Bounce Mitigation Tool (BMT)

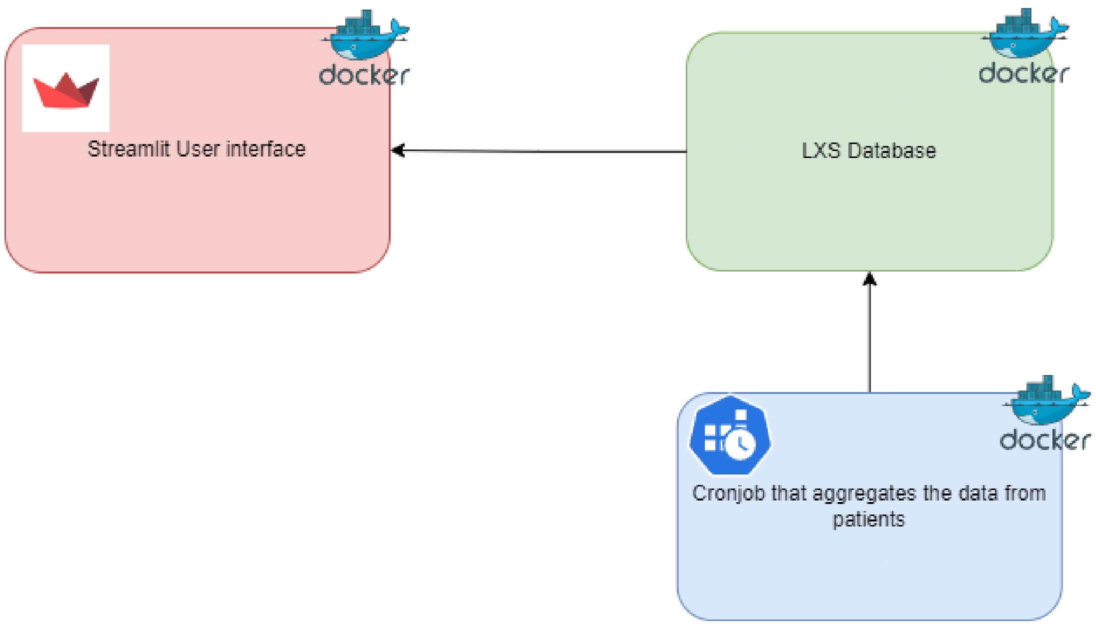


Figure 1: **Application Architecture**

Introduction

The Bounce Mitigation Tool (BMT) evaluates patient engagement in the program by analyzing the rate of questionnaire completions and the frequency of device utilization, providing valuable insights into a patient's commitment to the program. The data compiled is displayed using charts that can be analyzed and draw conclusions to what is need it to be done next.

Contents

1 Overview of BMT

The Bounce Mitigation Tool at its core is overall a straight froward solution for analyzing the commitment of a patient to the program, using a simple, but elegant user interface that is easy to use, utilizing the full potential of the streamlit framework and the python programming language. The tool make full utilization of graphs to showcase the different aspects that analyzes the commitment to the program, like the graph for showing the questionnaire completions, the device utilization graph, and also the most important part of the application, the risk assessment graph, which showcases the patients that are at risk of being thrown out of the program because of they did not participate sufficiently.

2 Prerequisites

Technology Stack:

1. Python
2. LeanXcale Database
3. Docker
4. Kubernetes
5. Helm Charts

The BTM is an application written in Python and utilizes the Streamlit framework for realizing the user interface, and besides the framework there are some Python packages that do not come with the vanilla libraries and modules of Python, and needs to be installed separately using pip.

The application is already wrapped up as a Docker image and can be run as a Docker container instead of using it directly, and also has the Helm Chart already made in case the application needs to be used and deployed inside a Kubernetes Cluster.

2.1 Python Requirements

The application is really small and doesn't use very much packages outside the default ones, and the packages that needs to be installed, besides the ones that already came with Python, are:

1. **Streamlit**, which is also the main component and the framework that creates the UI
2. **Pandas**, which is used for data processing that will be displayed on the graph inside the UI
3. **JayDeBeApi**, which is used for connecting to the LeanXcale database and retrieved the data

2.2 Running with Docker

There are two options to run the application with Docker:

1. To build the image and then run it as a docker container
2. To get the image that is already built from gitlab.ihelp-project.eu:5050/ihelp/bounce-mitigation/bounce-mitigation-ui:0.1.0

2.2.1 Build the image

In order to build the image the project from the repository `https://gitlab.ihelp-project.eu/ihelp/bounce-mitigation/bounce-mitigation-ui` needs to be cloned and after any additional changes to the application, or none if the application fits the needs, starting to build the image using:

```
docker build -t <image_name> .
```

And to run the application using:

```
docker run -d -p 8050:8050 -e DB_URL=jdbc:leanxcale://147.102.230.182:30001/ihelp
-e DB_USER=app -e DB_PW=app -e DB_DRIVER_CLASS=com.leanxcale.client.Driver
<image_name>
```

2.2.2 Run with the built image

This would be the recommended way to run the application to avoid any hiccups in trying to set up and build the image from zero. And as for running the application using the already built image the following command should be run in the terminal:

```
docker run -d -p 8050:8050 -e DB_URL=jdbc:leanxcale://147.102.230.182:30001/ihelp
-e DB_USER=app -e DB_PW=app -e DB_DRIVER_CLASS=com.leanxcale.client.Driver
<image_name>
```

2.3 Running with Kubernetes

To deploy the application on a Kubernetes Cluster, Helm Chart is needed to be able to deploy the Helm Chart that is provided inside the repository at the following link `https://gitlab.ihelp-project.eu/ihelp/bounce-mitigation/bounce-mitigation-ui`, and after getting the Helm Chart the only thing left to do, is to run inside the helm folder:

```
helm install <deployment_name> . –values ./values-uprc-prod.yaml
```

3 Components

The main component of the application is the Streamlit User Interface as its shown in Figure 1, but that is just the user interface that let's the user interact with the application, but in the backend part there is also two important components that serve as the data ingestion point for the user interface to display said data. The following components are, in order of the flow of the application:

1. **LeanXcale Database**
2. **Python Cronjobs**
3. **User Interface**

3.1 LeanXcale Database

he LeanXcale Database serves as a cornerstone of the architecture. As a SQL-based database, it houses critical information about patients participating in the program. The database is especially crucial for achieving the objectives of the BMT, as it not only stores patient data but also tracks key metrics such as questionnaire responses and device utilization for each individual.

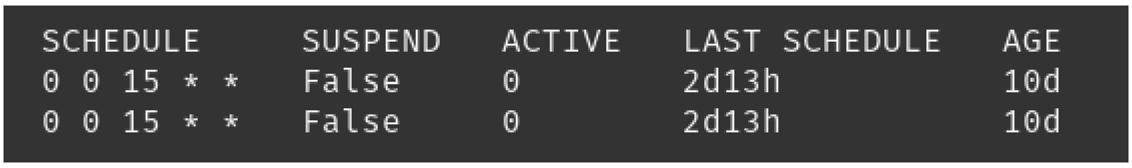
The connection to the LeanXcale Database is unique, requiring a specialized driver supplied by the LeanXcale project team. For Python-based interaction, the JayDeBeApi package is the optimal choice as it is capable of interfacing with the Java-compiled driver. Once connected, SQL queries can be executed by passing them as string commands to the cursor. Additionally, the output of 'SELECT' queries can be seamlessly converted into a Pandas DataFrame, simplifying data manipulation and accelerating the workflow.

The connection to the database can be realized with the following piece of code:

```
    with jaydebeapi.connect(com.leanxcale.client.Driver,
    jdbc:leanxcale://147.102.230.182:30001/ihelp, [<User>,
    <Password>], <path to the driver>) as conn:
```

3.2 Python Cronjobs

These python scripts that are made to run as a cronjob, are responsible with aggregating the data of questionnaire completion and device utilization, and with compiling the percentages for a specific month and year, for every patient that is enrolled in the program.



```
SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
0 0 15 * *    False     0        2d13h           10d
0 0 15 * *    False     0        2d13h           10d
```

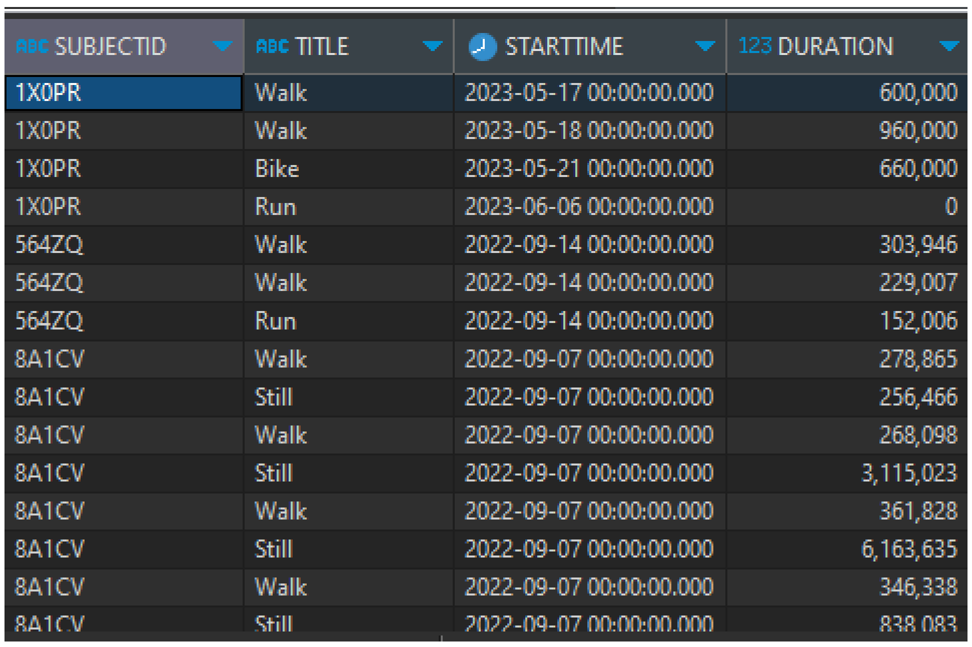
Figure 2: **CronJob Schedule**

Both scripts for the questionnaire part, and respectively for the device usage, have a Dockerfile and a Docker image built, to be run as CronJobs inside a Kubernetes cluster, on the 15th of every month as shown in Figure 2.

The script exclusively utilizes the JayDeBeApi package to connect to the database. Its core component is an SQL statement responsible for both data aggregation and the calculation of resulting percentages. This processed data is then stored back into separate tables within the database. This design choice allows the user interface to simply retrieve and display the data without performing any on-the-fly percentage calculations. By offloading this computational work to the database, the application runs more smoothly, thereby enhancing the overall user experience.

Next the logic behind the SQL statements will be described, for both the questionnaire completion in a month, and for the usage of the devices in a month:

3.2.1 Devices Adherence



SUBJECTID	TITLE	STARTTIME	DURATION
1X0PR	Walk	2023-05-17 00:00:00.000	600,000
1X0PR	Walk	2023-05-18 00:00:00.000	960,000
1X0PR	Bike	2023-05-21 00:00:00.000	660,000
1X0PR	Run	2023-06-06 00:00:00.000	0
564ZQ	Walk	2022-09-14 00:00:00.000	303,946
564ZQ	Walk	2022-09-14 00:00:00.000	229,007
564ZQ	Run	2022-09-14 00:00:00.000	152,006
8A1CV	Walk	2022-09-07 00:00:00.000	278,865
8A1CV	Still	2022-09-07 00:00:00.000	256,466
8A1CV	Walk	2022-09-07 00:00:00.000	268,098
8A1CV	Still	2022-09-07 00:00:00.000	3,115,023
8A1CV	Walk	2022-09-07 00:00:00.000	361,828
8A1CV	Still	2022-09-07 00:00:00.000	6,163,635
8A1CV	Walk	2022-09-07 00:00:00.000	346,338
8A1CV	Still	2022-09-07 00:00:00.000	838,083

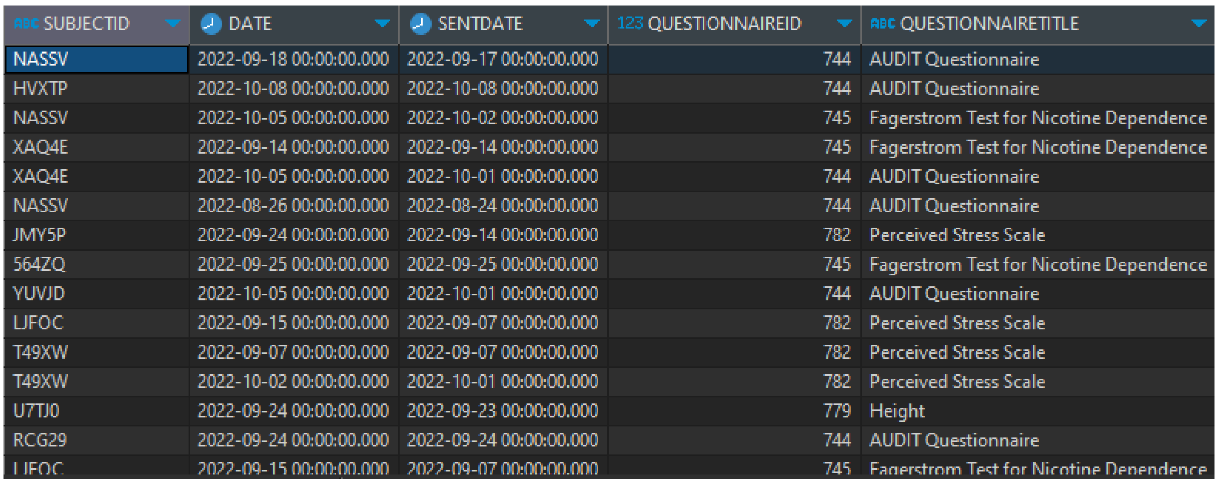
Figure 3: **Device Data**

As shown in Figure 3 the data from devices come based on the type and duration of the activity that was done by a specific patient, and it also includes a very important piece of information about when the patient started the specific activity, which makes it simple to extract all the entries for specific patients in a specific month and year with the following statement:

> SELECT SUBJECTID, TITLE, STARTTIME, DURATION FROM HEALTHENTIA_EXERCISES WHERE MONTH(STARTTIME) = <MONTH> AND YEAR(STARTTIME) = <YEAR> AND SUBJECTID = <PATIENT ID>;

Once patient data is retrieved, the sum function is applied to the 'DURATION' column to calculate the total exercise time a patient has completed within a month. This sum is then converted into hours. The percentage of device utilization is then determined based on a monthly quota, which is set at 14 hours — equivalent to 30 minutes of exercise per day. Depending on how closely the calculated total exercise time aligns with this 14-hour monthly target, a percentage is generated to evaluate the patient's level of engagement in the program.

3.2.2 Questionnaires Adherence



SUBJECTID	DATE	SENTDATE	QUESTIONNAIREID	QUESTIONNAIRETITLE
NASSV	2022-09-18 00:00:00.000	2022-09-17 00:00:00.000	744	AUDIT Questionnaire
HVXTP	2022-10-08 00:00:00.000	2022-10-08 00:00:00.000	744	AUDIT Questionnaire
NASSV	2022-10-05 00:00:00.000	2022-10-02 00:00:00.000	745	Fagerstrom Test for Nicotine Dependence
XAQ4E	2022-09-14 00:00:00.000	2022-09-14 00:00:00.000	745	Fagerstrom Test for Nicotine Dependence
XAQ4E	2022-10-05 00:00:00.000	2022-10-01 00:00:00.000	744	AUDIT Questionnaire
NASSV	2022-08-26 00:00:00.000	2022-08-24 00:00:00.000	744	AUDIT Questionnaire
JMY5P	2022-09-24 00:00:00.000	2022-09-14 00:00:00.000	782	Perceived Stress Scale
564ZQ	2022-09-25 00:00:00.000	2022-09-25 00:00:00.000	745	Fagerstrom Test for Nicotine Dependence
YUVJD	2022-10-05 00:00:00.000	2022-10-01 00:00:00.000	744	AUDIT Questionnaire
LJFOC	2022-09-15 00:00:00.000	2022-09-07 00:00:00.000	782	Perceived Stress Scale
T49XW	2022-09-07 00:00:00.000	2022-09-07 00:00:00.000	782	Perceived Stress Scale
T49XW	2022-10-02 00:00:00.000	2022-10-01 00:00:00.000	782	Perceived Stress Scale
U7TJ0	2022-09-24 00:00:00.000	2022-09-23 00:00:00.000	779	Height
RCG29	2022-09-24 00:00:00.000	2022-09-24 00:00:00.000	744	AUDIT Questionnaire
LJFOC	2022-09-15 00:00:00.000	2022-09-07 00:00:00.000	745	Fagerstrom Test for Nicotine Dependence

Figure 4: **Questionnaire Data**

In Figure 4 is presented the way the data for questionnaire completion is structured, containing for each patient the date when the questionnaire was sent to the patient, and when the date when the patient responded, the table also contains the id and the title of the questionnaire, which helped in the beginning phase when separating patient based on the pilot they belong to.

Based on which pilot, between FPG, MUP, TMU, HDM, UNIMAN, the patient belongs to, they have a monthly quota of questionnaires that are sent monthly to patients and they need to respond to. The quotas per pilot per month are the following:

1. FPG: 8 questionnaires/month
2. TMU: 5/6 questionnaires/month depending if the month is even of odd
3. HDM: 9 questionnaires/month
4. UNIMAN: 21 questionnaires/month
5. MUP: 29 questionnaires/month

After we collected the quotas for each pilot in the program, the calculation of the percentage, that assesses the commitment, became relatively simple, the only thing that needed to be done was to calculate the total number of questionnaires that were responded by a specific patient for a chosen month and year with the following SQL statement:

```
SELECT SUBJECTID, (COUNT(*) / CAST(MAX(<Quota based on pilot>) AS FLOAT))
* 100.00 PERC FROM ( SELECT DISTINCT SUBJECTID, QUESTIONNAIREID, SENT-
DATE FROM HEALTHENTIA_ANSWERS WHERE MONTH(SENTDATE) = <Month> AND
YEAR(SENTDATE) = <Year> AND SUBJECTID = <Patient ID>; )
```

The above statement counts all the response that were sent in a month by a specific patient and then divided by the pilot specific quota, and then multiply it by 100 to normalize the result as a percentage. Subsequently, this calculated percentage, along with other identifying information such as the subject identifier, month, year, and pilot program, are stored in a separate table. This streamlined data structure facilitates easier access and usability for the user interface.

As mentioned earlier, both scripts are scheduled to run as cronjobs on the 15th of each month, focusing on data aggregation for the prior month. The reason for this 15-day buffer between the start of the new month and the execution of the jobs is to allow ample time for all data from questionnaires and devices to be properly stored in the database. This process is not instantaneous and requires some time to accurately distribute the data across the relevant database tables.

3.3 Streamlit User Interface

As noted at the outset, the user interface (UI) is developed in Python, utilizing the Streamlit framework. This package streamlines the deployment of data-centric applications, even for those with minimal coding experience, while still harnessing the full capabilities of Python for data manipulation. The framework includes a variety of pre-built components, such as selection boxes, input fields, graphs, tabs, and tables that can display Pandas dataframes, significantly cutting down the time required to build an application. Below, we outline how a Streamlit component is instantiated and integrated into the application.

```
st.sidebar.selectbox( 'Select the Maximum Percentage', ('100%', '90%', '80%', '70%', '60%',
'50%', '40%', '30%', '20%', '10%'))
```

Each component of a streamlit application has multiple attributes that can be modified in order to make the component work in a specific way, also the components can be placed in specific ways, in the sidebar or inside the main page, to achieve desired look for the application.

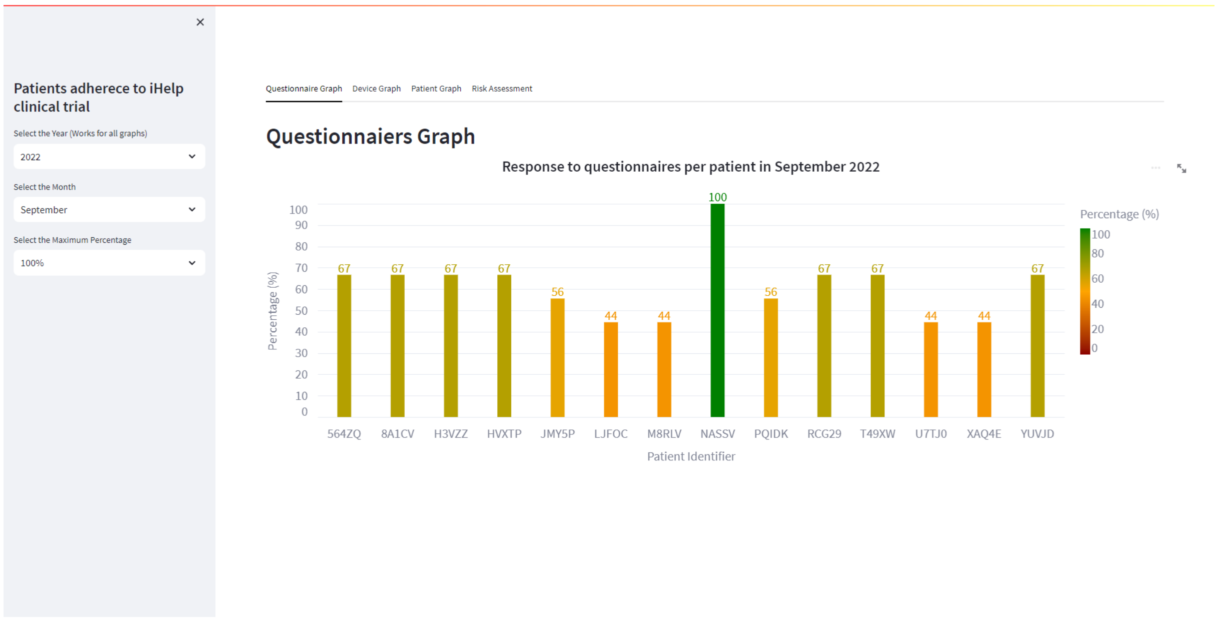


Figure 5: **User Interface**

The BMT UI consists of 4 tabs, as presented in Figure 5, each representing different aspects of the application, and on the sidebar some selectboxes, as shown in Figure 6 that are used for the selection of the month, year and maximum percentage, that effects which data is displayed on the different graphs.

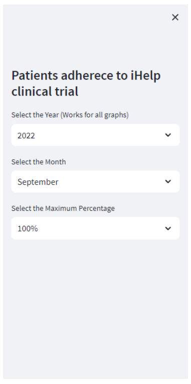

×
Patients adherece to iHelp clinical trial
Select the Year (Works for all graphs)
2022
Select the Month
September
Select the Maximum Percentage
100%

Figure 6: The Sidebar

Further the meaning and function of each tab will be presented:

3.3.1 Questionnaire Tab

In this tab is presented the graph that displays, based on the parameters that are selected in the side bar, the percentages for the questionnaire completion for each patient that is enrolled in the program. A 100% means that that patient responded to all the questionnaires that were sent to him/her, and a 0% means that the patient didn't respond to any of the questionnaires that were sent.

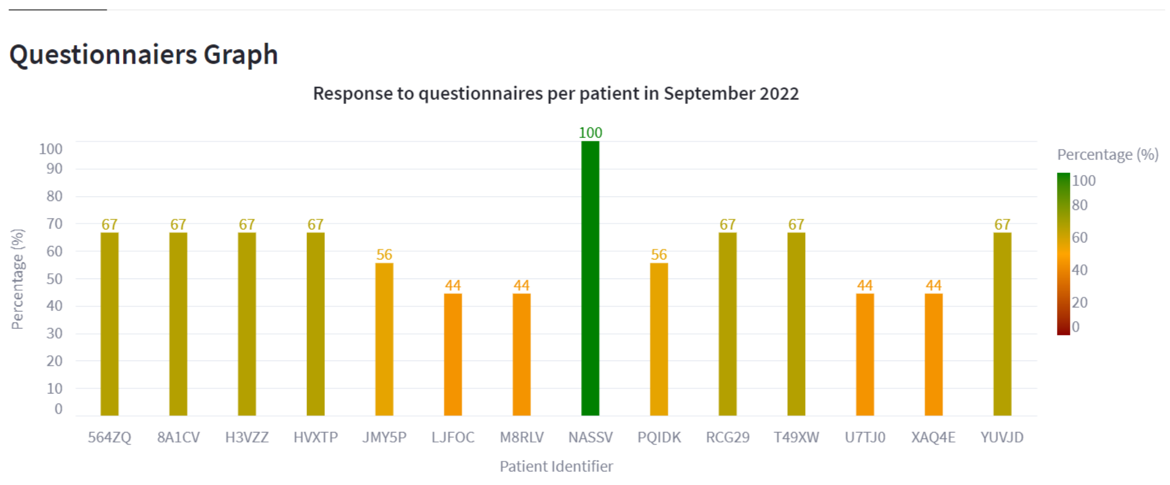

Questionnaiers Graph

Figure 7: Questionnaire Tab

The user that interacts with this graph can select, with the selectbox from the sidebar, the maximum percentage that is allowed to be displayed on the graph, making it so it can see only the patients that have a low response rate and sending them a remainder to respond to the questionnaires.

3.3.2 Device Tab

The "Device" tab functions similarly to the "Questionnaire" tab, but instead of displaying questionnaire results, it shows device usage percentages for each patient enrolled in the program. Just like in the "Questionnaire" section, the data presented can be customized using the selection boxes in the sidebar. If a patient's device usage percentage falls below 50%, clinicians have the option to send a reminder encouraging increased utilization of the program-provided devices.

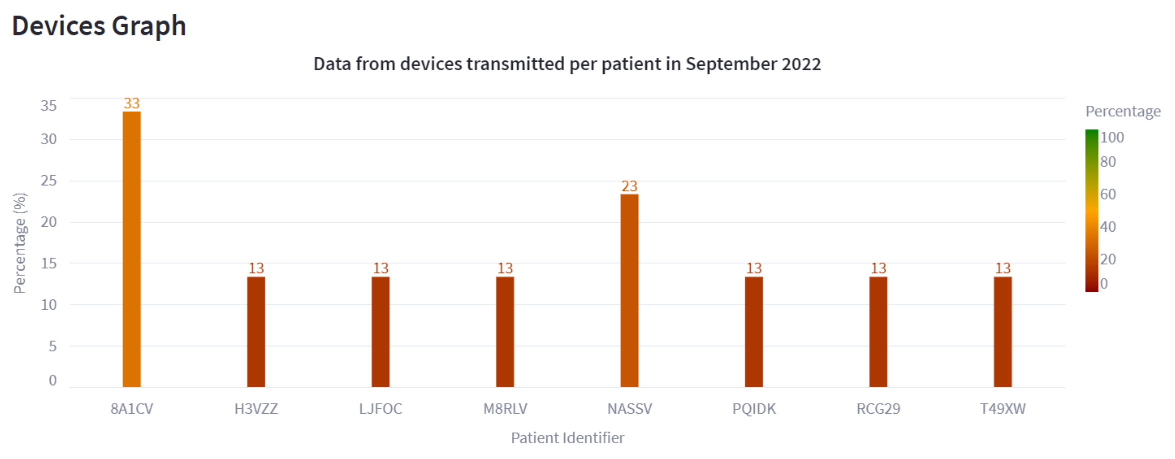


Figure 8: Device Tab

3.3.3 Patient Tab

Inside this tab are two graphs, as shown in Figure 9, one for the devices and one for the questionnaires, that presents the evolution throughout a year for a specific patient that can be selected using the patients selectbox and the beginning of the tab. With this graphs someone can asses the implication of that patient in the program, and if the overall performance of that patients does not exceed 50% in both graphs, then the patient should be excluded from the program for not showing enough interest in it.

The data in this graphs as been obtained by running the following SQL statement on the tables that contains the percentages:

```
SELECT MON, PERC FROM DEVICE_ADHERENCE WHERE SID=<Patient ID> AND YR=<Year>;
```

```
SELECT MON, PERC FROM QUEST_ADHERENCE WHERE SID=<Patient ID> AND YR=<Year>;
```

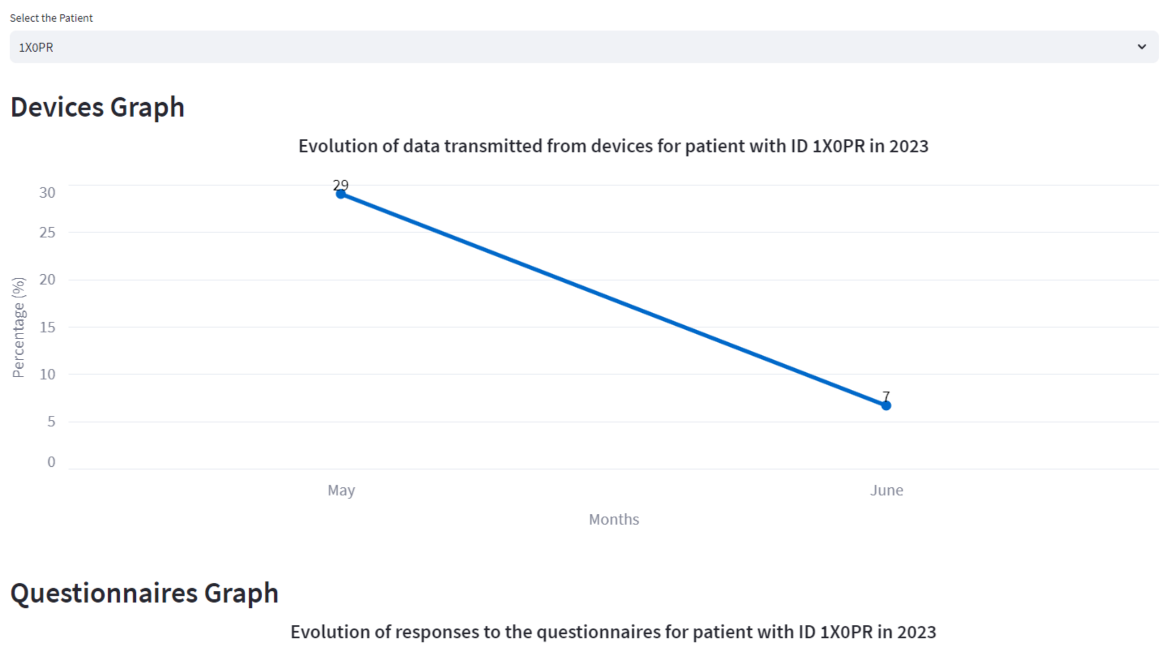

Select the Patient
1X0PR
Devices Graph
Evolution of data transmitted from devices for patient with ID 1X0PR in 2023
30
25
20
15
10
5
0
Percentage (%)
29
7
May
June
Months
Questionnaires Graph
Evolution of responses to the questionnaires for patient with ID 1X0PR in 2023

Figure 9: Patient Tab

3.3.4 Risk Assessment Tab

The risk assessment tab it is the main piece of the application, for which contains the aggregation of the percentages from both questionnaires and devices data, based on the following formula:

0.7 * questionnaire_percentage + 0.3 * device_percentage

The graph on this page is a heatmap that presents on the x-axis the month of the year, on the y-axis all the patients enrolled in the program, and for the z-axis or the color axis the percentages calculated with the formula shown above was chosen, so this graph represents a culmination of all the graphs, but is for all patient and is just to glance over at a high level to see if a patient did not participate, and after this to check that specific patient in the previous tabs, to get an in depth analysis about the patients, and draw conclusions on what is needed to be done in the future.

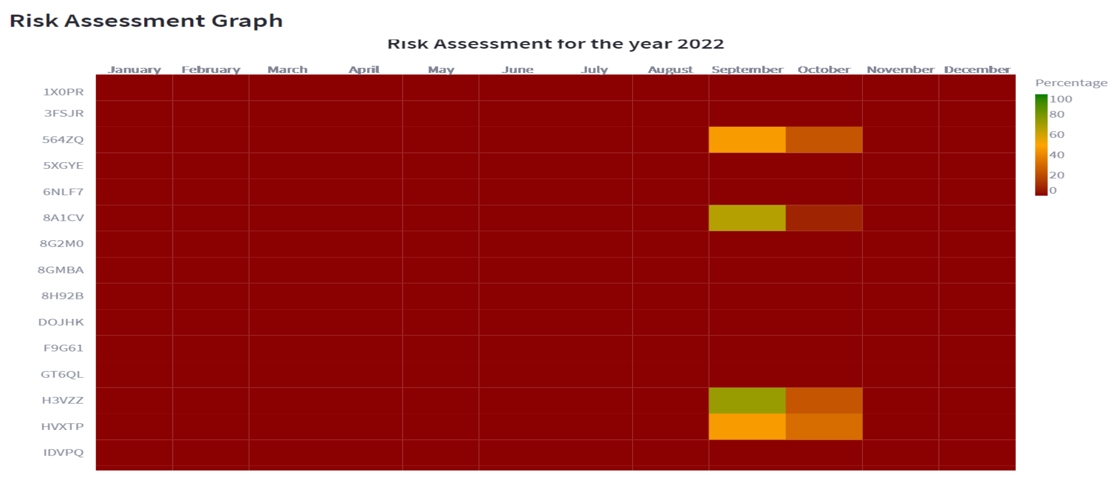


Figure 10: Risk Assessment Tab

References

[1] https://www.docker.com/

[2] https://kubernetes.io/

[3] https://helm.sh/

[4] https://www.python.org/

[5] https://github.com/baztian/jaydebeapi

[6] https://pandas.pydata.org/docs/getting_started/index.html

[7] https://streamlit.io/