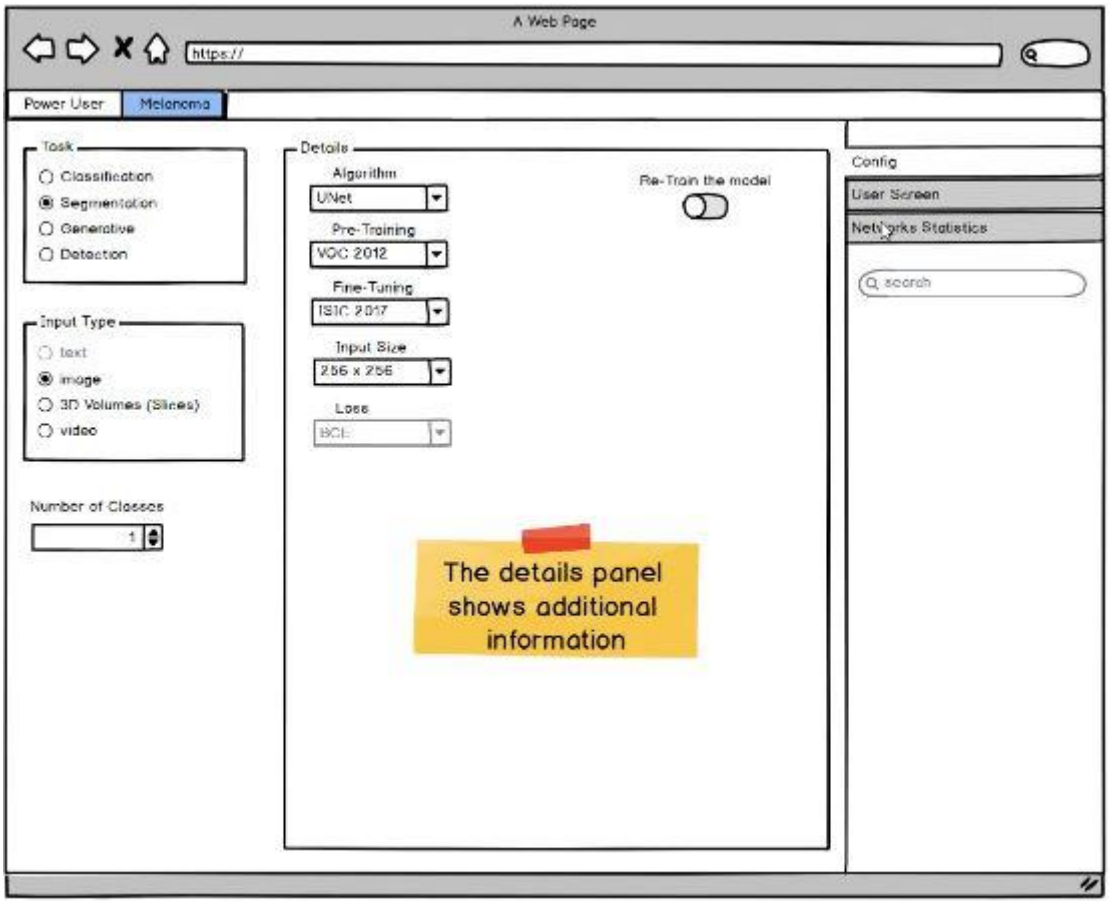

Figure 4: Project page of mock-up

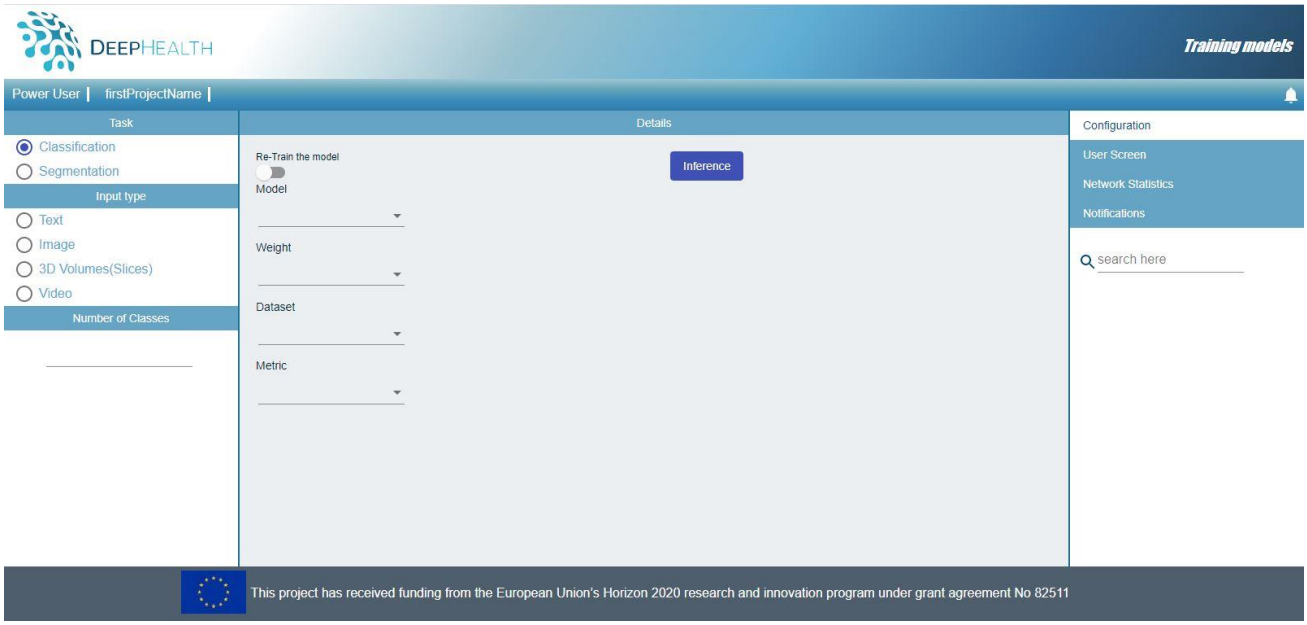

Figure 5: Project page front-end

The project configuration page includes on the right side buttons for choosing different pages afferent to a project. This panel stays the same on all pages.

The configuration page has another section on the left, which is divided into three parts. The first one is dedicated to choose the task which you wish to perform. The second one allows the user to choose the data type of the input file used. The last one allows the user to increase or decrease the number of classes.

In the middle section, the user can select specific details for each project, such as the model and the dataset used, as well as specific properties for a process. You can find a button for starting an inference process and one for starting a training process.

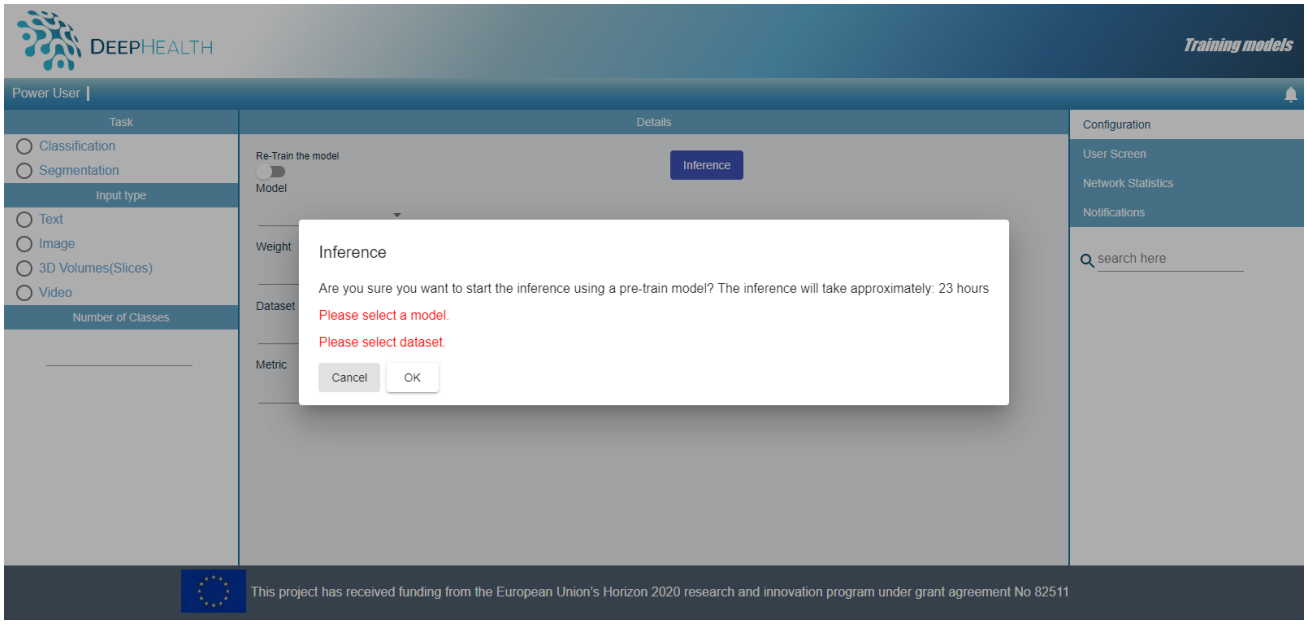

Figure 6: Project configuration page – inference process

For an inference process it is mandatory to provide a pre-trained model and a dataset in order to start the process as shown in Figure 6. The model and the dataset can be selected by using their dropdown selectors from the “Details” section of the page. When you click on the Inference button, the following dialog will appear in order to see if you have selected all the mandatory fields and to display how long the process might take to finish. You can choose if you still want to start the process by clicking “OK” or cancel the request by clicking “Cancel”.

Figure 7 illustrates the training process which needs a model and a training dataset. For starting a training process you need to click the “Train” button, which is displayed after enabling the “Re-Train the model” toggle. The following dialog will be displayed on the page, which shows if you have selected the mandatory elements and, if so, the model and dataset that you have selected. Similarly to the inference dialog, it also displays how long the training will take and it gives the option of cancelling or starting the process. In training process the Weight dropdown can be empty or not selected. If the user fills the Weight field, the back-end performs a retrain of the model; otherwise it launches training from scratch.

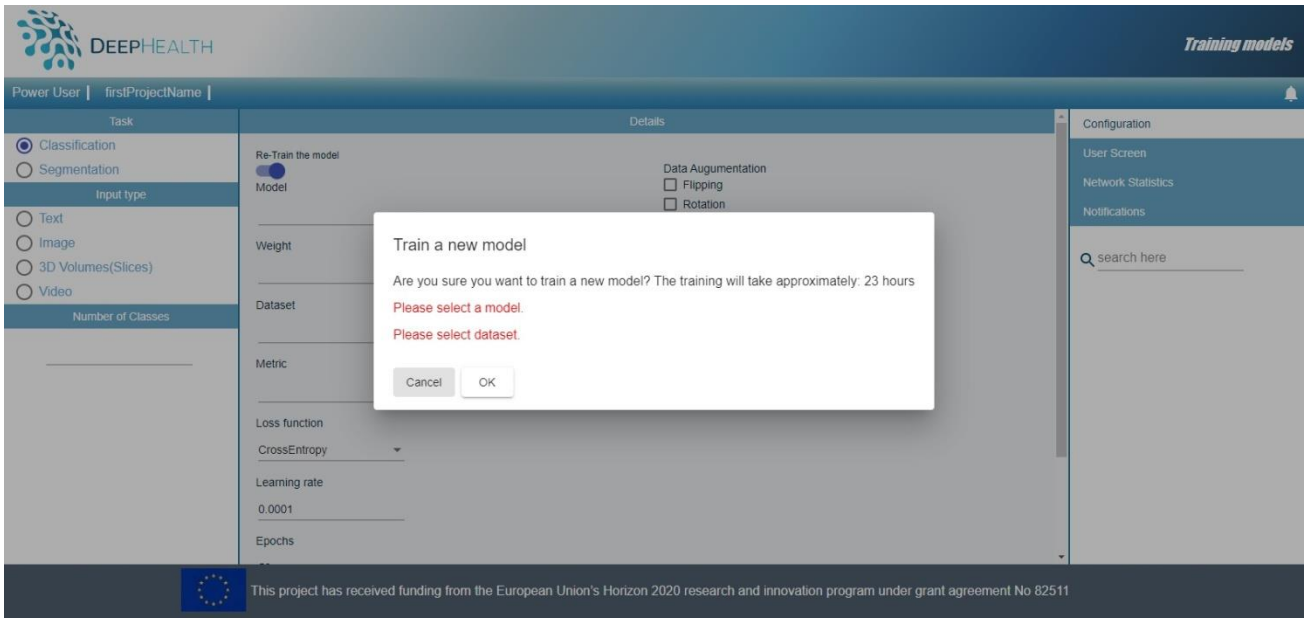

Figure 7: Project configuration page – training process

After you start a process, you will receive a notification. It is displayed in detail on the Notifications page (see Figure 8), which can be accessed by either clicking the Notifications button on the right side of the page or the little bell icon on the menu bar on top of the page. Every new notification is marked by a little black envelope, which dissapears when you click on the notification row.

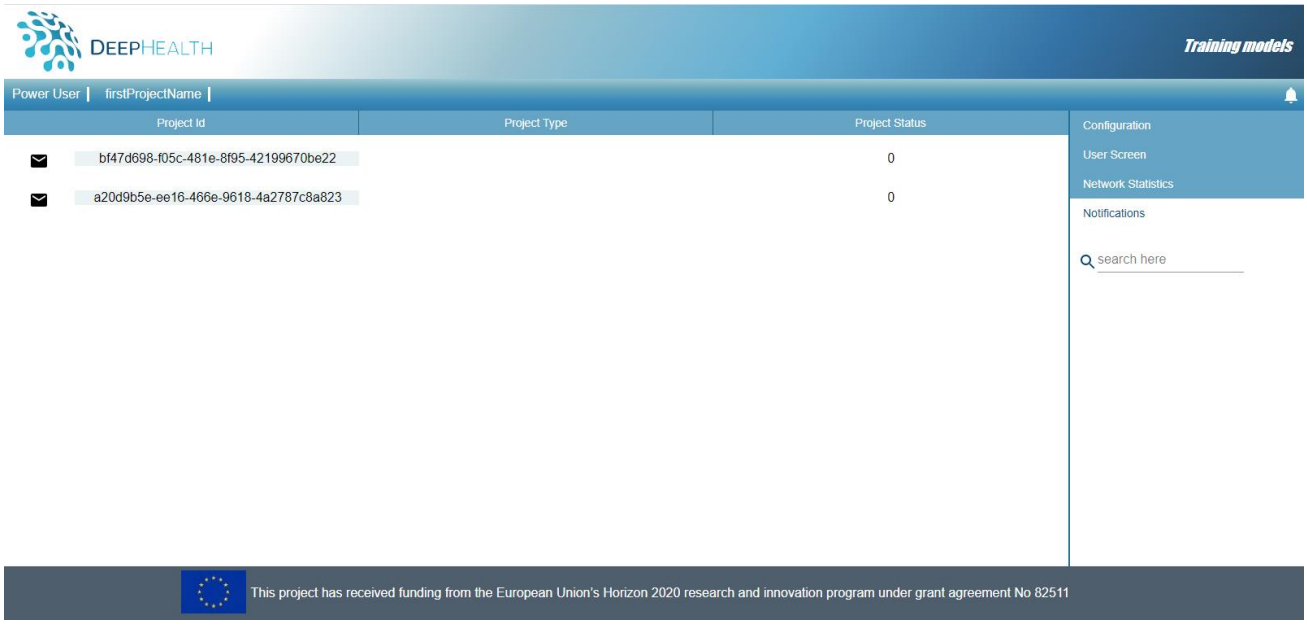

Figure 8: Project Notifications page

6 Back-end

As was mentioned in the Introduction chapter, The DeepHealth Toolkit (DHt) is composed by three main components, the two libraries ECVL and EDDLL plus the so called front-end (see Figure 1). The front-end of the DHt is a software module that is divided into two parts, one visible to the user (a.k.a. graphical user interface–GUI) and one invisible part that performs all the actions indicated by the user using the functionalities provided by both libraries. The GUI is also known as front-end. The invisible part is commonly known as back-end, which is described in D3.5 – ECVL Front-end Toolkit.

The DeepHealth back-end has been developed as a RESTful web service, where REST is acronym for **RE**presentational **S**tate **T**ransfer. The online interactive documentation of the APIs is available at https://app.swaggerhub.com/apis/pritt/DeepHealthToolkitAPI/1.0.2.

The GUI communicates with the DeepHealth Toolkit back-end in order to get all the information needed to prepare and run training or inference processes, to view the processing status and the processing results.

The methods implemented at this moment in the GUI interface are presented below:

1. **/datasets of type GET** is used for getting all the datasets used for training or inference. The datasets list loaded is shown on the left side of the User Page Component and is necessary for the training and inference procedures, which require a model and a dataset in order to start the process;
2. **/models of type GET** returns the models list available for training or inference processes. These models are also shown on the left side, in the same panel as the datasets list;
3. **/projects of type GET** loads all the available projects, which are shown on the right side of the User Page;
4. **/projects of type POST** is called when adding a new project. The user has the option to open an existing project or to create a new one; When a project is created it is mandatory to select the task which you wish to perform;
5. **/projects of type PUT** is used to update a project every time the user changes the task.
6. **/properties of type GET** returns all the properties related to a model supported by the back-end. These properties are shown in the middle of the Configuration page;
7. **/tasks of type GET** returns the tasks related to a model and are shown on the left section of configuration page. There are two options: classification or segmentation tasks.
8. **/weights of type GET** is used for getting all the weights available in the back-end. When the button ‘Re-train the model’ is selected, it is possible to query the back-end passing a model_id to obtain a list of datasets on which it was pretrained.
9. **/inference of type POST** is used to start the inference process. For this process it is mandatory to provide a model and a dataset. Inference processes return a process id which is used to view the processing status.
10. **/train of type POST** is used to start the training process. It is mandatory to specify a model and a dataset to be trained. In the training process, the Weight dropdown can be empty or not selected. If the user fills the Weight field, back-end performs a retrain of the model; otherwise it launches a training procedure from scratch, i.e. with weights initialized randomly. Also as inference process, training process return a process id used in method /status.
11. **/status of type GET** returns the status of a training or inference process. After a process is started, the status will be displayed in detail on the Notifications page.

7 Conclusions

This document presents the Graphical User Interface (GUI) as part of the DeepHealth toolkit front-end functionalities for both ECVL and EDDL, providing a general description of this part of the toolkit, its interaction with the back-end and a detailed documentation manual. The back-end counterpart for both ECVL and EDDL is reported in Deliverable 3.5.: ECVL Toolkit front-end.

The work started based on a mock-up of two pages: user-page and project configuration page. The development was done in HTML5 with CSS3, Typescript in Angular 7.3.9. and Material Angular.

The GUI queries the back-end of the DeepHealth Toolkit in order to obtain lists of models and datasets used for training or inference processes. The user can choose an existing project or create new ones. The user can also personalise the processes by editing the value of different properties/parameters by using the user interface. After a process has been started, the user receives notifications and the details that are displayed in the notifications page and after that, the Interface shows the final output of these processes.

In conclusion, the GUI solution developed is functional and ready to be used by medical experts and AI Experts. Improvements will come following the updates of libraries evolution and will be reported in following deliverables related to libraries: D2.1 EDDLL library, D2.2 EDDLL library (II), D3.1 ECVL library and D3.2 ECVL library (II).

8 Appendix

In order to install the front-end, expert users need to follow some simple steps. The build and run steps can be done in three versions:

i. manual build and run using just *nodejs*,
i. build and run using Apache Tomcat server - the best way for better compressing target files
ii. docker compose build and run.

Manual build and run using just Nodejs

➢ Prerequisites

- You need Nodejs 8 or up installed in your setup.
- You need to install Angular Cli version 7.3.9 by command:
  - npm install -g @angular/cli@7.3.9

In folder where angular.json and package.json exist run next commands:

1 install node_modules libraries:

- npm install

2 build application and start angular default server:

- ng serve

3 access in browser:

- http://localhost:4200

Manual build and run using Apache Tomcat server - the best way for better compressing target files

➢ Prerequisites

- You need Nodejs 8 or up installed in your setup.
- You need to install Angular Cli version 7.3.9 by command:
  - npm install -g @angular/cli@7.3.9
- You need Java 8 and JAVA_HOME environment variable for the installed java 8 set.
- You need binary Apache Tomcat 8 server.

In folder where angular.json and package.json exist run next commands:

1 install node_modules libraries:

- npm install

2 build the project / create target files for production mode - best compressing target files:

- ng build --base-href "." --prod

3 - deploy application:

The build command will create a /dist folder that contains target /deep-health folder with compiled files. You need to copy /deep-health folder in root /apache-tomcat8/webapps/

4 - start Tomcat server

From Tomcat root folder, in /bin run startup.bat(for Windows) or startup.sh(for Linux)

5 - access in browser:

default Tomcat start on port 8080, access:

- http://localhost:8080/deep-health

Docker compose build and run

➢ Prerequisites

- You need docker and docker-compose installed in your setup.
- You need Nodejs 8 or up installed in your setup.
- Node js will install (by docker config) Angular Cli 7.3.9

1 - start Docker - make sure docker is in running status

2 - Verify: $ docker -v

Docker version 18.09.2, build 6247962

$ docker-compose -v

docker-compose version 1.23.2, build 1110ad01

$ node -v

v8.11.3

3 - build the docker image and the application:

- docker build -t deep-health:prod .

4 - run docker image:

- docker run -d --name deep-health -p 4200:4200 deep-health:prod

5 - access the application in browser on port 4200

- http://localhost:4200

6 - administration steps:

stop - in case you want to stop docker image:

- docker stop deep-health

start - in case you want to start the stopped docker image:

- docker start deep-health

to have a status - view running docker images:

- docker ps -a

remove imagine docker in case of image running problems, remove the image with deep-health name(using the ID) and repeat the above steps:

EX: docker rm a09e5b182263