



**Politecnico
di Torino**

Politecnico di Torino

Master's Degree in ICT for Smart Societies (ICT4SS)
A.a. 10/2022

IBD Tool: Improvement in a web application for the monitoring of patients with Inflammatory Bowel Diseases

Relatori:

Prof. Michela Meo
Prof. Guido Pagana

Candidati:

Reihaneh Mehdizadeh Baroughi

Abstract

The IBD (Inflammatory Bowel Disease) tool has been designed by Mauriziano Hospital in Turin and the Links Foundation to develop a web application to support monitoring patients suffering from IBD. IBD describes two ailments: Crohn's Disease (CD) and Ulcerative Colitis (UC), that is characterized by chronic inflammation of the Gastrointestinal (GI) tract, and they have different symptoms such as abdominal pain, diarrhea, rectal bleeding, and weight loss. IBDs are idiopathic ailments, and alleviating patients' pain, and limiting the degeneration of pathology, is an effective treatments for this disease. Patients suffering from IBD experience two conditions,

- Periods of remission of the illness, with fewer symptoms detected
- Periods of relapse, with heavier symptoms.

The follow-up process is an important phase of the treatment, which means constant monitoring of the patient's health status, which has to send physicians intermittently the effects of the disease and the degree of influence on their daily activities. In this scenario, telemedicine is a perfect solution for the follow-up process, facilitating the interaction between patients and doctors to better follow the medical guidelines, with no need to go to the hospital when unnecessary. IBD tool is designed to provide an effective monitoring system to link patients and doctors. This web app has detailed questionnaires to help patients describe their ailment symptoms to their doctors in a swift and well-organized manner. The monitoring process is performed by questionnaires scheduled and sent automatically by the platform. So, patients can fill out these questionnaires periodically; however, it is essential to notify them that they can complete a questionnaire at any time when they need to manually. A notification is sent to physicians whenever a patient compiles a new questionnaire, and physicians can assess the details and evaluate the questionnaire results. An elaborated and fast chat system is also designed to facilitate the interaction between physicians and patients, allowing them to send and receive instant messages and clinical files using a dedicated chat. This research focuses on enhancing the platform architecture, boosting its overall efficiency by providing more scalability, and developing novel functionalities to supplement the platform further. The platform has been improved on the patient side in this research work. Problems related to page loading on different servers have been solved, making the web app even more user-friendly. A new questionnaire (LARS score) was also added. It is a 5-item scoring tool that measures bowel dysfunction after restorative surgery. Through this questionnaire, it is possible to validate the use of the IBD Tool in the subgroup of patients with operated UC, providing a new means of monitoring the subset mentioned above of patients. One of the objectives of using the platform is to make the patient feel more involved and not alone; to pursue this goal, the patient section has been implemented with personalized news about the pathology the patient suffers.

Acknowledgements

This master's thesis is the culmination of my education at Politecnico di Torino in pursuit of an MSc in ICT for smart societies. The experience I had while writing my thesis was both challenging and intellectually stimulating. This task could not have been accomplished without several individuals' extensive and persistent support and guidance. Before anything else, I would like to express my gratitude to professors Meo and Pagana, my supervisors at Politecnico di Torino, for their assistance in devising the approach to assess the objectives of my study. In addition, I would like to thank my daily supervisor, Dr. Valeria Figini, for taking time out of her busy schedule to provide feedback on my work and aid me in finding solutions to the various problems I encountered throughout the construction of the experimental setup.

Reihaneh Mehdizadeh Baroughi

Table of Contents

List of Tables	V
List of Figures	VI
Acronyms	IX
1 Introduction	1
2 Concepts definition	4
2.1 Diseases	4
2.1.1 Crohn's Disease	4
2.1.2 Ulcerative Colitis	6
2.2 Medical treatment for IBD	8
3 Materials Methods	11
3.1 Web development	11
3.1.1 Front end	11
3.1.2 Back end	15
4 Result and Discussion	17
4.1 IBD tools features	17
4.2 New updated on the IBD tools	40
4.3 Dashboard Evaluation	41
5 Conclusion and future work	46
Bibliography	48
A Appendix	54

List of Tables

3.1	Top 7 Programming Languages for developing back-end	15
-----	---	----

List of Figures

1.1	Trilateration	3
2.1	CD explanation [27]	5
2.2	CD explanation [35]	7
3.1	Most used programming languages among developers worldwide as of 2022 [56]	14
4.1	Registration page	18
4.2	Physicians Home page	19
4.3	Personal patients section	20
4.4	Registration of new patient	21
4.5	Patient's personal data	21
4.6	Questionnaire and periodicity management	22
4.7	The received compiled IBDQ questionnaires	22
4.8	The result of IBDQ questionnaires in physician side	23
4.9	Global patients section	24
4.10	Avvisi section	25
4.11	The number of total patients per type of disease, updated to the day:10/10/2022	26
4.12	Number of total active and inactive patients, updated to the day:10/10/2022	26
4.13	Number of total active and inactive patients, updated to the day:10/10/2022	27
4.14	Stable-improving-worsening patient rate chart for UC patients, updated to the day:10/10/2022	28
4.15	Stable-Improving-Worsening patient rate chart for CD patients, updated to the day:10/10/2022	28
4.16	Export data section	29
4.17	Home page of patients	30
4.18	Questionnaires are due to compile	31
4.19	Compiled questionnaires section	31
4.20	Completed questionnaire summary with status and responses	32

4.21 Interact with the doctor sections	34
4.22 List of possible questionnaires	35
4.23 Example of the chat page	35
4.24 Content of the questionnaires page for the doctors	37
4.25 Received questionnaires page	38
4.26 Questionnaire's result	39
4.27 Questionnaire's management	40
4.28 First question	42
4.29 Second question	43
4.30 Third question	44
4.31 Fourth question	45

Acronyms

CD

Chronic Disease

CSS

Cascading Style Sheet

DW-MRI

Diffusion-weighted MRI

HTML

Hypertext Markup Language

ICT

Information and Communication Technology

SCCAI

Simple Clinical Colitis Activity Index

SASS

Syntactically amazing style sheets

UC

Ulcerative Colitis

UI

User Interfaces

UX

User Experience

WCE

Wireless Capsule Endoscope

Chapter 1

Introduction

Nowadays, technology has been contributing to solving relevant issues by considering cutting-edge high tech. Different interdisciplinary fields of study and science can benefit from the technology topics. Recent research indicates the participation of several researchers in considering web applications to communicate between system and user by providing various features. For example, a patient management system has been developed within the hospital environment, the system benefits from two different User Interfaces (UI) to able final users for accessibility [1]. Moreover, Akbarzadeh [2] considered the mobile application, which is able to measure social distance; this tool is developed for severe pandemics such as COVID-19 to manage the patients' distance inside the hospitals. Healthcare is a topic requiring significant attention since it reduces the risk of decreasing the death rate in society and provides a novel solution for doctors to prevent the degeneration of the diseases. The role of technology in healthcare has been investigated in several research. Theses studies have assessed the applicability of different technologies in the healthcare sector and the challenges to be faced in order to fully exploit their potential [3], [4], [5], and [6]. Information and Communication Technology (ICT) are defined by Pratt et al. [7] as a combination of devices, networks, and applications to able users to interact in the modern world. ICT offers great promise to help developed and developing countries overcome difficulties in delivering affordable, accessible, high-quality healthcare services. Telemedicine, which is multidisciplinary [8], was described by Jamal et al. [9] as using electronic information to offer and assist healthcare while participants are physically separated. An example of the Telemedicine application can be seen in the research developed by P. de Toledo et al.[10], L. Kapoor et al.[11], and L. Su et al., [12]. They examined the various fields of application both from the point of view of the technologies for which telemedicine can be adopted, and with regard to the possible problems related to the use of these technologies and the possible strategies to overcome them. ICTs are used in Telemedicine to get beyond geographic limitations and enhance access to healthcare treatments. Rural and

underserved areas in developing nations that typically struggle with a lack of access to healthcare will benefit significantly from this [13]. Telemedicine is relevant to four factors,

- It is purpose is to provide clinical support.
- It is intended to overcome geographical barriers, connecting users, not in the exact physical location.
- It involves the use of various types of ICT.
- It is goal is to improve health outcomes.

Telemedicine is the delivery of healthcare services where distance is a significant barrier; all healthcare professionals can use ICTs for the exchange of accurate data for the diagnosis, treatment, and prevention of disease and injuries, research and evaluation, and the continuing education of healthcare professionals, all to enhance the health of people and their communities [14]. Telemedicine's benefits are enormous for the patient; it is more convenient. Virtual telemedicine visits allow patients to connect with their providers via a secure zoom meeting or an application. The development of telemedicine tools enables the discovery of novel solutions to long-standing medical issues and novel features that will enhance the care provided by fostering better cooperation between patients and healthcare professionals [14]. Therefore, technological advancement is promoting the use of extensive Telemedicine and pushing more toward improving health care. Telemedicine has become more prevalent in Italy due to the Covid-19 pandemic, with clinicians usage rising from 21% to 47% [15].

Telemedicine complements and integrates conventional medicine using new technology and communication channels, not replacing it. The telemedicine service that is the focus of this thesis is intended for people with IBD. Among the IBD, the two most frequent are CD and Ulcerative Colitis (UC).

The two diseases are not easy to distinguish as they have many similar symptoms. It is difficult to determine the precise reasons for IBD, but in general, it is forced by immune dysregulation, which means that the immune system is overreacting [16]. IBDs affect around 200,000 patients in Italy; the statistics show that this number is increasing. CD may develop at any age, although it is more prevalent in older adults. Often diagnosed in adolescents and adults varies around ages 20 and 30 [17].

IBD-Tool was born from this assumption. It is a web platform that aims to offer an efficient telemedicine service through periodic monitoring of patients suffering from chronic intestinal diseases. The physician can analyze the disease state by monitoring the patient's condition through questionnaires. The tool was developed at the LINKS Foundation research center with the collaboration of the

Gastroenterology Department of Mauriziano of Turin Hospital and is currently aimed exclusively at related patients at the Mauritian hospital in Turin.

In order to increase the efficiency of the provided telemedicine service, this research aims to enhance the features already present in the application and provide new tools. From a practical perspective, the first section was devoted to analyzing and improving the programming structure and code. In the second part, we moved on to the development and implementation phase of the questionnaires section through the digitalization of a new questionnaire (LARS score) for the patient with UC. In the end, the patient department has built a system that allows patients to get relevant information depending on their pathology in order to raise their awareness.

This research built the front end by relying on the Angular framework, while the back end utilized the Spring framework. Furthermore, MongoDB is the database used for data collection and processing. Fig. 1.1 represents an overview of the IBD tools, which contains the Front end, Back-end, and Database.

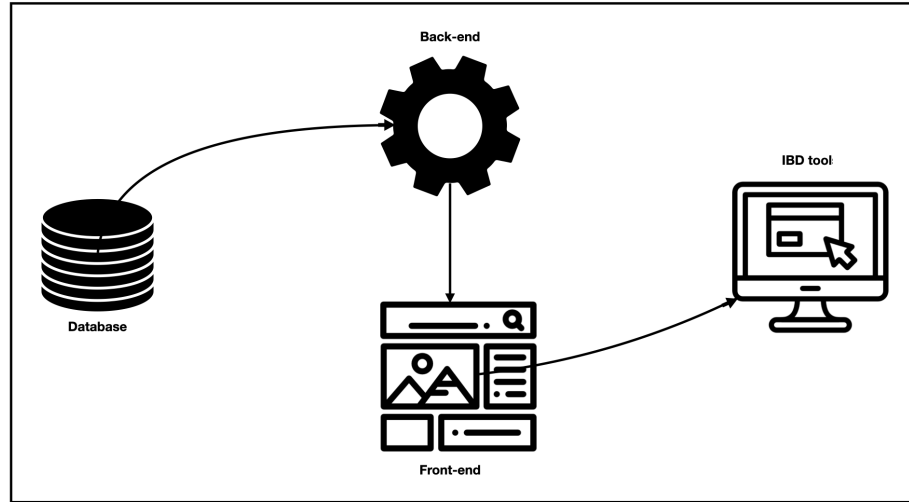


Figure 1.1: Trilateration

Chapter 2

Concepts definition

With the growing utilization of technology in various sectors, it is necessary to analyze all the possibilities for healthcare concerns in order to develop a data-driven system to aid physicians in illness detection. Not only are diseases with symptoms vital to examine, but diseases without symptoms are equally crucial to investigate. For example IBDs are pathologies with alternating periods of absence and the presence of symptoms.

However, developing technology-based tools is the responsibility of technology researchers and developers; comprehending the differences between each disease can be helpful. The section 2.1 will overview two ailments described by IBD. Then in section 3.1, three phases of web development are considered to understand the different usage and application of programming languages and it answers to the question of why a specific programming language is selected for the front and back end that is vital in which the assumption is that the final users of the web tools will be the different users with different level of knowledge in the case of using the Web tools; therefore the satisfaction of them is too and will explain the role of the selected Database, which is one of the vital parts of Web development since it stores all the data and it should be provided a secure option.

2.1 Diseases

2.1.1 Crohn's Disease

CD is a chronic, relapsing, and remitting IBD [18], for which there is no cure, but steroid pills, injections, and surgery to remove a small portion of the digestive tract may help manage symptoms[19]. Several research has been done in the area of study in the CD. For example, [20] used Gene Set Variation Analysis or, in short, GSEA and made a comparison in IBD of various mice. [21] considered

diffusion-weighted MRI (DW-MRI) and mentioned that it is one of the significant biomarkers in order to identify the area with restricted diffusion like IBD. [22] idea is related to using the Bayesian network to research patients with CD who have early surgery or take immunosuppressants can know if they will become disabled or need surgery again. Also, the contribution of Machine Learning can be seen in other research like [23], which compares Machine Learning methods to evaluate them in predicting CD. [24] used the concept of the Wireless Capsule Endoscope (WCE) and provided a new approach by considering the recorded images. Not only Machine Learning but also Deep Learning also contributed to the study related to the CD; such as [25] used Histopathological Imaging for Predicting Pediatric CD. [26] is another study in the area of research in CD by using the concept of Deep Learning.

Crohn's Disease

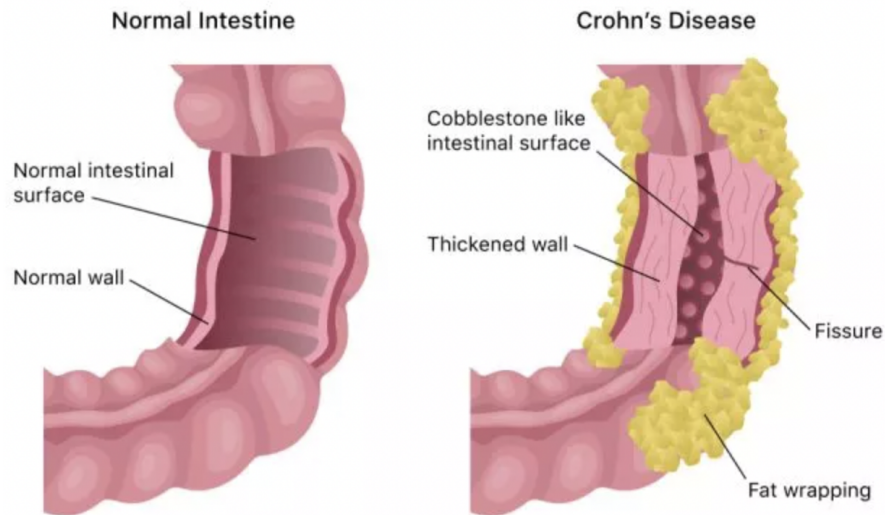


Figure 2.1: CD explanation [27]

Fig. 2.1 [27] compares two different pictures of Intestine, in which the left picture represents the Normal one, and the right image shows the CD. Fig. 2.1 shows that Fatwrapping reflects a portion of the connective tissue alterations that accompany CD of the intestines [28]. The thickened wall can be seen in the left center of figure 1, which compares to the Normal wall in the same position in the left picture. Cobblestone and fissure are other parts of the CD which are represented.

2.1.2 Ulcerative Colitis

The colon and rectum become inflamed due to a chronic illness known as UC [29]. This disorder is the outcome of an excessive immune system reaction [30]. [31] summarized symptoms of UC, which include:

- Diarrhea
- Rectal bleeding
- Rectal pain
- Weight loss
- Fever
- Fatigue

[32] T. Kadota et al. considered the multi-task learning approach in order to assess illness severity by evaluating learning to rank and regression. [33] H. Nosato et al. research demonstrates the approach, which uses image recognition methods and multi-discriminant analysis that describes objective evaluations for UC. [34] Sutton et al. examined the performance of Artificial intelligence in automated diagnosis of UC. Fig. 2.2 compares normal clone and UC [35].

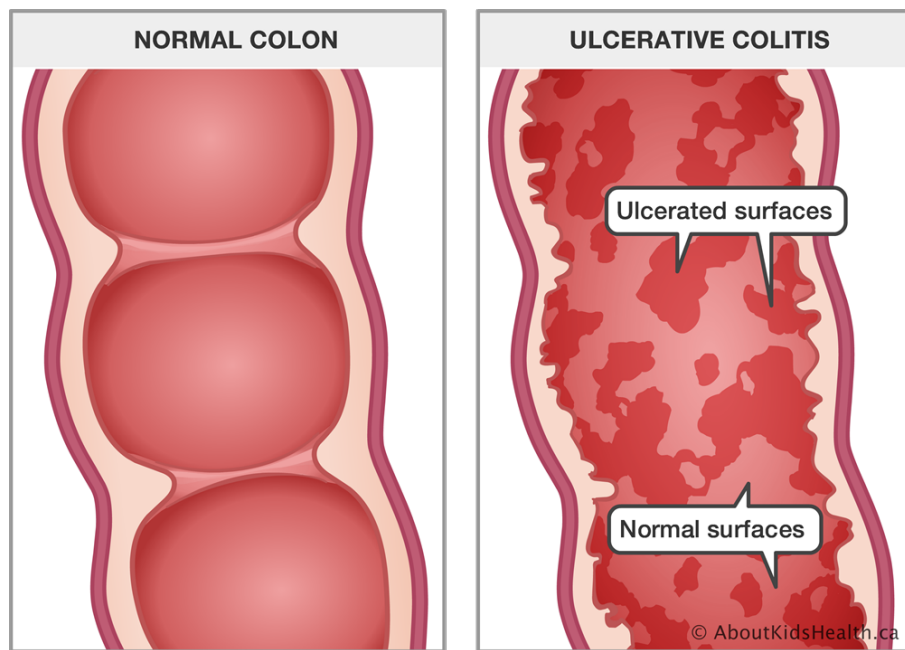


Figure 2.2: CD explanation [35]

2.2 Medical treatment for IBD

There are three goals of IBD, and they have to do with these relapses and remissions, so the first goal is to induce remission. The second goal is to maintain remission, and the last goal is to prevent and treat complications. It is helpful to think about the mechanism of IBD as having three steps. CD and UC are incurable as a result of IBDs. Medical therapies aim to get patients' symptoms to reduce until they disappear. For people with IBD, there is no conventional course of treatment; instead, individual patient therapies vary. For those with IBD, there are five primary medications [36]:

- **Antibiotics:** When abscesses develop in CD, antibiotics can be used to treat them. The use of antibiotics in the treatment of UC is not practical.
- **Aminosalicylates:** Aminosalicylates are anti-inflammatory substances that may be taken orally or rectally to reduce inflammation at the intestine's wall. They are typically employed to treat UC and lessen its symptoms. Ineffective in the treatment of CD.
- **Biologic therapies:** For patients with moderately to highly active IBDs, biologics and biopharmaceuticals are employed as a form of therapy.
- **Corticosteroids:** This kind of medication affects how the body responds to inflammation. The most typical application of corticosteroids is the management of flare-ups. Due to the potential adverse effects of corticosteroids, which might include infections, bone loss, cataracts, skin fragility, and sleep disruption, long-term administration of this therapy is uncommon.
- **Immunomodulators:** Immunomodulators alter the immune system's function to decrease the symptoms of inflammation.

Decreasing the inflammation that leads to the signs and symptoms of IBD is the aim of treatment. In ideal scenarios, this may reduce the risk of complications, long-term remission, and symptom alleviation. Surgery or medication therapy is often used to treat IBD [37].

Numerous research on the prevalence and diagnosis of IBD in recent years have indicated an increase in the number of IBD patients in Western nations, especially in Europe and North America. Three million individuals in North America, or 1.3% of all adults, received a diagnosis of Ulcerative colitis or Crohn's disease in 2015[9]. According to the data, there are now more individuals with IBD diagnoses than in 1999, when there were 2 million adults in the United States or about 0.9% of the total population[38]. The prevalence of CD and UC has also grown in Europe. Incidence rates of UC were 6.0 per 100,000 people per year, and CD was 1.0 per

100,000 people per year in 1962. The incidence rate of UC climbed to 9.8 per 100,000 people per year in 2010, whereas the incidence rate of CD increased to 6.3 per 100,000 people per year [38]. These findings suggest that the increased prevalence of IBD may be related to people's way of life in western nations. In order to offer a scalable solution to deliver the necessary treatment in the coming years, this developing tendency must be appropriately identified and observed. This problem, for its potential to provide healthcare that is more effective, affordable, and scalable[39].

The use of different communications tools for medical purposes, such as the phone, video calls, texts, emails, patient portals, and social media, is referred to as telemedicine. The ability to communicate in real-time or see and hear each other on video via several methods is available to both patients and physicians. For instance, users can save time and money by not having to travel or take time off from work by having a healthcare practitioner follow up via a telehealth session rather than an in-person office visit. Some physicians may employ telemedicine exclusively in specific situations or demand a specific percentage of in-person visits relative to online appointments. For those with special medical requirements, telehealth could be an affordable option. For instance, IBD may be difficult to treat, and patients frequently see their doctors manage flare-ups and prevent them from occurring. More frequent check-ins with patients with chronic illnesses like IBD may help healthcare systems avoid needing more intensive treatments. For people who reside in remote locations, telemedicine may be beneficial. Not only are there no IBD facilities or specialists nearby in certain areas, but it could take considerable travel to see a gastroenterologist. In this case, telemedicine can provide the team with much-needed knowledge by adding an IBD specialist[40].

Annually pilot study including patients with ulcerative colitis was one of the initial research on telemedicine in IBD. While some patients reported an increase in their quality of life and found telemedicine beneficial, some patients found the system challenging to use. Overall, the researchers concluded that telehealth has a chance of being employed to treat IBD [41]. Telemonitoring of Crohn's Disease and Ulcerative Colitis named TECCU, it is a specific web-based system, was the subject of another Spanish experiment. Sixty-three patients were enrolled and randomly assigned to get their usual care in a clinic, use TECCU, or receive nursing care over the phone. Researchers discovered that using TECCU resulted in both better treatment and lower costs. However, they admitted that more extensive research is required before the techniques may be used on a broader scale [42]. Telemedicine may have benefits for patient education. IBD is complicated, and many individuals have a learning curve when diagnosed. One modest trial evaluated the effectiveness of a web-based treatment center to a patient education center over six months for 21 individuals with ulcerative colitis. The findings indicated that web-based training "substantially" boosted understanding of IBD [43]. There were 95 patients with

mild to moderate ulcerative colitis in an enormous web-based learning experiment. Patients were treated for ulcerative colitis with mesalazine and had once-weekly fecal calprotectin (an indicator of inflammation) checks. Patients were asked five questions regarding their ulcerative colitis symptoms as part of an online survey known as the simple clinical colitis activity index (SCCAI), which was used to gauge the severity of the condition. The severity of the illness is correlated with the score. Eighty-six percent of the patients finished the three-month treatment. Throughout the trial, patients' SCCAI scores and fecal calprotectin levels decreased, according to researchers [41].

Chapter 3

Materials Methods

3.1 Web development

Web development is a way to make the possible interaction between users and the system. There are three different aspects considered in this thesis in order to make the interaction possible. The first is the Frontend, defined as all the components that users see, like fonts, colors, menus, and others [44]. Developing correct and eye-catching UI make the website more attractive to the users. The following important concept is the backend that represents the behind scene of the website, like the server, database, and other section [45], which is not visible to users. One of the significant points is that working with the website should be possible for all visitors with different levels of knowledge. The last concept is a database, which is essential since data is available and accessible in the database instead of local storage [46].

3.1.1 Front end

Front end can be used as the structure making of the website. Different programming languages can be used in order to make and design the UI of the website. [47] summarized all the programming languages and frameworks, including,

- Hypertext Markup Language (HTML)
- CSS (Cascading Style Sheet)
- JavaScript
- AngularJS
- React.js

- jQuery
- SASS (Syntactically amazing style sheets)
- Flutter

HTML is a standard language for creating web pages. It enables the construction and organization of sections, paragraphs, and hyperlinks using HTML elements (the building blocks of a web page), like tags and attributes [48].

Technically, CSS is not required, but it definitely would not want to see a web page with just HTML components since it would appear quite empty [49].

JavaScript is one of the text-based programming languages considered by both the client and server sides to create interactive web pages. HTML and CSS are languages that provide structure and style to online pages, but JavaScript adds user-engaging interactivity to websites.

Everyday instances of JavaScript use include the search box on Amazon, an embedded news recap video on The New York Times, and refreshing Twitter feed [50].

According to [51], AngularJS is a framework for building dynamic web applications, which allows operating HTML as the template language to enhance its syntax to represent the application components and concisely; In this case, the data binding and dependency injection features of AngularJS remove a large portion of the code that would have to write. Moreover, the whole process takes place inside the browser, making it an excellent partner for any server technology.

[52] provided information regarding React.js, which Facebook and an open-source developer community administer the React JavaScript-based UI development package. It can be said that React is not considered a language, but it is a library and is frequently utilized in web development. Since its debut in May 2013, it has become popular to operate frontend libraries for web development. React supports many architectural support extensions, such as Flux and React Native, that extends beyond UI.

A JavaScript library is comprised of JavaScript files with different functionality. A large number of JS libraries, such as React, may fulfill a particular role on the client or server side. Today, the jQuery Library is undoubtedly the most popular JS library available. It is simple to customize and use. In addition, jQuery has a big community that provides vast learning materials, tutorials, and documentation. The most excellent aspect of jQuery is its compatibility with other JavaScript libraries and plenty of plugins for scaling its functionalities. It is incompatible with all other programming languages [53].

Sass is an extension of CSS that permits the usage of variables, nested rules, inline imports, and other features. It also facilitates the organization and expedites the creation of style sheets. Sass is compatible with all existing CSS versions [54].

Google developed and published Flutter in May 2017 as a free, open-source mobile UI framework. In a few words, it enables the creation of native mobile applications from a single codebase. This implies that may design two specific applications using the same programming language and codebase for both iOS and Android [55]. According to the [56], JavaScript stands in first place among the most used programming languages in 2022, while second place is the HTML/CSS by nearly 55%.

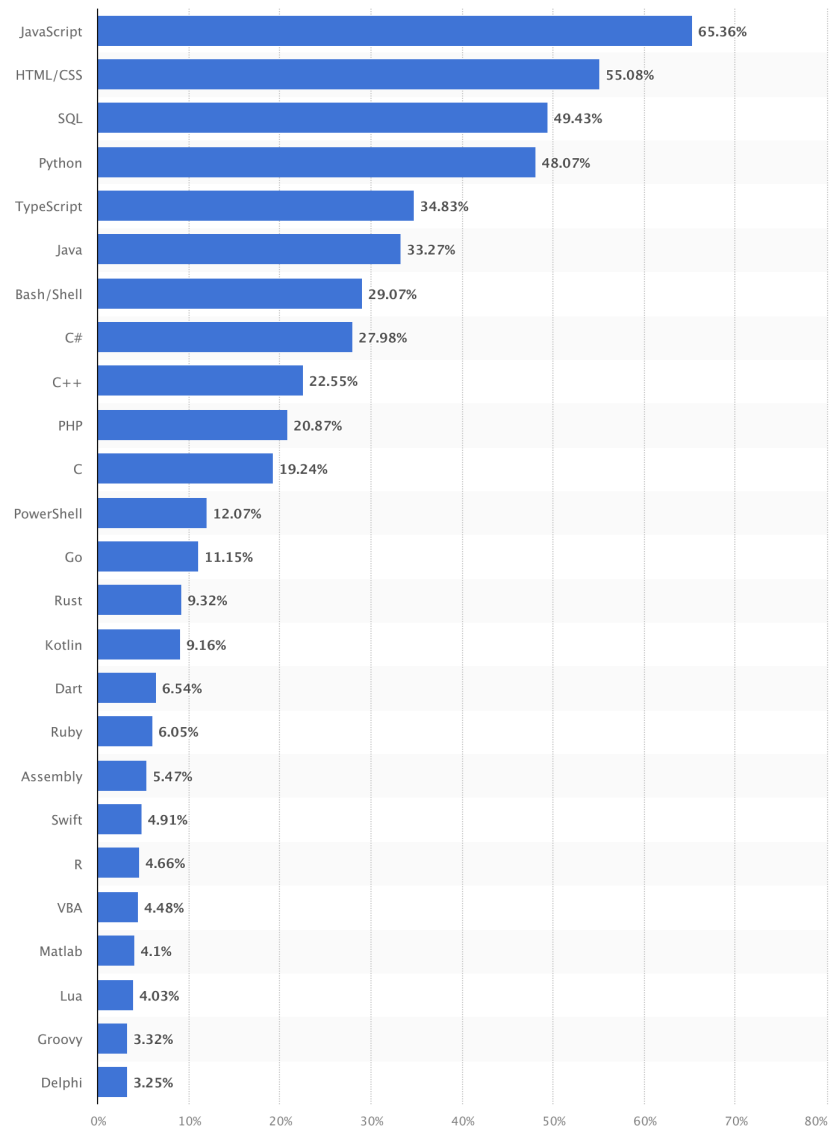


Figure 3.1: Most used programming languages among developers worldwide as of 2022 [56]

3.1.2 Back end

Each web application comprises a front end and a back end. The front end is responsible for offering the most appropriate design and functionality depending on the needs of the end users. The back end is responsible for building the dashboard's features and defining how the items defined in the front-end phase operate.

Top seven of the programming languages which are most appropriate to develop the back-end of the website showed by Table 3.1 with developed examples [57].

Table 3.1: Top 7 Programming Languages for developing back-end

Programming Languages	Example
JavaScript	Facebook, Google, and eBay,
Python	Spotify, Pinterest, and Instacart
PHP	WordPress, MailChimp, and Flickr
Java	LinkedIn, IRCTC, Yahoo
Ruby	Airbnb, Shopify, and Slideshare
Golang	Dropbox, SoundCloud, and Dailymotion
C#	GoDaddy, Marketwatch, and Stack Overflow

The selection of the final programming languages may be influenced by personal experience, the availability of a backend with robust functions, and the ability of the front end to adapt to give seamless service to the end users. Additionally, each programming language for backend development has unique characteristics.

- Python is renowned for its ease of use, yet it is also one of the most adaptable computer languages. However, it is not recommended for large-scale applications because to its lack of memory allocation and inadequate security.
- PHP is among the oldest programming languages and is notoriously difficult to master. In addition, a high degree of coding expertise is required to use PHP, since non-optimized code considerably increases the time complexity. However, it is a free source that offers a high degree of security, and its built-in Database connection facilitates a rapid and seamless connection to the Database.
- Java is another language used for backend development; since it is object-oriented and easy to use, it appeals to backend engineers. However, it is not the greatest solution for real-time applications such as IPS since it is sluggish and increases the execution time of code.
- Ruby offers a high degree of security, is object-oriented, and can be implemented on a number of platforms. Ruby has the same difficulty as Java; it is sluggish and cannot be used in IPS.

- Google provides the programming language Golang. It is quick and has a nice UI and simple syntax. However, limited library support hinders the ability of other developers to change it. One of the most important aspects of UI creation is that it must be customized over time. Once a developer has created the backend, it is tough for other developers to modify it.
- C# is one of the swift programming languages that supports a variety of libraries nicely. The issue with C is that it is insufficiently secure.

This study has produced a Web application that focuses on several objectives. First, to improve the user experience, increase the amount of reliable data, improve patients' awareness of their pathology. Additionally, it links effectively to the Database and operates quickly and efficiently. Java is employed as the back-end programming language for Dashboard development in this thesis.

Chapter 4

Result and Discussion

IBD-Tool is a telemedicine monitoring instrument for IBD patients. To assess the status of the illness and identify its impact on quality of life, monitoring is conducted by the administration of questionnaires on a semi-regular basis. The platform allows medical experts to examine the evolution of a disease over time and detect stages of remission and recurrence. LINKS Foundation, the Department of Gastroenterology of the Mauritian Hospital of Turin, and the Polytechnic of Turin collaborated to build it, and it is currently primarily aimed at the Mauritian hospital of Turin patients. The User Interface (UI) and User Experience (UX) are crucial aspects of IBD tools. In sections 1.1, 1.2, and 1.3, various UI elements will be described, and in section 1.4, the IBD-tools assessment will illustrate and explain how satisfied the final users (doctors and patients) are with the IBD-tools' UI and UX.

4.1 IBD tools features

Although a physician must invite a patient to the application before allocating it, a physician may register for an application by filling out the web-app registration form. A physician may sign by clicking the registration box on the homepage and inputting their personal information, email address, and password. After the request has been completed and authorized, an email is delivered to the administrator account of the platform. If the request is valid, an email is sent to the email address provided by the doctor during the registration process to authenticate their account. To register a patient, a physician must input the patient's personal information into the application. A patient's email address gets a message including a randomly generated password for their account, which must be entered in combination with the email in order to access the program. The patient has the opportunity to change the password for their account at any time as stated in Fig. 4.1.



Figure 4.1: Registration page

Physicians and patients are the two user types supported by the program, each with their own permissions and features. The program splits its operations into patient and physician features, each with their own webpage. Using the IBD tool, physicians may update the health status of their patients, review the surveys they have received, make assessments, and contact patients as needed. Doctors also get access to the application's entirety, which includes the overall number of users, the number of users with a particular disease, and the number of users actively using the service. After signing in, the doctor may view his page, as Fig. 4.2 represents.

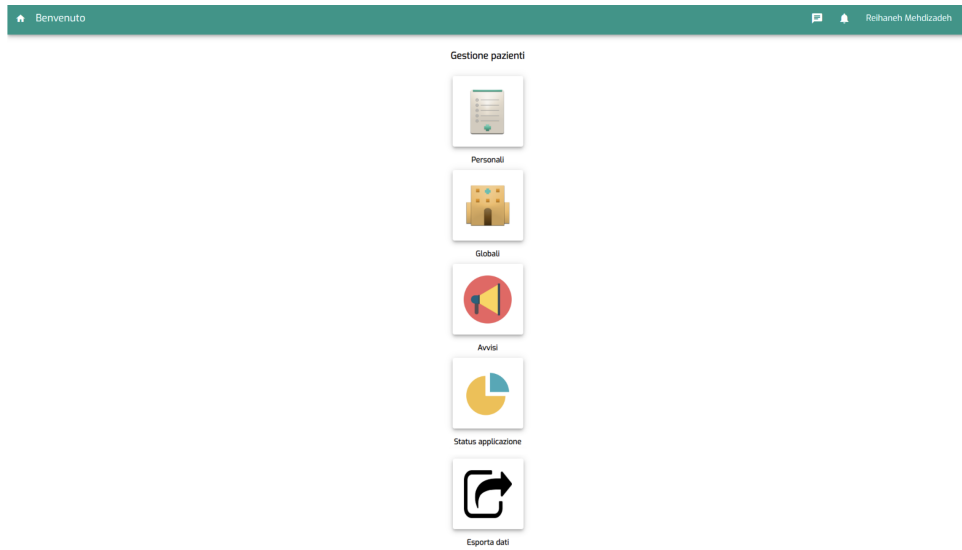


Figure 4.2: Physicians Home page

After signing in, the doctor may see his personal page, which displays sections pertaining to.

- Manage personal patients.
- Manage global patients of the application.
- Sent notice to all patients on the platform.
- Visualize the global data of the application.
- Export data of all patients to an excel file.

This area contains all of the functionality required for physicians to handle patients. In the application's list of all the logged-in doctor's patients, each patient's name, surname, SSN, date of birth, and type of assigned program (telemedicine or standard therapy) are shown in Fig. 4.3. To do any experimental research, a sample of patients and a reference sample should be considered. Usually, the reference has no access to the research product such as the web app). However, IBD Tool was launched during the pandemic for ethical reasons. The physicians chose to give access to the platform and the reference sample. IBD Tool was the easiest way to contact the physicians during the "lock-down" but with different timing of administration. Therefore, physicians divide patients into two categories telemedicine and standard care.

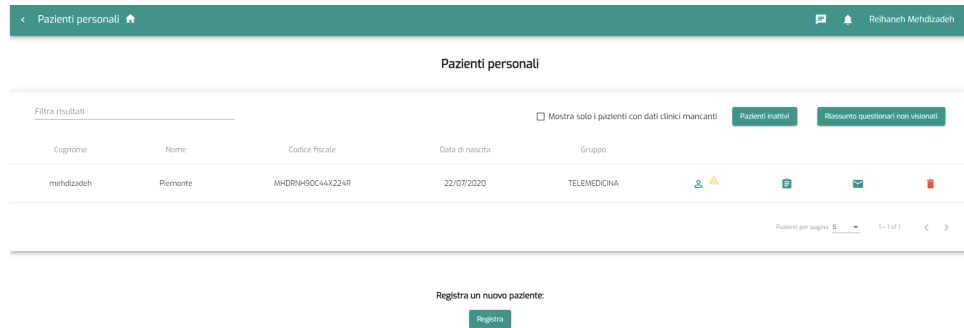


Figure 4.3: Personal patients section

This section displays a collection of clickable icons, enabling the physician to undertake various tasks for each patient's care. The doctor may access the patient's medical history by clicking on the icons. The capabilities provided make possible:

- Collect the patient's medical records, which include age, gender, weight, height, and date of illness diagnosis.
- Developing an assessment using the visualization of patient questionnaires: review the list of questions prepared by the patient.
- Email the patient to make contact.

In addition, selecting the Registra Icon enables the invitation of a new patient as demonstrates in Fig. 4.4. Fig. 4.5 depicts the part where all patient information may be seen. In part reserved for patient questions, the physician might submit numerous surgeries. Physicians may access all patient-completed surveys organized by kind. Fig. 4.6. depicts the window showing the summary of a patient's completed questionnaire, which includes the date, the answers provided, the questionnaire's

state, and any final assessment. Fig. 4.7 depicted the physician's ability to assess the patient's condition after receiving completed questionnaires from the patient. Fig. 4.8 represented the result of IBDQ questionnaires from the physician's side.

The screenshot shows a web application interface for registering a new patient. At the top, there is a green header bar with a back arrow, the text 'Registrazione paziente', a home icon, a chat icon, a bell icon, and the name 'Reihaneh Mehdizadeh'. The main content area is white and contains a central form titled 'Registra un nuovo paziente'. The form has several input fields: 'Inserisci l'email del paziente', 'Inserisci il nome del paziente', 'Inserisci il cognome del paziente', 'Inserisci il codice fiscale del pa...', 'Inserisci la data di nascita' (with a calendar icon), and 'Inserisci la patologia del paz...' (with a dropdown arrow). At the bottom of the form is a 'Registra' button.

Figure 4.4: Registration of new patient

The screenshot shows a web application interface for viewing a patient's personal data. At the top, there is a green header bar with a back arrow, the text 'Scheda paziente', a home icon, a chat icon, a bell icon, and the name 'Reihaneh Mehdizadeh'. The main content area is white and contains a central card titled 'Scheda paziente' with a subtitle 'Dati anagrafici'. The card lists the following information: 'Nome: Piemonte', 'Cognome: mehdizadeh', 'Codice fiscale: MHDRNH90C44X224R', 'Data di nascita: 22/07/2020' (with an edit icon), 'Luogo di nascita:', 'E-mail: reyhaneh.mehdizade@gmail.com', and 'Cellulare:' (with a phone icon).

Figure 4.5: Patient's personal data

A physician may monitor a patient's health status and see a list of all the patients

Paziente						
Tipo	Compilati	N. completati	Timer	Intervallo attuale	Invia nuovo	Rimuovi
EQ5D5L	Apri	0	Imposta timer	180	Invia	Rimuovi
HBI	Apri	0	Imposta timer	30	Invia	Rimuovi
IBD-DISK	Apri	0	Imposta timer	30	Invia	Rimuovi
IBDQ	Apri	1	Imposta timer	90	Invia	Rimuovi
IPAQ-SF	Apri	0	Imposta timer	90	Invia	Rimuovi

Questionari per pagina 5 1 - 5 of 15 |< < > >|

Figure 4.6: Questionnaire and periodicity management

Tipo	Compilati	N. completati	Timer	Intervallo attuale	Invia nuovo	Rimuovi
EQ5D5L	Apri	0	Imposta timer	180	Invia	Rimuovi
HBI	Apri	0	Imposta timer	30	Invia	Rimuovi
IBD-DISK	Apri	0	Imposta timer	30	Invia	Rimuovi
IBDQ	Apri ¹	3	Imposta timer	90	Invia	Rimuovi
IPAQ-SF	Apri	0	Imposta timer	90	Invia	Rimuovi

Figure 4.7: The received compiled IBDQ questionnaires

No.	Domanda	Valore
1	Quanto frequenti sono state le sue evacuazioni nelle ultime 2 settimane? Per favore indichi quanto frequenti sono state le sue evacuazioni nelle ultime 2 settimane, scegliendo una delle risposte seguenti:	2 - Estremamente frequenti
2	Quanto spesso nelle ultime 2 settimane la sensazione di affaticamento o di stanchezza e spossatezza è stata un problema per lei? Per favore indichi quanto frequentemente la sensazione di affaticamento o di stanchezza è stato un problema per lei nelle ultime 2 settimane, scegliendo una delle risposte seguenti:	1 - Tutto il tempo
3	Quanto spesso nelle ultime 2 settimane si è sentito frustrato, impaziente o agitato? Per favore, scelga una delle risposte seguenti:	6 - Una minima parte del tempo
4	Quanto spesso nelle ultime 2 settimane non è riuscito ad andare a scuola o a lavorare a	5 - Una piccola parte del

Valuta
Esci

Figure 4.8: The result of IBDQ questionnaires in physician side

registered on the platform and the doctors assigned to them. Except for eliminating patients from the platform, which is only accessible to personal patients, this page provides the same functionality as the page for managing personal patients. As stated in Fig. 4.9 allows physicians to visualize each patient's medical information, allowing them to collaborate and quickly assess each patient's health.

Pazienti registrati

Filtra risultati ☐ Mostra solo i pazienti con dati clinici mancanti

Cognome	Nome	Codice Fiscale	Curante	Gruppo			
Abello	Giulia	BLGL95D69F335R	Daperno Marco	TELEMEDICINA			
Acanfora	Mariano	CNFMK79C1F839D	Daperno Marco	TELEMEDICINA			
Accetulli	Alessio	CCTL5576D23D969W	Daperno Marco	STANDARD			
Acquadro	Loredana	CQDLN65R59A859C	Daperno Marco	TELEMEDICINA			
Acquaviva	Debora	COVDBRT6A67L219G	Daperno Marco	TELEMEDICINA			

Pazienti per pagina: 5 1 - 5 of 816 < >

Registra un nuovo paziente: [Registra](#)

Figure 4.9: Global patients section

Also, there is a section representing the notification sent to the users in the different sections. Fig. 4.10 shows the list of notifications in the IBD tools; if there is not any notification, the message shows there is not any message. Also, the notification is related to the chat; in this case, when the physicians or patients send different messages together, a notification is sent to both of them, representing that a new message is received.

Figure 4.10: Avvisi section

The status of the application in Fig. 4.11 gives physicians broad information about application users and how they utilize the service. On this page, it is possible to see graphs showing the number of patients in the platform broken down by disease type, disease state, and the number of platform patients that are active or inactive, as well as all numbers of global questionnaires.

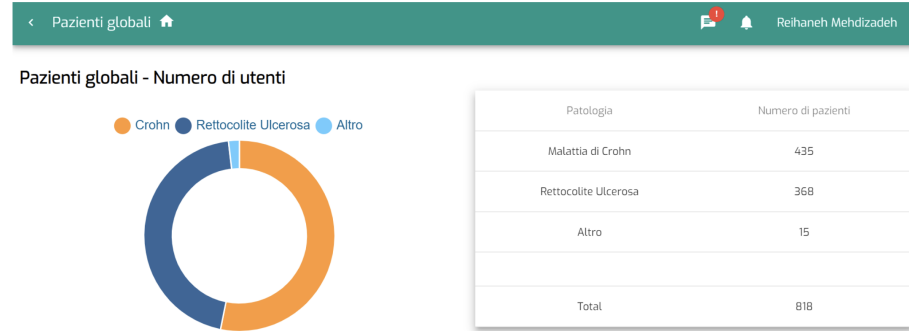


Figure 4.11: The number of total patients per type of disease, updated to the day:10/10/2022

And also, the number of active or inactive patients stated in Fig. 4.12, and all global questionnaires in Fig. 4.13.

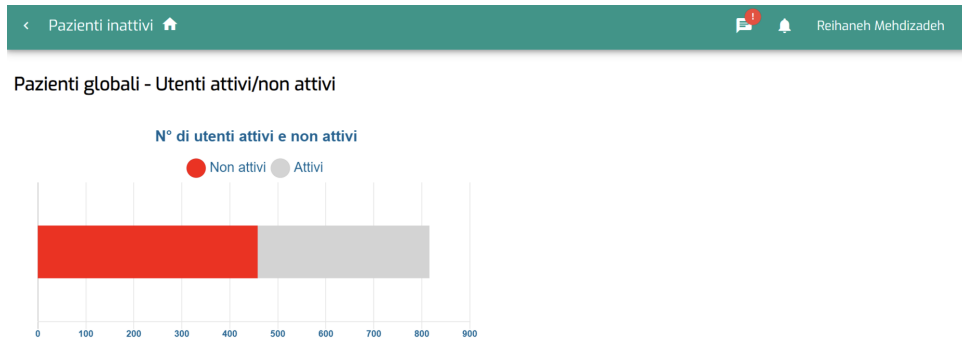


Figure 4.12: Number of total active and inactive patients, updated to the day:10/10/2022

Additionally, an overall summary of the disease's progression may be obtained among the enrolled patients. Fig. 4.14 and Fig. 4.15 depict two graphs that, when

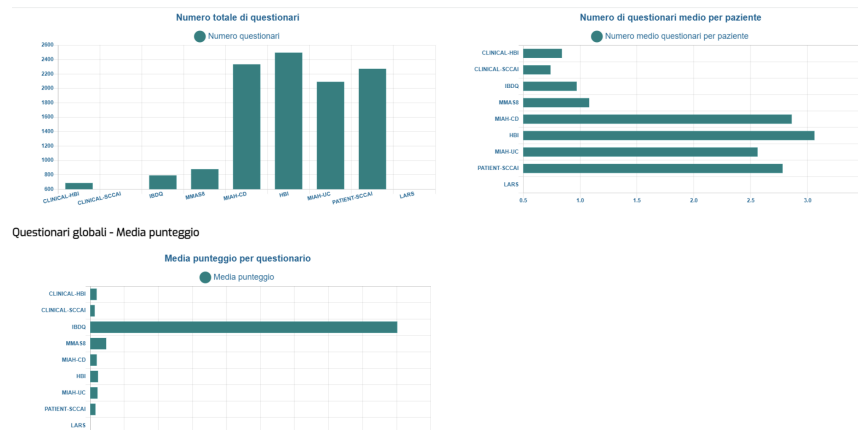


Figure 4.13: Number of total active and inactive patients, updated to the day:10/10/2022

separated by pathology and distinguished by the existence of illness or remission, illustrate the rate of stable, unstable, and improving patients.

Pazienti globali - Rettocolite ulcerosa: pazienti stabili e instabili

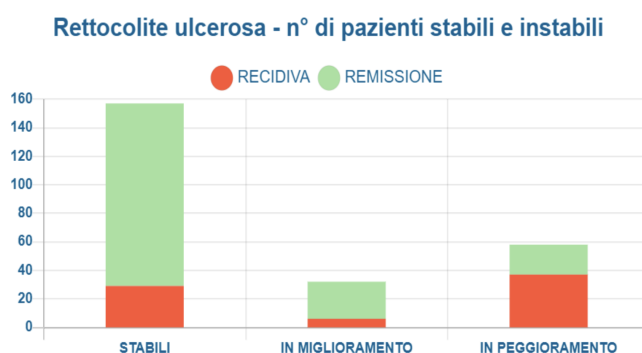


Figure 4.14: Stable-improving-worsening patient rate chart for UC patients, updated to the day:10/10/2022

Pazienti globali - Malattia di Crohn: pazienti stabili e instabili

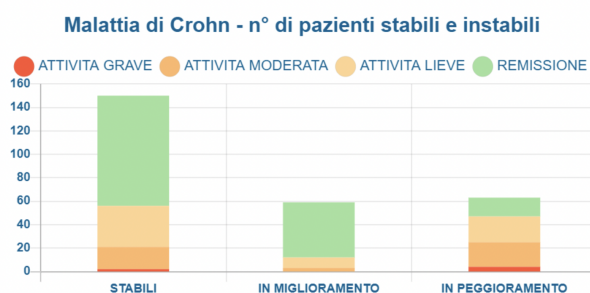
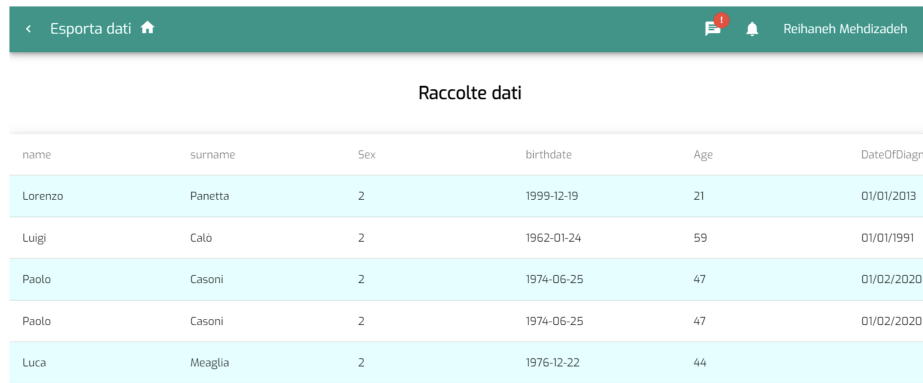


Figure 4.15: Stable-Improving-Worsening patient rate chart for CD patients, updated to the day:10/10/2022

Patients are placed in stable, unstable, or improving groups based on the assessments obtained in some questionnaires. This aspect will be analyzed in the

following paragraphs, listing and describing in detail the questionnaires present on the platform. The patient and the doctor fill out the main questionnaires in the same period. The result obtained from both questionnaires can be compared to evaluate the differences and be able to carry out concordance studies. It is possible to make this comparison in the Export data section in which, for each patient who has completed at least one questionnaire, the results obtained from the two questionnaires, the one filled in by the doctor and the one filled out by the patient, are compared and displayed globally in a table as showd in Fig. 4.16. It is possible to export the table in CSV format, which can be helpful not only for storing the data for the case of further research but also it can be used for analyzing the data and consider the different fields of study, such as data science and machine learning, for any other advanced study.



name	surname	Sex	birthdate	Age	DateOfDiagn
Lorenzo	Panetta	2	1999-12-19	21	01/01/2013
Luigi	Calo	2	1962-01-24	59	01/01/1991
Paolo	Casoni	2	1974-06-25	47	01/02/2020
Paolo	Casoni	2	1974-06-25	47	01/02/2020
Luca	Meaglia	2	1976-12-22	44	

Figure 4.16: Export data section

After logging in, the patient accesses his page Fig. 4.17, where the sections relating to,

- The questionnaires sent by the physician
- Satisfaction questionnaire
- Visualize compiled questionnaires
- Contact the physician
- Notices from physician
- Frequently asked questions

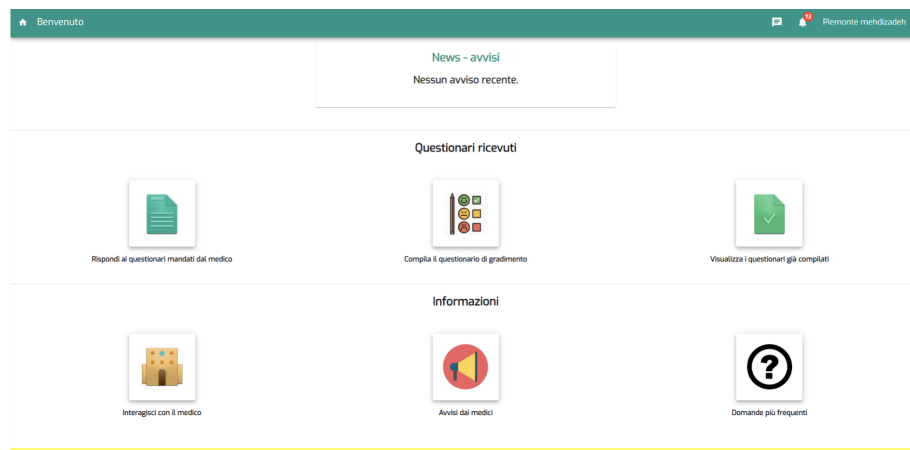


Figure 4.17: Home page of patients

The patient will receive an email notification each time a new questionnaire is available, along with a link that will take them straight to the compilation page. The Pending Questions area, Fig. 4.18, will still be available to the user, allowing them to view the list of all the pending questionnaires and, if necessary, complete them.

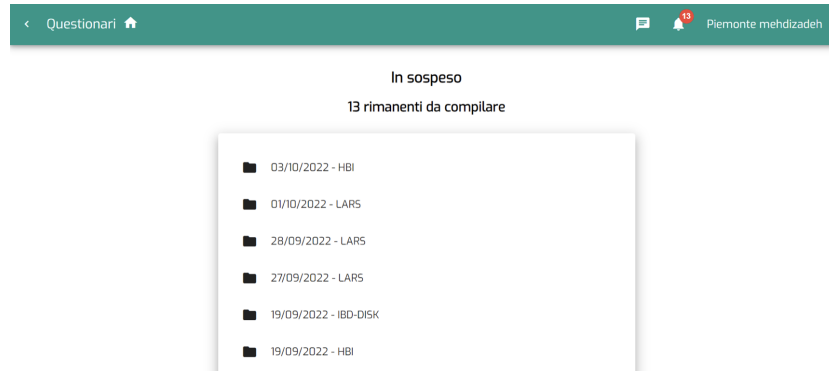


Figure 4.18: Questionnaires are due to compile

In the area titled Surveys already completed as stated in Fig. 4.19, the patient may access all previously completed surveys, organized by kind.

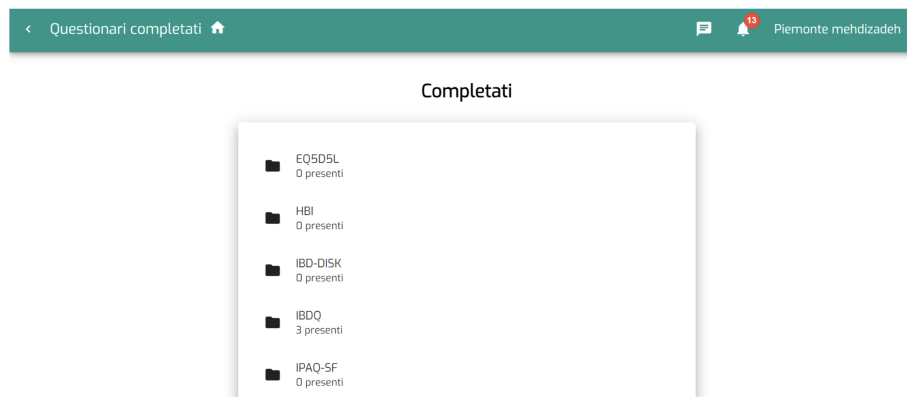


Figure 4.19: Compiled questionnaires section

For each questionnaire, the patient may examine the responses provided with the option to report any mistakes in Fig. 4.20, the date of completion, and the questionnaire status, viewed or not by the physician. If it has been reviewed and assessed by the physician, the final assessment and additional information, such as the occurrence of complications, will be accessible.

No.	Domanda	Valore
1	Quanto frequenti sono state le sue evacuazioni nelle ultime 2 settimane? Per favore indichi quanto frequenti sono state le sue evacuazioni nelle ultime 2 settimane, scegliendo una delle risposte seguenti:	2 - Estremamente frequenti
2	Quanto spesso nelle ultime 2 settimane la sensazione di affaticamento o di stanchezza e spossatezza è stata un problema per lei? Per favore indichi quanto frequentemente la sensazione di affaticamento o di stanchezza è stato un problema per lei nelle ultime 2 settimane, scegliendo una delle risposte seguenti:	1 - Tutto il tempo
3	Quanto spesso nelle ultime 2 settimane si è sentito frustrato, impaziente o agitato? Per favore sceglia una delle risposte seguenti:	6 - Una minima parte del tempo
4	Quanto spesso nelle ultime 2 settimane non è riuscito ad andare a scuola o a lavorare a	5 - Una piccola parte del

Segnala

Esci

Figure 4.20: Completed questionnaire summary with status and responses

In the Interact with the physicians section, seen in Fig. 4.21, the user may communicate with their doctor through a chat. It is also possible to attach photographs and other things to make communication as efficient as possible. In case of a necessity or change in health, it is possible to send surveys independently by selecting from a list of accessible questions, as stated in Fig. 4.22.



Figure 4.21: Interact with the doctor sections

Result and Discussion

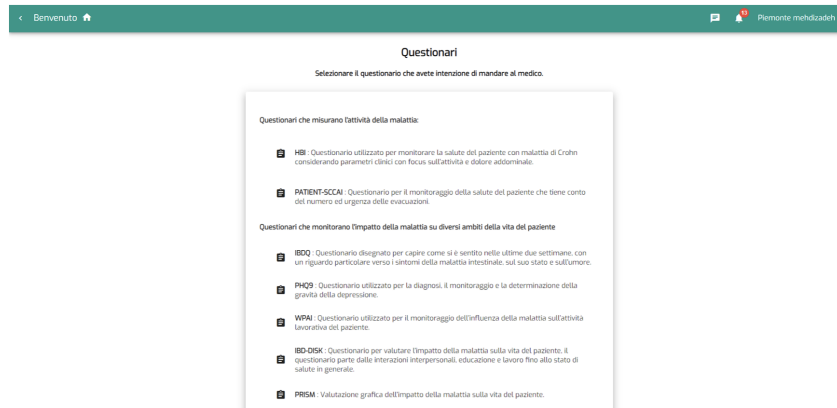


Figure 4.22: List of possible questionnaires

Patients and physicians may converse through a platform-specific chat. Through this conversation, a patient may send direct messages to all of the platform's physicians, seeking treatment comments, expressing concerns about utilizing the site or requesting medical visits. Sharing their perspectives on the chat messages allows for more incredible medical teamwork and a quicker response to each patient's request. The chat also supports communication files, making it easier to exchange photographs or pdf files related to medical treatments, as seen in Fig. 4.23.

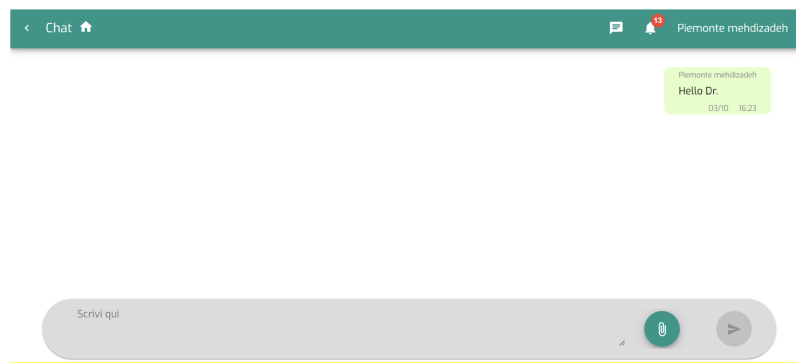


Figure 4.23: Example of the chat page

Fig. 4.23 shows an "Apri" button, which refers to by opening. By selecting this option, physicians may see the results of patient questionnaires for further evaluation. Fig. 4.24 depicts the contents table that physicians may interpret. In the first column are the titles of the questionnaires that the patients complete. The third one is the number of received questionnaires, which shows how many questionnaires are sent to the doctors from the doctors. Next is Timer, which regulates how often physicians may submit periodic questionnaires. Next, represents the interval of the questionnaires. The button before the last one is an offer for physicians to send further questionnaires to patients. In the last column is the remove, which is evaluated while eliminating the inquiries from the list. Fig. 4.25 depicts the page where physicians may handle received questionnaires. In this case, they permit selecting the desired link from a list of clickable links. Users can access the questionnaires and see the results by clicking on each link. Fig. 4.26 displays the responses obtained from each patient, and medical experts may choose to examine these responses in preparation for further inquiry. When treating a patient, a physician may place more emphasis on some risk concerns than others. These risk factors include a concomitant ailment, the failure of the transplant procedure itself, infection, or rejection. In point of fact, there are a number of objective questions whose answers may be used to assist in making a diagnosis and planning a course of therapy for a patient: What sort of transplant procedure is being carried out? The last aspect of the IBD tool is the validation of the questionnaires; there are four choices that are offered, each of which enables medical professionals to provide a response to the questionnaires.

- "Ok": that questionnaire was accepted by the doctors.
- "Segnala": It reports the received questionnaires.
- "Conferma": it represents that doctors accped the questionnaire.
- "Annulla": which is equivalent to canceling the options.

Fig. 4.27 shows the management questionnaires page.

Paziente						
Tipo	Compilati	N. completati	Timer	Intervallo attuale	Invia nuovo	Rimuovi
LARS	Apri	1	Imposta timer	90	Invia	Rimuovi
MIAH-CD	Apri	0	Imposta timer	30	Invia	Rimuovi
MIAH-UC	Apri	0	Imposta timer	-	Invia	Rimuovi
MMAS8	Apri	0	Imposta timer	90	Invia	Rimuovi
PATIENT-SCCAI	Apri	0	Imposta timer	-	Invia	Rimuovi

Figure 4.24: Content of the questionnaires page for the doctors

Questionari LARS di Piemonte mehdizadeh

Mostra grafico

Filtra risultati

Seleziona

📁 07/10/2022

Figure 4.25: Received questionnaires page

07/10/2022 - Letto

No.	Domanda	Valore
1	Le è mai capitato di non riuscire a controllare la flatulenza? (Emissione di Aria)	3 - Sì, almeno una volta a settimana
2	Le è mai capitato di avere una perdita involontaria di feci liquide?	2 - Sì, meno di una volta a settimana
3	Quante volte va di corpo?	2 - 4-7 volte al giorno (su 24 ore)
4	Le è mai capitato di dover andare in corpo dopo un'ora dall'ultima scarica di feci?	3 - Sì, almeno una volta a settimana
5	Le capita mai di aver impellente urgenza di evacuare tanto da costringerla a correre alla toilette?	2 - Sì, meno di una volta a settimana

Valuta Esci

Figure 4.26: Questionarie's result

No.	Domanda	Valore
1	Le è mai capitato di non riuscire a controllare la flatulenza? (Emissione di Aria)	3 - Sì, almeno una volta a settimana
2	Le è mai capitato di avere una perdita di sangue?	2 - Sì, meno di una volta a settimana
3	Quante volte va di corpo?	2 - 4-7 volte al giorno (su 24 ore)
4	Le è mai capitato di dover andare in bagno di notte?	3 - Sì, almeno una volta a settimana
5	Le capita mai di aver impellente urgente?	2 - Sì, meno di una volta a settimana

Figure 4.27: Questionnaire's management

4.2 New updated on the IBD tools

This section will be explained the types of features updated in this research. The previous version of the IBD tool contains different errors, which are addressed in this thesis. The IBD tool developed in this research is based on the Italian language since the physicians requested that, and it is based on Italian research.. One of the IBD tools aims is to provide the telemedicine feature to allow physicians to examine patients by considering different questionnaires without any in-person needed. This section explained that adding the new questionnaires called LARS is another way to improve the current IBD tools.

Considering the new methods, some of the important issues in the previous IBD tools are faded. Some modifications include,

- Adding a new icon helps doctors publish news to different groups of patients and other icons with the same functionalities.
- Some bugs are reported by patients and physicians related to the bugs of the system, which are faced in the new version of the IBD tools.
- In the previous version of the IBD tools, it was impossible to use the tool in the different browsers, which is solved in the current version.

- In the previous version of the IBD tools, it was impossible to use the tool in the different browsers, which is solved in the current version.

In these years, clinical treatment has grown more attuned to the requirements of patients and the significance of involving patients more actively, and patients themselves have increased their awareness of the disease that ails them. The news section may be of assistance in this regard and can also help in acquiring a new tool for more efficient communication.

4.3 Dashboard Evaluation

Consideration must first and foremost be given to gauging the level of contentment experienced by the final User. In a survey of IBD tools that were developed using a Google form, fifty researchers provided their responses to the questions. The user interface design is the most critical factor to consider while thinking about the end user, as seen in Fig. 4.28 and Fig. 4.29.

In addition, they report that they did not have any problems with accessibility, which indicates that the buttons' design and the panel's general structure have been established in detail. Designers often use color palettes as a tool to help them create new colors by mixing two or more existing hues. In addition, palettes allow one to modify the saturation, value (also known as strength), and other color schemes of the color. These aspects of the colors have an effect on the overall visual presentation of the artwork because they pique people's interest and improve the user experience (UX). The second question enables other researchers and developers who wish to enhance the present system to evaluate the three various possibilities in order to extend the IBD tool, including video calls, voice calls, and other features. This may be done by considering the IBD tool expansion options. The majority of the researchers who participated in the survey and provided responses are of the opinion that voice calls might be an option that could be added to the system in order to enhance the features. Since of this, functions like phone calls would be an improvement to the system because they would assist patients in becoming more educated about their condition. Another feature that the patients questioned would like to see included are video calls. This is because video calls make it possible for patients to communicate with their doctors through a third party, which is helpful in the event that people have a few questions for their physician.

Fig. 4.30 represents the result of the second question. Among the three options of chat, voice, and video call, most users prefer to use the chat to connect with their doctors, according to Fig. 4.31. However, more than half of the users are also interested in voice calls. Moreover, in the end, half of the patients are curious about using video calls as communication to communicate with the doctors.

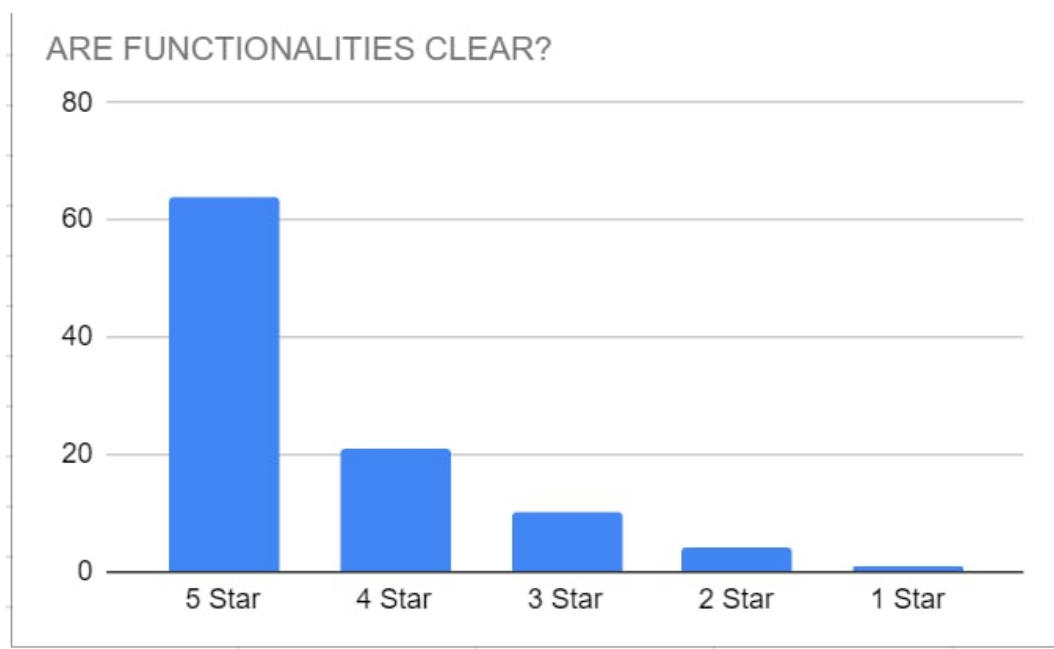


Figure 4.28: First question

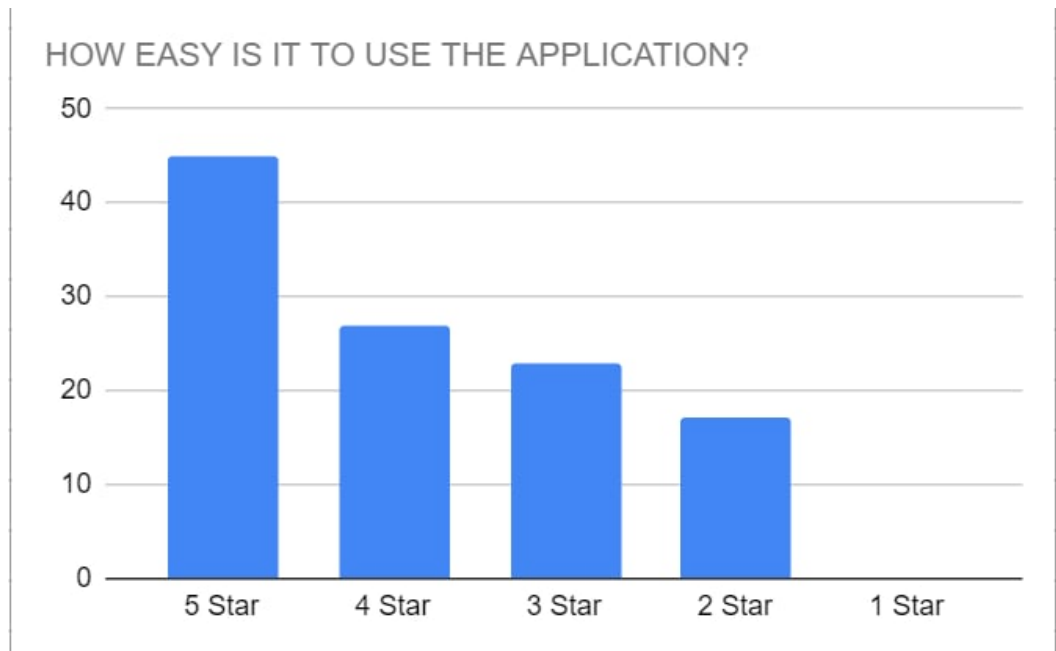


Figure 4.29: Second question

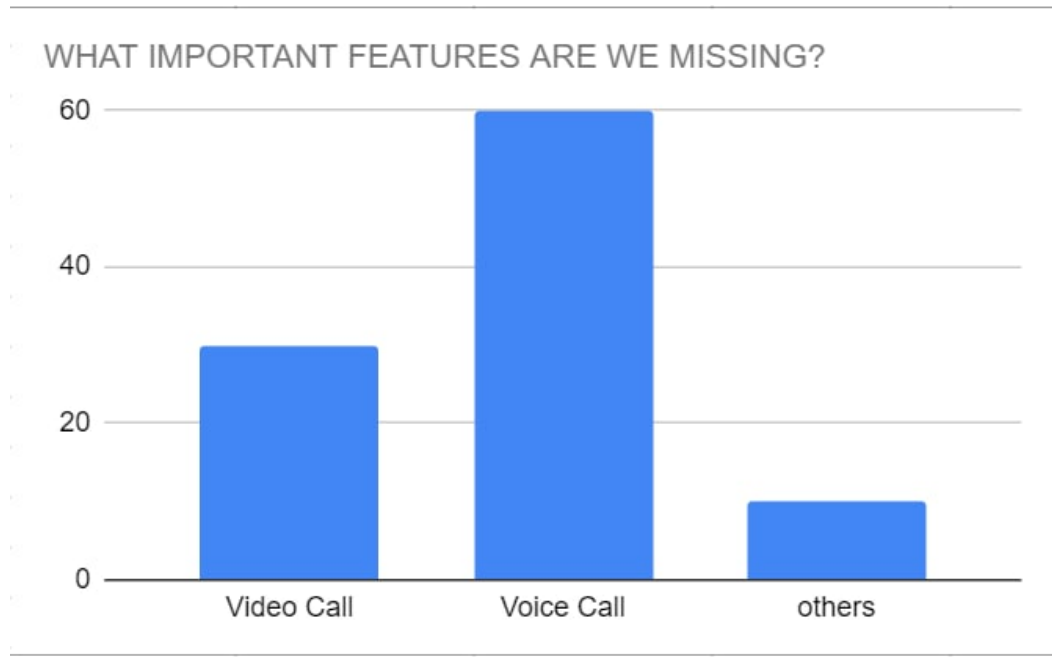


Figure 4.30: Third question

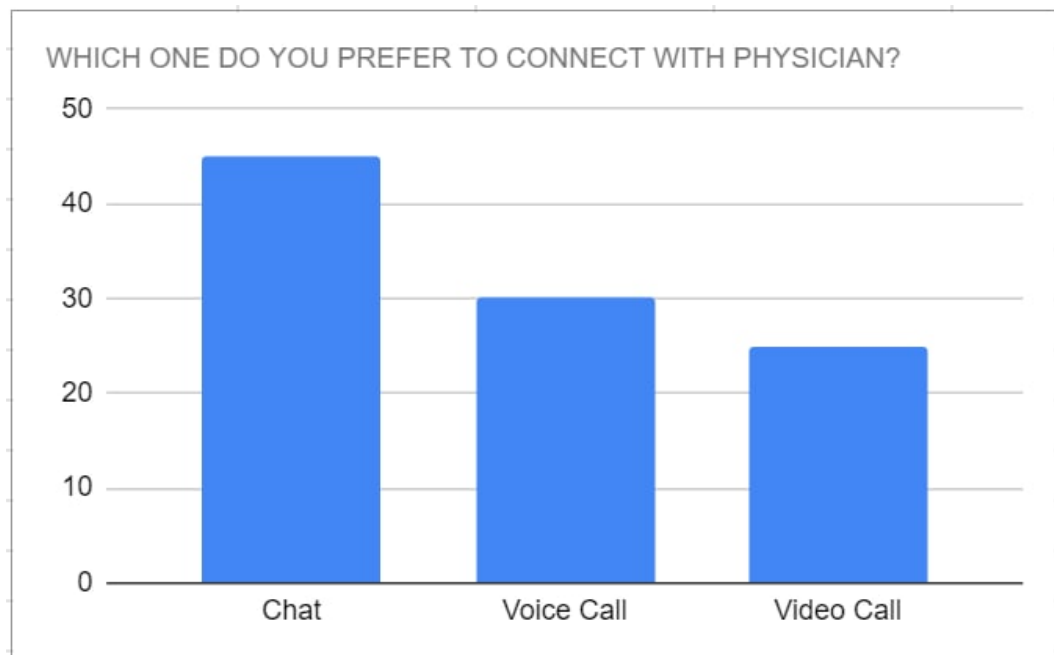


Figure 4.31: Fourth question

Chapter 5

Conclusion and future work

IBD tools can be used to improve telemedicine, a communication technology that helps patients and physicians interact remotely. In this case, patients can contribute to filling out the questionnaires, and physicians can check up on them remotely by considering the results without any in-person activity.

The proposed IBD tools in this research were inspired by two primary questions listed below.

- How current IBD tools can be improved to able all the users to utilize that?
- How do IBD tools satisfy users in the case of the UI and UX?

In the first step, one of the challenges of each system, which needs interaction with final users, is the website. This study developed some improvements to the current IBD to overcome issues like opening the IBD tools in different browsers, adding some icons to the first page to increase the functionalities, and adding more questionnaires to the currently available one. UIs designed in this research can work smoothly for patients and physicians since the programming language and Database are fully suitable for use in different types of environments including hospitals. Physicians and patients can easily interact with each other by using the chat box developed in the IBD tools. The role of the IBD Tool as a data collector has been further implemented in this thesis work; in fact, a new tool for retrieving data of interest for research purposes has been provided. Finally, the last point covered by the thesis concerns the need to make the patient increasingly aware and involved through a system of alerts in a new web app section.

As regards the improvement work that can be carried out in the future, the system for sending information through the news section can be started in the first place. In addition, further improvement also can be considered to enhance the current features of IBD tools, including adding video or voice calls between

patients and physicians. Moreover, more details can be used to improve the current system and make the examination more powerful and accessible for all physicians.

Bibliography

- [1] Hamzehei, Sahand. "Gateways and Wearable Tools for Monitoring Patient Movements in a Hospital Environment - Webthesis." webthesis.biblio.polito.it. Accessed August 14, 2022. <http://webthesis.biblio.polito.it/id/eprint/22711>.
- [2] Omid Akbarzadeh, Mehrshid Baradaran, Mohammad R. Khosravi, "IoT-Based Smart Management of Healthcare Services in Hospital Buildings during COVID-19 and Future Pandemics", Wireless Communications and Mobile Computing, vol. 2021, Article ID 5533161, 14 pages, 2021. <https://doi.org/10.1155/2021/5533161>
- [3] M. M. Alam and E. B. Hamida, "Surveying wearable human assistive technology for life and safety critical applications: Standards challenges and opportunities", Sensors, vol. 14, pp. 9153-9209, Mar. 2014.
- [4] D. M. West, "How 5G technology enables the health Internet of Things", pp. 1-20, 2016, [online] Available: <https://www.brookings.edu/research/how-5g-technology-enables-the-health-internet-of-things/>.
- [5] P. Doupi et al., Oct. 2010., [online] Available: [http://www.ehealth-strategies.eu/database/documents/Estonia\\$_CountryBrief\\$_eHStrategies.pdf](http://www.ehealth-strategies.eu/database/documents/Estonia$_CountryBrief$_eHStrategies.pdf)
- [6] N. P. Pai, C. Vadnais, C. Denkinger, N. Engel and M. Pai, "Point-of-care testing for infectious diseases: Diversity complexity and barriers in low- and middle-income countries", PLoS Med., vol. 9, no. 9, pp. e1001306, 2012.
- [7] Pratt, M. K. (2019, July 1). What is ICT (Information and Communications Technology)? Search-CIO. <https://www.techtarget.com/searchcio/definition/ICT-information-and-communications-technology-or-technologies>
- [8] 2009. TELEMEDICINE Opportunities and developments in member states. [ebook] WHO, p.6. Available at: https://apps.who.int/iris/bitstream/handle/10665/44497/9789241564144_eng.pdf;jsessionid=D2EC288311A8259B4BC8CEE1F2BC2045?sequence=1 [Accessed 1 September 2022]
- [9] A. Jamal, S. Hussain, A. Zafar and A. Z. Malik, "Role of Telemedicine during disaster: A Case Study," 2007 9th International Conference on e-Health Networking, Application and Services, 2007, pp. 261-263, doi:

- 10.1109/HEALTH.2007.381645.
- [10] P. de Toledo, S. Jimenez, F. del Pozo, J. Roca, A. Alonso and C. Hernandez, "Telemedicine Experience for Chronic Care in COPD," in *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 3, pp. 567-573, July 2006, doi: 10.1109/TITB.2005.863877.
 - [11] L. Kapoor, R. Basnet, R. D. Chand, S. Singh and S. K. Mishra, "An Audit of Problems in Implementation of Telemedicine Programme," 2007 9th International Conference on e-Health Networking, Application and Services, 2007, pp. 87-89, doi: 10.1109/HEALTH.2007.381610.
 - [12] L. Su, "Application of Telemedicine Diagnosis Assistant System for Breast Diseases Patients," 2020 13th International Conference on Intelligent Computation Technology and Automation (ICICTA), 2020, pp. 453-456, doi: 10.1109/ICICTA51737.2020.00102.
 - [13] L. van Dyk, M. Groenewald and J. F. Abrahams, "Towards a Regional Innovation System for Telemedicine in South Africa," 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine, 2010, pp. 1-4, doi: 10.1109/eTELEMED.2010.8.
 - [14] 2022. TELEMEDICINA Linee di indirizzo nazionali. [ebook] A cura di Ministero della Salute, p.18. Available at: https://www.salute.gov.it/imgs/C_17_pubblicazioni_2129_allegato.pdf [Accessed 10 September 2022].
 - [15] "ANSA: Boom della telemedicina durante la pandemia", https://www.ansa.it/canale_saluteebenessere/notizie/medicina/2021/06/23/e-boom-della-telemedicina-20-durante-la-pandemia_2f179bcc-540f-42fb-838d-9916e36371fa.html
 - [16] Inflammatory Bowel Disease (IBD): Symptoms, Diagnosis and Treatment. (n.d.). Inflammatory Bowel Disease (IBD): Symptoms, Diagnosis and Treatment. Retrieved August 20, 2022, from <https://www.nationwidechildrens.org/conditions/inflammatory-bowel-disease>
 - [17] Overview of Crohn's Disease | Crohn's / Colitis Foundation. (n.d.). Crohn's Colitis Foundation. Retrieved September 20, 2022, from <https://www.crohnscolitisfoundation.org/what-is-crohns-disease/overview>
 - [18] A. Oto et al., "Evaluation of diffusion-weighted MR imaging for detection of bowel inflammation in patients with Crohn's disease", *Academic radiology*, vol. 16, pp. 597-603, 2009.
 - [19] Crohn's disease. (n.d.). Nhs.Uk. Retrieved September 2022, from <https://www.nhs.uk/conditions/crohns-disease/>
 - [20] W. Schultz et al., "Analysis of integrated inflammatory bowel disease mouse models to assess their disease driving pathways and relevance for Crohn's disease and Ulcerative colitis," 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2018, pp. 2806-2807, doi: 10.1109/BIBM.2018.8621158.
 - [21] V. Taimouri, S. Kurugol, S. Clancy, M. Freiman and S. K. Warfield, "Structural

- and diffusion weighted MRI registration for biomarker fusion in Crohn's disease diagnosis," 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI), 2015, pp. 592-595, doi: 10.1109/ISBI.2015.7163943.
- [22] C. C. Dias, F. Magro and P. P. Rodrigues, "Disabling and Reoperation in Patients with Crohn's Disease Subject to Early Surgery or Immunosuppression: A Bayesian Network Prognostic Model," 2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS), 2016, pp. 265-268, doi: 10.1109/CBMS.2016.65.
- [23] Z. U. Hussain et al., "A Comparison of Machine Learning Approaches for Predicting the Progression of Crohn's Disease," 2020 IEEE Student Conference on Research and Development (SCoReD), 2020, pp. 529-533, doi: 10.1109/SCoReD50371.2020.9251019.
- [24] H. Z. Girgis, B. R. Mitchell, T. Dassopoulos, G. Mullin and G. Hager, "An intelligent system to detect Crohn's disease inflammation in Wireless Capsule Endoscopy videos," 2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2010, pp. 1373-1376, doi: 10.1109/ISBI.2010.5490253.
- [25] A. H. Sharma et al., "Deep Learning for Predicting Pediatric Crohn's Disease Using Histopathological Imaging," 2022 Systems and Information Engineering Design Symposium (SIEDS), 2022, pp. 122-127, doi: 10.1109/SIEDS55548.2022.9799299.
- [26] M. Shand et al., "Identifying Pediatric Crohn's Disease Using Deep Learning to Classify Magnetic Resonance Enterography (MRE) Images," 2021 Systems and Information Engineering Design Symposium (SIEDS), 2021, pp. 1-6, doi: 10.1109/SIEDS52267.2021.9483720.
- [27] Llamas, M. (2022, July 28). Crohn's Disease | Causes, Symptoms, Diagnosis, Treatment. Drugwatch.Com. <https://www.drugwatch.com/health/digestive-health/crohns-disease/>
- [28] Sheehan AL, Warren BF, Gear MW, Shepherd NA. Fat-wrapping in Crohn's disease: pathological basis and relevance to surgical practice. *Br J Surg*. 1992 Sep;79(9):955-8. doi: 10.1002/bjs.1800790934. PMID: 1422768.
- [29] Ulcerative colitis. (n.d.). Nhs.Uk. Retrieved September 2022, from <https://www.nhs.uk/conditions/ulcerative-colitis/>
- [30] What is Ulcerative Colitis? | Crohn's Colitis Foundation. (n.d.). Crohn's Colitis Foundation. Retrieved September 23, 2022, from <https://www.crohnscolitisfoundation.org/what-is-ulcerative-colitis>
- [31] Ulcerative colitis - Symptoms and causes. (2022, September 16). Mayo Clinic. <https://www.mayoclinic.org/diseases-conditions/ulcerative-colitis/symptoms-causes/syc-20353326>
- [32] T. Kadota et al., "Automatic Estimation of Ulcerative Colitis Severity by Learning to Rank With Calibration," in *IEEE Access*, vol. 10, pp. 25688-25695, 2022, doi: 10.1109/ACCESS.2022.3155769.

- [33] H. Nosato, H. Sakanashi, E. Takahashi and M. Murakawa, "An objective evaluation method of ulcerative colitis with optical colonoscopy images based on higher order local auto-correlation features," 2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI), 2014, pp. 89-92, doi: 10.1109/ISBI.2014.6867816.
- [34] Sutton, R.T., Zaiane, O.R., Goebel, R. et al. Artificial intelligence enabled automated diagnosis and grading of ulcerative colitis endoscopy images. *Sci Rep* 12, 2748 (2022). <https://doi.org/10.1038/s41598-022-06726-2>
- [35] staff. (2021, January 1). Ulcerative colitis. AboutKidsHealth. Retrieved September 1, 2022, from <https://www.aboutkidshealth.ca/article?contentid=924&language=english>
- [36] Inflammatory bowel disease (IBD) - Diagnosis and treatment - Mayo Clinic. (2022, September 3). Inflammatory Bowel Disease (IBD) - Diagnosis and Treatment - Mayo Clinic. Retrieved October 1, 2022, from <https://www.mayoclinic.org/diseases-conditions/inflammatory-bowel-disease/diagnosis-treatment/drc-20353320>
- [37] <https://www.crohnscolitisfoundation.org/sites/default/files/2019-02/Updated%20IBD%20Factbook.pdf>
- [38] Nguyen GC, Chong CA, Chong RY. National estimates of the burden of inflammatory bowel disease among racial and ethnic groups in the United States. *J Crohns Colitis*. 2014 Apr;8(4):288-95. doi: 10.1016/j.crohns.2013.09.001. Epub 2013 Sep 24. PMID: 24074875.
- [39] Johan Burisch & Pia Munkholm (2015) The epidemiology of inflammatory
- [40] J. Tresca. (2022, April 28). How Telemedicine Can Work for IBD. Verywell Health. Retrieved October 1, 2022, from <https://www.verywellhealth.com/how-telemedicine-can-work-for-ibd-4843429>
- [41] Cross RK, Cheevers N, Rustgi A, Langenberg P, Finkelstein J. Randomized, controlled trial of home telemanagement in patients with ulcerative colitis (UC HAT). *Inflamm Bowel Dis*. 2012;18(6):1018-1025. doi:10.1002/ibd.21795
- [42] Del Hoyo J, Nos P, Bastida G, et al. Telemonitoring of crohn's disease and ulcerative colitis (TECCU): cost-effectiveness analysis. *J Med Internet Res*. 2019;21:e15505. doi:10.2196/15505
- [43] Elkjaer M, Burisch J, Avnstrøm S, Lynge E, Munkholm P. Development of a Web-based concept for patients with ulcerative colitis and 5-aminosalicylic acid treatment. *Eur J Gastroenterol Hepatol*. 2010;22:695-704. doi:10.1097/MEG.0b013e32832e0a18
- [44] Wales. (2020, December 8). Front-End vs Back-End vs Full Stack Web Developers | Udacity. Udacity. Retrieved October 9, 2022, from [https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-webdevelopers.html#:~:text=The%20front%2Dend%20of%20a,controlled%20by%20your%](https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-webdevelopers.html#:~:text=The%20front%2Dend%20of%20a,controlled%20by%20your%20)

- 20computer&~s%20browser.
- [45] <https://airfocus.com/glossary/what-is-a-back-end/>. (n.d.). What Is a Back End (in a Website)? Definition FAQs. Retrieved October 9, 2022, from <https://airfocus.com/glossary/what-is-a-back-end/>
 - [46] A. (2018, January 30). Everything You Need to Know About Web Databases - Zenkit. Zenkit. Retrieved October 9, 2022, from <https://zenkit.com/en/blog/everything-you-need-to-know-about-web-databases/#:~:text=a%20web%20database%3F-,Web%20Database%20Definition,desktop%20or%20its%20attached%20storage.>
 - [47] Frontend vs Backend - GeeksforGeeks. (2019, July 11). GeeksforGeeks. Retrieved October 12, 2022, from <https://www.geeksforgeeks.org/frontend-vs-backend/:.text=HTML>
 - [48] S., A. (2018, November 15). What Is HTML? Hypertext Markup Language Basics for Beginners. Hostinger Tutorials. Retrieved October 12, 2022, from <https://www.hostinger.com/tutorials/what-is-html>
 - [49] B., A. (2019, January 13). What is CSS: Cascading Style Sheet Explained for Beginners. Hostinger Tutorials. Retrieved October 12, 2022, from <https://www.hostinger.com/tutorials/what-is-css>
 - [50] What is JavaScript used for? | Hack Reactor. (n.d.). What Is JavaScript Used For? Retrieved October 12, 2022, from <https://www.hackreactor.com/blog/what-is-javascript-used-for>
 - [51] AngularJS. (n.d.). AngularJS. Retrieved October 12, 2022, from <https://docs.angularjs.org/guide/introduction>
 - [52] The Best Guide to Know What Is React [Updated]. (n.d.). Simplilearn.com. Retrieved October 12, 2022, from <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs>
 - [53] A., J. (2018, December 29). What is jQuery: Beginner's Guide to jQuery Library. Hostinger Tutorials. Retrieved October 12, 2022, from <https://www.hostinger.com/tutorials/what-is-jquery/>
 - [54] C. (2020, January 10). What is Sass? Your guide to this top CSS preprocessor | Creative Bloq. What Is Sass? Your Guide to This Top CSS Preprocessor. Retrieved October 12, 2022, from <https://www.creativebloq.com/web-design/what-is-sass-111517618>
 - [55] What is Flutter and Why You Should Learn it in 2020. (2019, December 12). freeCodeCamp.org. Retrieved October 12, 2022, from <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/>
 - [56] Most used languages among software developers globally 2022 | Statista. (n.d.). Statista. Retrieved October 12, 2022, from <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/>

- [57] (2021, April 27). Top 7 Programming Languages for Backend Web Development - GeeksforGeeks. GeeksforGeeks. Retrieved October 14, 2022, from <https://www.geeksforgeeks.org/top-7-programming-languages-for-backend-web-development/>

Appendix A

Appendix

Questionnaires components for patients developed with Angular programming languages.

```
1
2 import {Component, OnInit, Renderer2} from '@angular/core';
3 import {FormControl, FormGroup, Validators} from '@angular/forms';
4 import {QuestionnaireService} from '../../../services/
   questionnaire.service';
5 import {MatDialog} from '@angular/material/dialog';
6 import {ActivatedRoute, Router} from '@angular/router';
7 import {ErrorDialogComponent} from '../../../user-actions/error-
   dialog/error-dialog.component';
8 import {ClinicalHBIResults, Q_CLINICAL_HBI} from '../../../model/
   questionnaires/questionnaire-clinical-hbi';
9 import {MatOptionSelectionChange} from '@angular/material/core';
10 import {HbiDialogComponent} from '../questionnaire-hbi/questionnaire-
   hbi.component';
11
12 @Component({
13   selector: 'app-questionnaire-clinical-hbi',
14   templateUrl: './questionnaire-clinical-hbi.component.html',
15   styleUrls: ['./questionnaire-clinical-hbi.component.css']
16 })
17 export class QuestionnaireClinicalHbiComponent implements OnInit {
18
19   qForm: FormGroup;
20   dataSource = Q_CLINICAL_HBI;
21   notFocused = false;
22   submitted = false;
23   show = false;
24   displayedColumns = ['position', 'question', 'value'];
25   values: string[] = [
```

```

26     'Artralgia ',
27     'Uveite ',
28     'Eritema nodoso ',
29     'Ulcera aftoide ',
30     'Pioderma gangrenoso ',
31     'Fissurazione anale ',
32     'Comparsa di una nuova fistola ',
33     'Ascesso '];
34 dialogResults = [ ' '];
35 dialogScore = 0;
36 results = [];
37 q = 'question ';
38 found = false;
39
40 /// toolbar
41 back = true;
42 destination: string;
43 title = 'CLINICAL-HBI';
44 user = null;
45 uuid = null;
46 authorization: boolean;
47 doctorID: string;
48 patientSSN: string;
49
50 constructor(private questionnaireService: QuestionnaireService,
51             public dialog: MatDialog,
52             private route: ActivatedRoute,
53             private router: Router) { }
54
55 ngOnInit(): void {
56     this.qForm = new FormGroup({
57         question3: new FormControl({ value: '' }, [ Validators.required,
58             Validators.min(0), Validators.max(25) ]),
59     });
60     // tslint:disable-next-line:prefer-for-of
61     for (let i = 0; i < this.dataSource.length; ++i) {
62         this.results.push(null);
63     }
64
65     this.user = JSON.parse(localStorage.getItem('user'));
66     const doctorID = JSON.parse(localStorage.getItem('doctorID'));
67     this.route.params.subscribe(params => {
68         this.doctorID = params.doctorID;
69         this.patientSSN = params.patientSSN;
70         this.uuid = params.uuid;
71         this.authorization = doctorID.localeCompare(params.doctorID)
72             === 0;
73         this.destination = '/patient-clinical-pendings/' + this.
74             doctorID + '/' + this.patientSSN + '/' + this.title.toLowerCase();

```

```

72     if (params.back.localeCompare('global') === 0) {
73         this.destination += '/global';
74     } else {
75         this.destination += '/local';
76     }
77     sessionStorage.setItem('destination', JSON.stringify(this.
destination));
78 });
79 }
80
81 selectResult($event: MatOptionSelectionChange, value: string,
position: number) {
82     if ($event.isUserInput) {
83         if (position !== 5) {
84             this.results[position - 1] = value.slice(0, 1);
85             console.log(this.results);
86         } else {
87             if (value === 'Si') {
88                 this.openHbiDialog($event, position, 'Presenza di
complicanze');
89             } else if (value === 'No') {
90                 this.results[position - 1] = '0';
91                 console.log(this.results);
92             }
93         }
94     }
95 }
96
97 setResult(position: number) {
98     const question = 'question' + position;
99     if (position === 3) {
100         this.results[position - 1] = this.qForm.get(question).value.
toString();
101     }
102 }
103
104 onSubmit(value) {
105     this.submitted = true;
106     this.show = true;
107     this.found = false;
108
109     if (this.results.length !== this.dataSource.length) {
110         this.show = false;
111         this.submitted = false;
112         this.found = true;
113         const msg = 'Non hai finito di completare il questionario!';
114         this.openErrorDialog(msg);
115     } else {
116         for (const r of this.results) {

```

```

117         if (r === null) {
118             this.show = false;
119             this.submitted = false;
120             this.found = true;
121             const msg = 'Non hai finito di completare il questionario
!';
122             this.openErrorDialog(msg);
123             break;
124         }
125     }
126
127     if (!this.found) {
128         let finalScore = 0;
129
130         for (let i = 0; i < this.results.length - 1; ++i) {
131             finalScore += parseInt(this.results[i], 10);
132         }
133
134         finalScore += this.dialogScore;
135         console.log(finalScore);
136         const resultToSend = new ClinicalHBIResults(this.doctorID,
this.patientSSN, this.results, this.dialogResults, finalScore);
137         this.questionnaireService.sendForm(resultToSend, 'clinical-
hbi', this.uuid);
138     }
139 }
140 }
141
142 openErrorDialog(message: string): void {
143     const dialogRef = this.dialog.open(ErrorDialogComponent, {
144         width: '500px',
145         height: '270px',
146         data: message
147     });
148
149     dialogRef.afterClosed().subscribe(result => {
150         console.log('Finestra chiusa');
151     });
152 }
153
154 openHbiDialog($event: MatOptionSelectionChange, position: number,
message: string): void {
155     const dialogRef = this.dialog.open(HbiDialogComponent, {
156         width: '500px',
157         height: '400px',
158         data: {message, values: this.values}
159     });
160
161     dialogRef.afterClosed().subscribe(result => {

```

```

162
163     if (result === undefined) {
164         this.openErrorDialog('Non hai effettuato alcuna selezione!');
165         $event.source.deselect();
166         this.results[position - 1] = null;
167     } else {
168         this.dialogResults = result;
169         this.dialogScore = this.dialogResults.length;
170         this.results[position - 1] = this.dialogScore.toString(10);
171     }
172     console.log('Finestra chiusa');
173     console.log(this.results);
174 }));
175 }
176
177 goHome() {
178     if (this.user === null) {
179         this.router.navigateByUrl('/startup');
180     } else {
181         this.router.navigateByUrl('/landing');
182     }
183 }
184 }
185
186 import {AfterViewInit, Component, OnInit} from '@angular/core';
187 import {PrismObject, PRISMResult} from '../../../../../model/
188     questionnaires/questionnaire-prism';
189 import {QuestionnaireService} from '../../../../../services/
190     questionnaire.service';
191 import {MatDialog} from '@angular/material/dialog';
192 import {ActivatedRoute, Router} from '@angular/router';
193 import {fabric} from 'fabric';
194
195 @Component({
196     selector: 'app-questionnaire-clinical-prism',
197     templateUrl: './questionnaire-clinical-prism.component.html',
198     styleUrls: ['./questionnaire-clinical-prism.component.css']
199 })
200 export class QuestionnaireClinicalPrismComponent implements OnInit,
201     AfterViewInit {
202
203     notFocused = false;
204     submitted = false;
205     show = false;
206     displayedColumns = ['position', 'question', 'value'];
207     results = [];
208     found = false;
209     finalScore = 0;

```

```

208  /// toolbar
209  back = true;
210  destination = '/questionnaires';
211  title = 'CLINICAL-PRISM';
212  user = null;
213  uuid = null;
214  authorization: boolean;
215  doctorID: string;
216  patientSSN: string;
217  canvas: any;
218  self: PrismObject;
219  disease: PrismObject;
220  distance: number;
221
222  constructor(private questionnaireService: QuestionnaireService,
223              public dialog: MatDialog,
224              private route: ActivatedRoute,
225              private router: Router) {}
226
227  ngOnInit(): void {
228      this.user = JSON.parse(localStorage.getItem('user'));
229      const doctorID = JSON.parse(localStorage.getItem('doctorID'));
230      this.route.params.subscribe(params => {
231          this.doctorID = params.doctorID;
232          this.patientSSN = params.patientSSN;
233          this.uuid = params.uuid;
234          this.authorization = doctorID.localeCompare(params.doctorID)
235          === 0;
236          this.destination = '/patient-clinical-pendings/' + this.
237          doctorID + '/' + this.patientSSN + '/' + this.title.toLowerCase();
238          if (params.back.localeCompare('global') === 0) {
239              this.destination += '/global';
240          } else {
241              this.destination += '/local';
242          }
243          sessionStorage.setItem('destination', JSON.stringify(this.
244          destination));
245      });
246
247      this.self = new PrismObject(500, 200);
248      this.disease = new PrismObject(40, 40);
249      // tslint:disable-next-line:max-line-length
250      this.distance = parseFloat((Math.hypot(this.self.x - this.disease
251      .x,
252      this.self.y - this.disease.y) * 0.026458).toFixed(1));
253
254      console.log('start distance: ', this.distance);
255  }

```

```

253 ngAfterViewInit(): void {
254     const canvas = new fabric.Canvas('canvas');
255     const disease = new fabric.Circle({
256         radius: 30, left: 40, top: 40, fill: '#cc0000',
257         originX: 'center', originY: 'center',
258         hasControls: false, hasBorders: false, hoverCursor: 'pointer',
259         moveCursor: 'pointer',
260         stroke: '#990000', strokeWidth: 2
261     });
262     const self = new fabric.Circle({
263         radius: 50, left: 500, top: 200, fill: '#ffe014', hasControls:
264         false, hasBorders: false,
265         originX: 'center', originY: 'center',
266         evented: false, stroke: '#ffc414', strokeWidth: 2
267     });
268     canvas.add(self, disease);
269     canvas.selection = false;
270     canvas.on({
271         'object:moving': e => {
272             const circle = canvas.getActiveObject();
273             const x1 = circle.left;
274             const y1 = circle.top;
275             const x2 = 500;
276             const y2 = 200;
277             const distance = parseFloat((Math.hypot(x2 - x1, y2 - y1) *
278             0.026458).toFixed(1));
279             console.log('Distance in cm: ', distance);
280             const selfObj = new PrismObject(x2, y2);
281             const diseaseObj = new PrismObject(x1, y1);
282             this.self = selfObj;
283             this.disease = diseaseObj;
284             this.distance = distance;
285             console.log('log', this.self, this.disease, this.distance);
286         }
287     });
288 }
289 goHome() {
290     if (this.user === null) {
291         this.router.navigateByUrl('/startup');
292     } else {
293         this.router.navigateByUrl('/landing');
294     }
295 }
296
297 onSubmit() {
298     this.submitted = true;

```

```

299     this.show = true;
300     this.found = false;
301     const resultToSend = new PRISMResult(this.doctorID, this.
patientSSN, this.self, this.disease, this.distance.toFixed(2));
302     this.questionnaireService.sendForm(resultToSend, 'clinical-prism
', this.uuid);
303   }
304 }
305
306 import {Component, OnInit, Renderer2} from '@angular/core';
307 import {FormGroup} from '@angular/forms';
308 import {QuestionnaireService} from '../services/
questionnaire.service';
309 import {MatDialog} from '@angular/material/dialog';
310 import {ActivatedRoute, Router} from '@angular/router';
311 import {ErrorDialogComponent} from '../user-actions/error-
dialog/error-dialog.component';
312 import {ClinicalSccaiResults, Q_CLINICAL_SCCAI} from '../model/questionnaires/questionnaire-clinical-sccai';
313
314 @Component({
315   selector: 'app-questionnaire-clinical-sccai',
316   templateUrl: './questionnaire-clinical-sccai.component.html',
317   styleUrls: ['./questionnaire-clinical-sccai.component.css']
318 })
319 export class QuestionnaireClinicalSCCAIComponent implements OnInit {
320
321   qForm: FormGroup;
322   dataSource = Q_CLINICAL_SCCAI;
323   notFocused = false;
324   submitted = false;
325   show = false;
326   displayedColumns = ['position', 'question', 'value'];
327   results = [];
328   q = 'question';
329   found = false;
330   finalScore = 0;
331
332   /// toolbar
333   back = true;
334   destination = '/landing';
335   title = 'CLINICAL-SCCAI';
336   user = null;
337   uuid = null;
338   authorization: boolean;
339   doctorID: string;
340   patientSSN: string;
341   currentCheckedValue = [null];
342

```

```

343     constructor(private questionnaireService: QuestionnaireService,
344                  public dialog: MatDialog,
345                  private ren: Renderer2,
346                  private route: ActivatedRoute,
347                  private router: Router) { }
348
349     ngOnInit(): void {
350         // tslint:disable-next-line:prefer-for-of
351         for (let i = 0; i < this.dataSource.length; ++i) {
352             this.results.push(null);
353         }
354
355         this.qForm = new FormGroup({});
356
357         this.user = JSON.parse(localStorage.getItem('user'));
358         const doctorID = JSON.parse(localStorage.getItem('doctorID'));
359         this.route.params.subscribe(params => {
360             this.doctorID = params.doctorID;
361             this.patientSSN = params.patientSSN;
362             this.uuid = params.uuid;
363             this.authorization = doctorID.localeCompare(params.doctorID)
364             === 0;
365             this.destination = '/patient-questionnaire-panel/' + this.
366             patientSSN;
367             if (params.back.localeCompare('global') === 0) {
368                 this.destination += '/global';
369             } else {
370                 this.destination += '/local';
371             }
372             sessionStorage.setItem('destination', JSON.stringify(this.
373             destination));
374         });
375     }
376
377     onSubmit(value: any) {
378         this.submitted = true;
379         this.show = true;
380         this.found = false;
381
382         if (this.results.length !== this.dataSource.length) {
383             this.show = false;
384             this.submitted = false;
385             this.found = true;
386             const msg = 'Non hai finito di completare il questionario!';
387             this.openErrorDialog(msg);
388         } else {
389             for (const r of this.results) {
390                 if (r === null) {
391                     this.show = false;

```

```

389         this.submitted = false;
390         this.found = true;
391         const msg = 'Non hai finito di completare il questionario
!';
392         this.openErrorDialog(msg);
393         break;
394     }
395 }
396
397 if (!this.found) {
398     console.log(this.finalScore);
399     const resultToSend = new ClinicalSccaiResults(this.doctorID,
this.patientSSN, this.results, this.finalScore);
400     this.questionnaireService.sendForm(resultToSend, 'clinical-
sccai', this.uuid);
401 }
402 }
403 }
404
405 setAnswer(answer: string, position: number, answers: string[]) {
406     console.log(answer, position, answers);
407     if (position >= 6) {
408         if (this.results[position - 1] !== null) {
409             console.log('Precedente: ', this.results[position - 1]);
410             console.log('Final Score Before: ', this.finalScore);
411             if (this.results[position - 1].localeCompare('Si') === 0) {
412                 this.finalScore -= 1;
413             }
414             console.log('Final Score After: ', this.finalScore);
415         }
416         if (answer.localeCompare('Si') === 0) {
417             this.finalScore += 1;
418         }
419     } else {
420         if (this.results[position - 1] !== null) {
421             let dec = 0;
422             answers.forEach(a => {
423                 if (a === this.results[position - 1]) {
424                     console.log('Final Score Before != 7: ', this.finalScore)
;
425                     this.finalScore -= dec;
426                     console.log('Final Score After: != 7', this.finalScore);
427                 }
428                 dec++;
429             });
430         }
431         let inc = 0;
432         answers.forEach(a => {
433             if (a === answer) {

```

```

434         this.finalScore += inc;
435     }
436     inc++;
437 });
438 }
439
440     this.results[position - 1] = answer;
441
442     console.log('Risposta: ', answer);
443     console.log('Risultati: ', this.results);
444     console.log('Final Score: ', this.finalScore);
445     console.log(this.results);
446 }
447
448 checkState(el, position: number, answers: string[]) {
449     setTimeout(() => {
450         if (this.currentCheckedValue[position - 1] && this.
451         currentCheckedValue[position - 1] === el.value) {
452             el.checked = false;
453             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
454             focused');
455             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
456             program-focused');
457             if (this.results[position - 1] !== null) {
458                 if (position >= 6) {
459                     if (this.results[position - 1] !== null) {
460                         console.log('Precedente: ', this.results[position - 1])
461                     };
462                     console.log('Final Score Before: ', this.finalScore);
463                     if (this.results[position - 1].localeCompare('Si') ===
464                     0) {
465                         this.finalScore -= 1;
466                     }
467                     console.log('Final Score After: ', this.finalScore);
468                 } else {
469                     let dec = 0;
470                     answers.forEach(a => {
471                         if (a === this.results[position - 1]) {
472                             console.log('Final Score Before != 7: ', this.
473                             finalScore);
474                             this.finalScore -= dec;
475                             console.log('Final Score After: != 7', this.
476                             finalScore);
477                         }
478                     })
479                     dec++;
480                 }
481             }
482         }
483     });
484 }

```

```

476         this.currentCheckedValue[position - 1] = null;
477         this.results[position - 1] = null;
478     } else {
479         this.currentCheckedValue[position - 1] = el.value;
480     }
481     });
482 }
483
484 openErrorDialog(message: string): void {
485     const dialogRef = this.dialog.open(ErrorDialogComponent, {
486         width: '500px',
487         height: '270px',
488         data: message
489     });
490
491     dialogRef.afterClosed().subscribe(result => {
492         console.log('Finestra chiusa');
493     });
494 }
495
496 goHome() {
497     if (this.user === null) {
498         this.router.navigateByUrl('/startup');
499     } else {
500         this.router.navigateByUrl('/landing');
501     }
502 }
503
504 }
505
506 import { Component, OnInit } from '@angular/core';
507 import { FormGroup } from '@angular/forms';
508 import { MatOptionSelectionChange } from '@angular/material/core';
509 import { EortcQlqResults, Q_EORTC_QLQ } from '../../../model/
    questionnaires/questionnaire-eortc-qlq';
510 import { QuestionnaireService } from '../../../services/
    questionnaire.service';
511 import { MatDialog } from '@angular/material/dialog';
512 import { ErrorDialogComponent } from '../../../user-actions/error-
    dialog/error-dialog.component';
513 import { Store } from '@ngrx/store';
514 import { AppState } from '../../../store/app.states';
515 import { ActivatedRoute, Router } from '@angular/router';
516
517 @Component({
518     selector: 'app-questionnaire-eortc-qlq-c30',
519     templateUrl: './questionnaire-eortc-qlq.component.html',
520     styleUrls: ['./questionnaire-eortc-qlq.component.css']
521 })

```

```

522 export class QuestionnaireEortcQlqComponent implements OnInit {
523
524   qForm: FormGroup;
525   dataSource = Q_EORTC_QLQ;
526   notFocused = false;
527   submitted = false;
528   show = false;
529   displayedColumns = ['position', 'question', 'value'];
530   values: string[] = ['1', '2', '3', '4'];
531   values2q: string[] = ['1', '2', '3', '4', '5', '6', '7'];
532   results = [];
533
534   /// toolbar
535   back = true;
536   title = 'EORTC-QLQ';
537   destination = '/questionnaires';
538   user = null;
539   uuid = null;
540   authorization: boolean;
541   doctorID: string;
542   patientSSN: string;
543
544   constructor(private questionnaireService: QuestionnaireService,
545               private store: Store<AppState>,
546               private route: ActivatedRoute,
547               public dialog: MatDialog,
548               private router: Router) { }
549
550   ngOnInit(): void {
551
552     // tslint:disable-next-line:prefer-for-of
553     for (let i = 0; i < this.dataSource.length; ++i) {
554       this.results.push(null);
555     }
556
557     this.user = JSON.parse(localStorage.getItem('user'));
558     this.route.params.subscribe(params => {
559       this.doctorID = params.doctorID;
560       this.patientSSN = params.patientSSN;
561       this.uuid = params.uuid;
562       this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
563       if (params.back.localeCompare('questionnaires') === 0) {
564         this.destination = 'questionnaires';
565       } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
566         this.destination = 'choose-questionnaire';
567       }
568     });

```

```

569 }
570
571 selectResult($event: MatOptionSelectionChange, value: string,
572             position: number) {
573   if ($event.isUserInput) {
574     this.results[position - 1] = value;
575     console.log(this.results);
576   }
577 }
578
579 onSubmit() {
580   this.submitted = true;
581   this.show = true;
582
583   if (this.results.length < this.dataSource.length) {
584     this.show = false;
585     this.submitted = false;
586
587     const msg = 'Non hai finito di completare il questionario!';
588     this.openDialog(msg);
589   } else {
590
591     let count = 0;
592
593     for (const r of this.results) {
594       if (r === null) {
595         count++;
596       }
597     }
598
599     // tslint:disable-next-line:max-line-length
600     if ((count === 2 && this.results[this.results.length - 1] ===
601         null && this.results[this.results.length - 2] === null) || (count
602         === 0)) {
603       const res = [];
604       this.results.forEach(r => res.push(parseInt(r, 10)));
605       console.log(res);
606       const resultToSend = new EortcQlqResults(this.doctorID,
607         this.patientSSN, res);
608       this.questionnaireService.sendForm(resultToSend, 'eortc-qlq',
609         this.uuid);
610     } else {
611       this.show = false;
612       this.submitted = false;
613       const msg = 'Non hai finito di completare il questionario!';
614       this.openDialog(msg);
615     }
616   }
617 }

```

```

613 }
614
615 openDialog(message: string): void {
616     const dialogRef = this.dialog.open(ErrorDialogComponent, {
617         width: '500px',
618         height: '270px',
619         data: message
620     });
621
622     dialogRef.afterClosed().subscribe(result => {
623         console.log('Finestra chiusa');
624     });
625 }
626
627 goHome() {
628     if (this.user === null) {
629         this.router.navigateByUrl('/startup');
630     } else {
631         this.router.navigateByUrl('/landing');
632     }
633 }
634 }
635
636
637 import {Component, OnInit, Renderer2, ViewChild} from '@angular/core';
638
639 import {
640     barChartData,
641     barChartLabels,
642     barChartOptions,
643     barChartType,
644     EQ5D5LResults,
645     Q_EQ_5D_5L
646 } from '../../../../../model/questionnaires/questionnaire-eq5d5l';
647 import {FormGroup} from '@angular/forms';
648 import {QuestionnaireService} from '../../../../services/questionnaire.service';
649 import {MatDialog} from '@angular/material/dialog';
650 import {ErrorDialogComponent} from '../../../../user-actions/error-dialog/error-dialog.component';
651 import {ActivatedRoute, Router} from '@angular/router';
652
653 @Component({
654     selector: 'app-questionnaire-eq5d5l',
655     templateUrl: './questionnaire-eq5d5l.component.html',
656     styleUrls: ['./questionnaire-eq5d5l.component.css']
657 })
658 export class QuestionnaireEq5d5lComponent implements OnInit {

```

```

659 qForm: FormGroup;
660 notFocused = false;
661 source = Q_EQ_5D_5L;
662 results = [];
663 submitted = false;
664 show = false;
665 currentCheckedValue = [null];
666 found = false;
667
668 /// toolbar
669 back = true;
670 destination = '/questionnaires';
671 title = 'EQ5D5L';
672 user = null;
673 uuid = null;
674 authorization: boolean;
675 doctorID: string;
676 patientSSN: string;
677 sliderValue = 0;
678
679 constructor(private questionnaireService: QuestionnaireService,
680             private ren: Renderer2,
681             public dialog: MatDialog,
682             private route: ActivatedRoute,
683             private router: Router) { }
684
685 ngOnInit(): void {
686     this.qForm = new FormGroup({
687         // health: new FormControl('', [Validators.required, Validators
688         .min(0), Validators.max(100)])
689     });
690
691     this.user = JSON.parse(localStorage.getItem('user'));
692     this.route.params.subscribe(params => {
693         this.doctorID = params.doctorID;
694         this.patientSSN = params.patientSSN;
695         this.uuid = params.uuid;
696         this.authorization = this.user.ssn.localeCompare(params.
697         patientSSN) === 0;
698         if (params.back.localeCompare('questionnaires') === 0) {
699             this.destination = 'questionnaires';
700         } else if (params.back.localeCompare('choose-questionnaire')
701         === 0) {
702             this.destination = 'choose-questionnaire';
703         }
704     });
705 }
706
707 onSubmit() {

```

```

705     this.submitted = true;
706     this.show = true;
707     this.found = false;
708
709     console.log(this.results);
710     if (this.results.length !== this.source.length) {
711         const msg = 'Non hai finito di completare il questionario!';
712         this.openDialog(msg);
713         this.show = false;
714         this.submitted = false;
715     } else {
716         for (const r of this.results) {
717             if (r === null) {
718                 const msg = 'Non hai finito di completare il questionario
!';
719                 this.openDialog(msg);
720                 this.show = false;
721                 this.submitted = false;
722                 this.found = true;
723                 break;
724             }
725         }
726
727         if (!this.found) {
728             console.log(this.doctorID + ' ' + this.patientSSN + ' ' +
this.results + ' ' + this.sliderValue);
729             const resultToSend = new EQ5D5LResults(this.doctorID, this.
patientSSN, this.results, this.sliderValue);
730             this.questionnaireService.sendForm(resultToSend, 'eq5d5l',
this.uuid);
731         }
732     }
733
734 }
735
736 openDialog(message: string): void {
737     const dialogRef = this.dialog.open(ErrorDialogComponent, {
738         width: '500px',
739         height: '270px',
740         data: message
741     });
742
743     dialogRef.afterClosed().subscribe(result => {
744         console.log('Finestra chiusa');
745     });
746 }
747
748 setAnswer(answer: string, position: number) {
749     this.results[position - 1] = answer;

```

```

750     console.log(this.results);
751 }
752
753 checkState(el, position: number) {
754     setTimeout(() => {
755         if (this.currentCheckedValue[position - 1] && this.
756             currentCheckedValue[position - 1] === el.value) {
757             el.checked = false;
758             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
759             focused');
760             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
761             program-focused');
762             this.currentCheckedValue[position - 1] = null;
763             this.results[position - 1] = null;
764         } else {
765             this.currentCheckedValue[position - 1] = el.value;
766         }
767     });
768 }
769
770 goHome() {
771     if (this.user === null) {
772         this.router.navigateByUrl('/startup');
773     } else {
774         this.router.navigateByUrl('/landing');
775     }
776 }
777
778 setHealth() {
779     console.log(this.sliderValue);
780 }
781
782 import {Component, OnInit, ViewChild} from '@angular/core';
783 import {QuestionnaireService} from '../services/
784     questionnaire.service';
785 import {FormControl, FormGroup, Validators} from '@angular/forms';
786 import {IBDDiskResults} from '../model/questionnaires/
787     questionnaire-ibddisk';
788 import {MatDialog} from '@angular/material/dialog';
789 import {ErrorDialogComponent} from '../user-actions/error-
790     dialog/error-dialog.component';
791 import {BaseChartDirective} from 'ng2-charts';
792 import {ActivatedRoute, Router} from '@angular/router';
793 import {GradimentoResults, Q_GRADIMENTO} from 'src/app/model/
794     questionnaires/questionnaire-gradimento';

```

```

792 @Component({
793   selector: 'app-questionnaire-gradimento',
794   templateUrl: './questionnaire-gradimento.component.html',
795   styleUrls: ['./questionnaire-gradimento.component.css']
796 })
797 export class QuestionnaireGradimentoComponent implements OnInit {
798   toShow: number[] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]; // forse da
       impostare a -1
799
800   resultToSend: GradimentoResults;
801
802
803   qForm: FormGroup;
804   dataSource = Q_GRADIMENTO;
805   displayedColumns = ['position', 'question', 'value'];
806   notFocused = false;
807   completed = false;
808   private found = false;
809   submitted = false;
810   show = false;
811
812   /// toolbar
813   back = true;
814   destination = '/landing';
815   title = 'Questionario di gradimento';
816   user = null;
817   uuid = null;
818   authorization: boolean;
819   already_done= false;
820   doctorID: string;
821   patientSSN: string;
822   sliderValue: number[] = [0];
823   box_suggerimenti: string = "";
824   window: any;
825
826   constructor(private questionnaireService: QuestionnaireService,
827               private route: ActivatedRoute,
828               public dialog: MatDialog,
829               private router: Router) {
830   }
831
832   ngOnInit(): void {
833
834     this.user = JSON.parse(localStorage.getItem('user'));
835
836     this.route.params.subscribe(params => {
837       this.patientSSN = params.patientSSN;
838       console.log("AUTH");
839       console.log(this.user.ssn);

```

```

840     console.log(params.patientSSN);
841     this.questionnaireService.checkCompiledGradimento(this.
patientSSN).subscribe( res => this.already_done=res );
842     this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
843     if (params.back.localeCompare('questionnaires') === 0) {
844         this.destination = 'questionnaires';
845     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
846         this.destination = 'choose-questionnaire';
847     }
848     });
849     this.qForm = new FormGroup({
850         boxSuggerimenti: new FormControl(''),
851     });
852     this.window = window;
853
854 }
855
856 selectResult(position: number) {
857     if(position===10)
858     {
859         this.box_suggerimenti = this.qForm.get("boxSuggerimenti").value
;
860     }
861 else {
862     thisToShow[position - 1] = this.sliderValue[position - 1];
863     console.log('SHOW: ' + thisToShow);
864 } }
865
866
867 onSubmit() {
868     this.submitted = true;
869     this.show = true;
870     this.found = false;
871
872     const sorted = [];
873
874     sorted.push(...thisToShow);
875
876     sorted.sort();
877
878
879     for (const s of sorted) {
880         if (s === -1) { // TODO: forse da impostare a -1
881             const msg = 'Non hai finito di completare il questionario!';
882             this.openDialog(msg);
883             this.found = true;
884             this.show = false;

```

```

885         this.submitted = false;
886         break;
887     }
888 }
889
890 if (!this.found) {
891     this.resultToSend = new GradimentoResults(this.patientSSN, this.
box_suggerimenti, this.toShow);
892     console.log(this.resultToSend);
893     this.questionnaireService.sendFormGradimento(this.resultToSend,
'gradimento');
894 }
895 }
896 }
897
898 openFileDialog(message: string): void {
899     const dialogRef = this.dialog.open(ErrorDialogComponent, {
900         width: '500px',
901         height: '270px',
902         data: message
903     });
904
905     dialogRef.afterClosed().subscribe(result => {
906         console.log('Finestra chiusa');
907     });
908 }
909
910 goHome() {
911     if (this.user === null) {
912         this.router.navigateByUrl('/startup');
913     } else {
914         this.router.navigateByUrl('/landing');
915     }
916 }
917 }
918
919
920 import { Component, OnInit } from '@angular/core';
921 import { FormGroup } from '@angular/forms';
922 import { QuestionnaireService } from '../../../services/
questionnaire.service';
923 import { MatDialog } from '@angular/material/dialog';
924 import { ActivatedRoute, Router } from '@angular/router';
925 import { MatOptionSelectionChange } from '@angular/material/core';
926 import { ErrorDialogComponent } from '../../../user-actions/error-
dialog/error-dialog.component';
927 import { HADSResults, Q_HADS } from '../../../model/questionnaires/
questionnaire-hads';
928

```

```

929 @Component({
930   selector: 'app-questionnaire-hads',
931   templateUrl: './questionnaire-hads.component.html',
932   styleUrls: ['./questionnaire-hads.component.css']
933 })
934 export class QuestionnaireHadsComponent implements OnInit {
935
936   qForm: FormGroup;
937   dataSource = Q_HADS;
938   notFocused = false;
939   submitted = false;
940   show = false;
941   displayedColumns = ['position', 'question', 'value'];
942   results = [];
943   q = 'question';
944   found = false;
945   finalScore = 0;
946
947   /// toolbar
948   back = true;
949   destination = '/questionnaires';
950   title = 'HADS';
951   user = null;
952   uuid = null;
953   authorization: boolean;
954   doctorID: string;
955   patientSSN: string;
956
957   constructor(private questionnaireService: QuestionnaireService,
958               public dialog: MatDialog,
959               private route: ActivatedRoute,
960               private router: Router) { }
961
962   ngOnInit(): void {
963     // tslint:disable-next-line:prefer-for-of
964     for (let i = 0; i < this.dataSource.length; ++i) {
965       this.results.push(null);
966     }
967
968     this.qForm = new FormGroup({});
969
970     this.user = JSON.parse(localStorage.getItem('user'));
971     this.route.params.subscribe(params => {
972       this.doctorID = params.doctorID;
973       this.patientSSN = params.patientSSN;
974       this.uuid = params.uuid;
975       this.authorization = this.user.ssn.localeCompare(params.
976         patientSSN) === 0;
977       if (params.back.localeCompare('questionnaires') === 0) {

```

```

977         this.destination = 'questionnaires';
978     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
979         this.destination = 'choose-questionnaire';
980     }
981     });
982 }
983
984 onSubmit(value: any) {
985     this.submitted = true;
986     this.show = true;
987     this.found = false;
988
989     if (this.results.length !== this.dataSource.length) {
990         this.show = false;
991         this.submitted = false;
992         this.found = true;
993         const msg = 'Non hai finito di completare il questionario!';
994         this.openErrorDialog(msg);
995     } else {
996         for (const r of this.results) {
997             if (r === null) {
998                 this.show = false;
999                 this.submitted = false;
1000                 this.found = true;
1001                 const msg = 'Non hai finito di completare il questionario
!';
1002                 this.openErrorDialog(msg);
1003                 break;
1004             }
1005         }
1006
1007         if (!this.found) {
1008             console.log(this.finalScore);
1009             const resultToSend = new HADSResults(this.doctorID, this.
patientSSN, this.results, this.finalScore);
1010             this.questionnaireService.sendForm(resultToSend, 'hads', this
.uuid);
1011         }
1012     }
1013 }
1014
1015 selectResult($event: MatOptionSelectionChange, value: string,
position: number, answers: string[]) {
1016     if ($event.isUserInput) {
1017         if (position === 2 || position === 4 || position === 6 ||
position === 8 ||
1018             position === 10 || position === 12 || position === 14) {
1019             if (this.results[position - 1] !== null) {

```

```

1020         let dec = 0;
1021         answers.forEach(answer => {
1022             if (answer === this.results[position - 1]) {
1023                 this.finalScore -= dec;
1024                 console.log('Final Score After:', this.finalScore);
1025             }
1026             dec++;
1027         });
1028     }
1029     let inc = 0;
1030     answers.forEach(answer => {
1031         if (answer === value) {
1032             this.finalScore += inc;
1033         }
1034         inc++;
1035     });
1036     } else if (position === 1 || position === 3 || position === 5
1037 || position === 7 ||
1038 position === 9 || position === 11 || position === 13) {
1039     if (this.results[position - 1] !== null) {
1040         let dec = answers.length - 1;
1041         answers.forEach(answer => {
1042             if (answer === this.results[position - 1]) {
1043                 this.finalScore -= dec;
1044                 console.log('Final Score After:', this.finalScore);
1045             }
1046             dec--;
1047         });
1048         let inc = answers.length - 1;
1049         answers.forEach(answer => {
1050             if (answer === value) {
1051                 this.finalScore += inc;
1052             }
1053             inc--;
1054         });
1055     }
1056     console.log('Final Score Value set:', this.finalScore);
1057     this.results[position - 1] = value;
1058     console.log(this.results);
1059 }
1060 }
1061
1062 openErrorDialog(message: string): void {
1063     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1064         width: '500px',
1065         height: '270px',
1066         data: message
1067     });

```

```

1068     dialogRef.afterClosed().subscribe(result => {
1069         console.log('Finestra chiusa');
1070     });
1071 }
1072
1073 goHome() {
1074     if (this.user === null) {
1075         this.router.navigateByUrl('/startup');
1076     } else {
1077         this.router.navigateByUrl('/landing');
1078     }
1079 }
1080 }
1081
1082 }
1083
1084 import {Component, Inject, OnInit} from '@angular/core';
1085 import {FormControl, FormGroup, Validators} from '@angular/forms';
1086 import {QuestionnaireService} from '../services/questionnaire.service';
1087 import {MAT_DIALOG_DATA, MatDialog, MatDialogRef} from '@angular/
1088     material/dialog';
1089 import {MatOptionSelectionChange} from '@angular/material/core';
1090 import {ErrorDialogComponent} from '../user-actions/error-
1091     dialog/error-dialog.component';
1092 import {HBIResults, Q_HBI} from '../model/questionnaires/
1093     questionnaire-hbi';
1094 import {MatCheckboxChange} from '@angular/material/checkbox';
1095 import {ActivatedRoute, Router} from '@angular/router';
1096
1097 export interface HbiDialogData {
1098     message: string;
1099     values: string[];
1100 }
1101
1102 @Component({
1103     selector: 'app-questionnaire-hbi',
1104     templateUrl: './questionnaire-hbi.component.html',
1105     styleUrls: ['./questionnaire-hbi.component.css']
1106 })
1107 export class QuestionnaireHbiComponent implements OnInit {
1108     qForm: FormGroup;
1109     dataSource = Q_HBI;
1110     notFocused = false;
1111     submitted = false;
1112     show = false;

```

```

1113 displayedColumns = [ 'position', 'question', 'value' ];
1114 values: string [] = [
1115     'Artralgia ',
1116     'Uveite ',
1117     'Eritema nodoso ',
1118     'Ulcera aftoide ',
1119     'Pioderma gangrenoso ',
1120     'Fissurazione anale ',
1121     'Comparsa di una nuova fistola ',
1122     'Ascesso '];
1123 dialogResults = [ ' '];
1124 dialogScore = 0;
1125 results = [];
1126 q = 'question';
1127 found = false;
1128
1129 /// toolbar
1130 back = true;
1131 destination = '/questionnaires';
1132 title = 'HBI';
1133 user = null;
1134 uuid = null;
1135 authorization: boolean;
1136 doctorID: string;
1137 patientSSN: string;
1138
1139 constructor(private questionnaireService: QuestionnaireService,
1140             public dialog: MatDialog,
1141             private route: ActivatedRoute,
1142             private router: Router) { }
1143
1144 ngOnInit(): void {
1145     this.qForm = new FormGroup({
1146         question3: new FormControl({ value: '' }, [ Validators.required,
1147             Validators.min(0), Validators.max(25) ]),
1148     });
1149     // tslint:disable-next-line:prefer-for-of
1150     for (let i = 0; i < this.dataSource.length; ++i) {
1151         this.results.push(null);
1152     }
1153
1154     this.user = JSON.parse(localStorage.getItem('user'));
1155     this.route.params.subscribe(params => {
1156         this.doctorID = params.doctorID;
1157         this.patientSSN = params.patientSSN;
1158         this.uuid = params.uuid;
1159         this.authorization = this.user.ssn.localeCompare(params.
1160             patientSSN) === 0;
1161         if (params.back.localeCompare('questionnaires') === 0) {

```

```

1160         this.destination = 'questionnaires';
1161     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
1162         this.destination = 'choose-questionnaire';
1163     }
1164     });
1165 }
1166
1167 selectResult($event: MatOptionSelectionChange, value: string,
position: number) {
1168     if ($event.isUserInput) {
1169         if (position !== 5) {
1170             this.results[position - 1] = value.slice(0, 1);
1171             console.log(this.results);
1172         } else {
1173             if (value === 'Si') {
1174                 this.openHbiDialog($event, position, 'Presenza di
complicanze');
1175             } else if (value === 'No') {
1176                 this.results[position - 1] = '0';
1177                 console.log(this.results);
1178             }
1179         }
1180     }
1181 }
1182
1183 setResult(position: number) {
1184     const question = 'question' + position;
1185     if (position === 3) {
1186         this.results[position - 1] = this.qForm.get(question).value.
toString();
1187     }
1188 }
1189
1190 onSubmit(value) {
1191     this.submitted = true;
1192     this.show = true;
1193     this.found = false;
1194
1195     if (this.results.length !== this.dataSource.length) {
1196         this.show = false;
1197         this.submitted = false;
1198         this.found = true;
1199         const msg = 'Non hai finito di completare il questionario!';
1200         this.openErrorDialog(msg);
1201     } else {
1202         for (const r of this.results) {
1203             if (r === null) {
1204                 this.show = false;

```

```

1205         this.submitted = false;
1206         this.found = true;
1207         const msg = 'Non hai finito di completare il questionario
!';
1208         this.openErrorDialog(msg);
1209         break;
1210     }
1211 }
1212
1213     if (!this.found) {
1214         let finalScore = 0;
1215
1216         for (let i = 0; i < this.results.length - 1; ++i) {
1217             finalScore += parseInt(this.results[i], 10);
1218         }
1219
1220         finalScore += this.dialogScore;
1221         console.log(finalScore);
1222         const resultToSend = new HBIResults(this.doctorID, this.
patientSSN, this.results, this.dialogResults, finalScore);
1223         this.questionnaireService.sendForm(resultToSend, 'hbi',
this.uuid);
1224     }
1225 }
1226 }
1227
1228 openErrorDialog(message: string): void {
1229     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1230         width: '500px',
1231         height: '270px',
1232         data: message
1233     });
1234
1235     dialogRef.afterClosed().subscribe(result => {
1236         console.log('Finestra chiusa');
1237     });
1238 }
1239
1240 openHbiDialog($event: MatOptionSelectionChange, position: number,
message: string): void {
1241     const dialogRef = this.dialog.open(HbiDialogComponent, {
1242         width: '500px',
1243         height: '400px',
1244         data: {message, values: this.values}
1245     });
1246
1247     dialogRef.afterClosed().subscribe(result => {
1248
1249         if (result === undefined) {

```

```

1250         this.openErrorDialog('Non hai effettuato alcuna selezione!');
1251         $event.source.deselect();
1252         this.results[position - 1] = null;
1253     } else {
1254         this.dialogResults = result;
1255         this.dialogScore = this.dialogResults.length;
1256         this.results[position - 1] = this.dialogScore.toString(10);
1257     }
1258     console.log('Finestra chiusa');
1259     console.log(this.results);
1260 });
1261 }
1262
1263 goHome() {
1264     if (this.user === null) {
1265         this.router.navigateByUrl('/startup');
1266     } else {
1267         this.router.navigateByUrl('/landing');
1268     }
1269 }
1270 }
1271
1272 @Component({
1273     selector: 'app-hbi-dialog',
1274     templateUrl: './hbi-dialog.component.html',
1275     styleUrls: ['./hbi-dialog.component.css']
1276 })
1277 export class HbiDialogComponent implements OnInit {
1278
1279     results = [];
1280
1281     constructor(public dialogRef: MatDialogRef<ErrorDialogComponent>,
1282                 @Inject(MAT_DIALOG_DATA) public data: HbiDialogData) {
1283
1284     }
1285
1286     ngOnInit(): void {
1287
1288     }
1289
1290     cancel() {
1291         this.dialogRef.close();
1292     }
1293
1294     setResults($event: MatCheckboxChange, value: string) {
1295         if ($event.checked) {
1296             this.results.push(value);
1297         } else {
1298             this.results.forEach( (r, index) => {
1299                 if (r.localeCompare(value) === 0) {
1300                     this.results.splice(index, 1);
1301                 }
1302             });
1303         }
1304     }
1305 }

```

```

1298         }
1299     });
1300 }
1301 console.log(this.results);
1302 }
1303 }
1304
1305
1306 import {Component, OnInit, ViewChild} from '@angular/core';
1307 import {QuestionnaireService} from '../services/questionnaire.service';
1308 import {FormGroup} from '@angular/forms';
1309 import {
1310     IBDDiskResults,
1311     Q_IBD_DISK,
1312     radarChartData,
1313     radarChartLabels,
1314     radarChartOptions,
1315     radarChartType
1316 } from '../model/questionnaires/questionnaire-ibddisk';
1317 import {MatDialog} from '@angular/material/dialog';
1318 import {ErrorDialogComponent} from '../user-actions/error-dialog/error-dialog.component';
1319 import {BaseChartDirective} from 'ng2-charts';
1320 import {ActivatedRoute, Router} from '@angular/router';
1321
1322 @Component({
1323     selector: 'app-questionnaire-ibd-disk',
1324     templateUrl: './questionnaire-ibd-disk.component.html',
1325     styleUrls: ['./questionnaire-ibd-disk.component.css']
1326 })
1327
1328 export class QuestionnaireIbdDiskComponent implements OnInit {
1329     toShow: number[] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]; // forse da
1330         impostare a -1
1331
1332     resultToSend: IBDDiskResults;
1333
1334     // FOR RADAR CHART CONFIG, SEE questionnaire-ibddisk.ts
1335     radarChartLabels = radarChartLabels;
1336     radarChartData = radarChartData;
1337     radarChartType = radarChartType;
1338     radarChartOptions = radarChartOptions;
1339
1340     @ViewChild('baseChart') radarChart: BaseChartDirective;
1341
1342     qForm: FormGroup;
1343     dataSource = Q_IBD_DISK;

```

```

1343 displayedColumns = [ 'position', 'question', 'description', 'value
    '];
1344 notFocused = false;
1345 completed = false;
1346 private found = false;
1347 submitted = false;
1348 show = false;
1349
1350 /// toolbar
1351 back = true;
1352 destination = '/questionnaires';
1353 title = 'IBD-DISK';
1354 user = null;
1355 uuid = null;
1356 authorization: boolean;
1357 doctorID: string;
1358 patientSSN: string;
1359 sliderValue: number[] = [-1];
1360 window: any;
1361
1362 constructor(private questionnaireService: QuestionnaireService,
1363             private route: ActivatedRoute,
1364             public dialog: MatDialog,
1365             private router: Router) {
1366 }
1367
1368 ngOnInit(): void {
1369     this.user = JSON.parse(localStorage.getItem('user'));
1370     this.route.params.subscribe(params => {
1371         this.doctorID = params.doctorID;
1372         this.patientSSN = params.patientSSN;
1373         this.uuid = params.uuid;
1374         this.authorization = this.user.ssn.localeCompare(params.
1375 patientSSN) === 0;
1376         if (params.back.localeCompare('questionnaires') === 0) {
1377             this.destination = 'questionnaires';
1378         } else if (params.back.localeCompare('choose-questionnaire')
1379 === 0) {
1380             this.destination = 'choose-questionnaire';
1381         }
1382     });
1383     this.window = window;
1384 }
1385
1386 selectResult(position: number) {
1387     this.toShow[position - 1] = this.sliderValue[position - 1];
1388
1389     console.log('SHOW: ' + this.toShow);
1390     console.log('DATA: ' + this.radarChartData[0].data);

```

```
1389     console.log(this.radarChartData);
1390
1391     this.radarChartData[0] = Object.assign({}, this.radarChartData
1392     [0], {
1393         data: this.toShow
1394     });
1395
1396     this.reloadChart();
1397
1398 }
1399
1400 reloadChart() {
1401     if (this.radarChart !== undefined) {
1402         this.radarChart.chart.destroy();
1403         this.radarChart.chart = null;
1404
1405         this.radarChart.datasets = this.radarChartData;
1406         this.radarChart.labels = this.radarChartLabels;
1407         this.radarChart.ngOnInit();
1408     }
1409 }
1410
1411 onSubmit() {
1412     this.submitted = true;
1413     this.show = true;
1414     this.found = false;
1415
1416     const sorted = [];
1417
1418     sorted.push(...this.toShow);
1419
1420     sorted.sort();
1421
1422     console.log('SORTED: ' + sorted);
1423     console.log('RADAR: ');
1424
1425     for (const s of sorted) {
1426         if (s === -1) { // TODO: forse da impostare a -1
1427             const msg = 'Non hai finito di completare il questionario!';
1428             this.openDialog(msg);
1429             this.found = true;
1430             this.show = false;
1431             this.submitted = false;
1432             break;
1433         }
1434     }
1435
1436     if (!this.found) {
```

```

1437     this.resultToSend = new IBDDiskResults(this.doctorID, this.
1438     patientSSN, this.toShow);
1439     this.questionnaireService.sendForm(this.resultToSend, 'ibd-disk
1440     ', this.uuid);
1441   }
1442   openDialog(message: string): void {
1443     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1444       width: '500px',
1445       height: '270px',
1446       data: message
1447     });
1448     dialogRef.afterClosed().subscribe(result => {
1449       console.log('Finestra chiusa');
1450     });
1451   }
1452 }
1453
1454 goHome() {
1455   if (this.user === null) {
1456     this.router.navigateByUrl('/startup');
1457   } else {
1458     this.router.navigateByUrl('/landing');
1459   }
1460 }
1461 }
1462
1463 import {Component, OnInit, Renderer2} from '@angular/core';
1464 import {FormGroup} from '@angular/forms';
1465 import {QuestionnaireService} from '../../services/
1466     questionnaire.service';
1467 import {MatDialog} from '@angular/material/dialog';
1468 import {ActivatedRoute, Router} from '@angular/router';
1469 import {ErrorDialogComponent} from '../../user-actions/error-
1470     dialog/error-dialog.component';
1471 import {MatRadioButton} from '@angular/material/radio';
1472 import {Q_IBDQ, IBDQResults} from '../../model/questionnaires/
1473     questionnaire-ibdq';
1474
1475 @Component({
1476   selector: 'app-questionnaire-ibdq',
1477   templateUrl: './questionnaire-ibdq.component.html',
1478   styleUrls: ['./questionnaire-ibdq.component.css']
1479 })
1480 export class QuestionnaireIbdqComponent implements OnInit {
1481   qForm: FormGroup;
1482   notFocused = false;

```

```

1481 source = Q_IBDQ;
1482 results = [];
1483 submitted = false;
1484 show = false;
1485 currentCheckedValue = [null];
1486 found = false;
1487
1488 /// toolbar
1489 back = true;
1490 destination = '/questionnaires';
1491 title = 'IBDQ';
1492 user = null;
1493 uuid = null;
1494 authorization: boolean;
1495 doctorID: string;
1496 patientSSN: string;
1497
1498 constructor(private questionnaireService: QuestionnaireService,
1499             private ren: Renderer2,
1500             public dialog: MatDialog,
1501             private route: ActivatedRoute,
1502             private router: Router) {}
1503
1504 ngOnInit(): void {
1505     // tslint:disable-next-line:prefer-for-of
1506     for (let i = 0; i < this.source.length; ++i) {
1507         this.results.push(null);
1508     }
1509
1510     this.user = JSON.parse(localStorage.getItem('user'));
1511     this.route.params.subscribe(params => {
1512         this.doctorID = params.doctorID;
1513         this.patientSSN = params.patientSSN;
1514         this.uuid = params.uuid;
1515         this.authorization = this.user.ssn.localeCompare(params.
1516         patientSSN) === 0;
1517         if (params.back.localeCompare('questionnaires') === 0) {
1518             this.destination = 'questionnaires';
1519         } else if (params.back.localeCompare('choose-questionnaire')
1520         === 0) {
1521             this.destination = 'choose-questionnaire';
1522         }
1523     });
1524 }
1525
1526 openDialog(message: string): void {
1527     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1528         width: '500px',
1529         height: '270px',

```

```

1528     data: message
1529   });
1530
1531   dialogRef.afterClosed().subscribe(result => {
1532     console.log('Finestra chiusa');
1533   });
1534 }
1535
1536 setAnswer(button: MatRadioButton, answer: string, position: number)
1537 {
1538   this.results[position - 1] = answer;
1539   console.log(this.results);
1540 }
1541
1542 checkState(el, position: number) {
1543   setTimeout(() => {
1544     if (this.currentCheckedValue[position - 1] && this.
1545     currentCheckedValue[position - 1] === el.value) {
1546       el.checked = false;
1547       this.renderer.removeClass(el._elementRef.nativeElement, 'cdk-
1548       focused');
1549       this.renderer.removeClass(el._elementRef.nativeElement, 'cdk-
1550       program-focused');
1551       this.currentCheckedValue[position - 1] = null;
1552       this.results[position - 1] = null;
1553     } else {
1554       this.currentCheckedValue[position - 1] = el.value;
1555     }
1556   });
1557 }
1558
1559 onSubmit() {
1560   this.submitted = true;
1561   this.show = true;
1562   this.found = false;
1563
1564   console.log(this.results);
1565   if (this.results.length !== this.source.length) {
1566     const msg = 'Non hai finito di completare il questionario!';
1567     this.openDialog(msg);
1568     this.show = false;
1569     this.submitted = false;
1570   } else {
1571     for (const r of this.results) {
1572       if (r === null) {
1573         const msg = 'Non hai finito di completare il questionario
1574         !';
1575         this.openDialog(msg);
1576         this.show = false;

```

```

1572         this.submitted = false;
1573         this.found = true;
1574         break;
1575     }
1576 }
1577
1578 if (!this.found) {
1579     let finalScore = 0;
1580     // tslint:disable-next-line:prefer-for-of
1581     for (let i = 0; i < this.results.length; ++i) {
1582         finalScore += parseInt(this.results[i].split('-')[0][0],
1583     10);
1584     }
1585     const resultToSend = new IBDQResults(this.doctorID, this.
1586 patientSSN, this.results, finalScore);
1587     this.questionnaireService.sendForm(resultToSend, 'ibdq', this
1588 .uuid);
1589 }
1590 }
1591 }
1592 goHome() {
1593     if (this.user === null) {
1594         this.router.navigateByUrl('/startup');
1595     } else {
1596         this.router.navigateByUrl('/landing');
1597     }
1598 }
1599 }
1600
1601 import {Component, OnInit, Renderer2} from '@angular/core';
1602 import {AbstractControl, FormControl, FormGroup, ValidatorFn,
1603     Validators} from '@angular/forms';
1604 import {QuestionnaireService} from '../services/
1605 questionnaire.service';
1606 import {MatDialog} from '@angular/material/dialog';
1607 import {ActivatedRoute, Router} from '@angular/router';
1608 import {MatOptionSelectionChange} from '@angular/material/core';
1609 import {ErrorDialogComponent} from '../user-actions/error-
1610 dialog/error-dialog.component';
1611 import {IPAQSFResults, Q_IPAQ_SF} from '../model/
1612 questionnaires/questionnaire-ipaq-sf';
1613
1614 @Component({
1615     selector: 'app-questionnaire-ipaq-sf',
1616     templateUrl: './questionnaire-ipaq-sf.component.html',
1617     styleUrls: ['./questionnaire-ipaq-sf.component.css']
1618 })

```

```

1614 export class QuestionnaireIpaqSfComponent implements OnInit {
1615
1616     qForm: FormGroup;
1617     dataSource = Q_IPAQ_SF;
1618     notFocused = false;
1619     submitted = false;
1620     show = false;
1621     results = [];
1622     q = 'question';
1623     currentCheckedValue = [null];
1624     found = false;
1625
1626     /// toolbar
1627     back = true;
1628     destination = '/questionnaires';
1629     title = 'IPAQ-SF';
1630     user = null;
1631     uuid = null;
1632     authorization: boolean;
1633     doctorID: string;
1634     patientSSN: string;
1635     box1bDisabled = true;
1636     box2bDisabled = true;
1637     box3bDisabled = true;
1638
1639
1640     constructor(private questionnaireService: QuestionnaireService,
1641                 public dialog: MatDialog,
1642                 private ren: Renderer2,
1643                 private route: ActivatedRoute,
1644                 private router: Router) { }
1645
1646     ngOnInit(): void {
1647         this.qForm = new FormGroup({
1648             question1: new FormControl({ value: '' }, [/* Validators.required
1649             ,*/ Validators.min(0), Validators.max(7)]),
1650             question2: new FormControl({ value: '' }, [/* Validators.required
1651             ,*/ Validators.min(0)]),
1652             question3: new FormControl({ value: '' }, [/* Validators.required
1653             ,*/ Validators.min(0), Validators.max(7)]),
1654             question4: new FormControl({ value: '' }, [/* Validators.required
1655             ,*/ Validators.min(0)]),
1656             question5: new FormControl({ value: '' }, [/* Validators.required
1657             ,*/ Validators.min(0), Validators.max(7)]),
1658             question6: new FormControl({ value: '' }, [/* Validators.required,
1659             Validators.min(0), Validators.max(7)*/]),
1660             question8: new FormControl({ value: '' }, [Validators.required,
1661             Validators.min(0)]),

```

```

1655     question9: new FormControl({ value: '' }, [Validators.required,
Validators.min(0)]),
1656   });
1657   // tslint:disable-next-line:prefer-for-of
1658   for (let i = 0; i < this.dataSource.length; ++i) {
1659     this.results.push(null);
1660   }
1661
1662   this.user = JSON.parse(localStorage.getItem('user'));
1663   this.route.params.subscribe(params => {
1664     this.doctorID = params.doctorID;
1665     this.patientSSN = params.patientSSN;
1666     this.uuid = params.uuid;
1667     this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
1668     if (params.back.localeCompare('questionnaires') === 0) {
1669       this.destination = 'questionnaires';
1670     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
1671       this.destination = 'choose-questionnaire';
1672     }
1673   });
1674 }
1675
1676 checkState(el, position: number) {
1677   setTimeout(() => {
1678     if (this.currentCheckedValue[position - 1] && this.
currentCheckedValue[position - 1] === el.value) {
1679       el.checked = false;
1680       this.ren.removeClass(el.__elementRef.nativeElement, 'cdk-
focused');
1681       this.ren.removeClass(el.__elementRef.nativeElement, 'cdk-
program-focused');
1682       this.currentCheckedValue[position - 1] = null;
1683       this.results[position - 1] = null;
1684     } else {
1685       this.currentCheckedValue[position - 1] = el.value;
1686     }
1687   });
1688 }
1689
1690 setAnswer(answer: string, position: number) {
1691   this.results[position - 1] = answer;
1692   //console.log(this.results);
1693 }
1694
1695 setResult(position: number) {
1696   const question = 'question' + position;
1697

```

```

1698 // tslint:disable-next-line:max-line-length
1699 if (this.qForm.get(question).value !== null && this.qForm.get(
question).value >= 0) {
1700     this.results[position - 1] = this.qForm.get(question).value.
toString();
1701 } else {
1702     this.results[position - 1] = null;
1703 }
1704
1705 if(position === 1 && this.results[position - 1] > 0){
1706     this.box1bDisabled = false
1707 }
1708 if(position === 3 && this.results[position - 1] > 0){
1709     this.box2bDisabled = false
1710 }
1711 if(position === 5 && this.results[position - 1] > 0){
1712     this.box3bDisabled = false
1713 }
1714 console.log(this.results);
1715 }
1716
1717 onSubmit() {
1718     this.submitted = true;
1719     this.show = true;
1720     this.found = false;
1721
1722     if (this.results.length !== this.dataSource.length) {
1723         this.show = false;
1724         this.submitted = false;
1725         this.found = true;
1726         const msg = 'Non hai finito di completare il questionario!';
1727         this.openErrorDialog(msg);
1728     } else {
1729         // tslint:disable-next-line:prefer-for-of
1730         for (let i = 0; i < this.results.length; i++) {
1731             if (i === 0 || i === 2 || i === 4) {
1732                 if (this.results[i] === '0' || this.results[i] === null) {
1733                     if (this.results[i + 1] !== null) {
1734                         this.show = false;
1735                         this.submitted = false;
1736                         this.found = true;
1737                         let j = 0;
1738                         if (i === 0) {
1739                             j = 1;
1740                         } else if (i === 2) {
1741                             j = 2;
1742                         } else if (i === 4) {
1743                             j = 3;
1744                         }

```

```

1745         // const msg = 'Non hai completato correttamente il
questionario!\n Controlla le domande 1a), 2a) e 3a)';
1746         const msg = 'Non hai completato correttamente il
questionario!\n' +
1747             'Hai compilato il campo della domanda ' + j + 'b)
nonostante nella domanda ' + j + 'a) tu abbia lasciato il campo
vuoto o uguale a 0';
1748         this.openErrorDialog(msg);
1749         break;
1750     }
1751     } /* else {
1752         if (this.results[i] === null) {
1753             this.show = false;
1754             this.submitted = false;
1755             this.found = true;
1756             const msg = 'Non hai finito di completare il
questionario!';
1757             this.openErrorDialog(msg);
1758             break;
1759         }
1760     } */
1761     } else if (i !== 1 && i !== 3 && i !== 5) {
1762         if (i === 6) {
1763             if (this.results[i] === null && (this.results[i - 1] !==
null || this.results[i - 2] !== null)){
1764                 this.show = false;
1765                 this.submitted = false;
1766                 this.found = true;
1767                 const msg = 'Non hai finito di completare il
questionario! Risposta 3c) incompleta';
1768                 this.openErrorDialog(msg);
1769                 break;
1770             }
1771         } else {
1772             if (this.results[i] === null) {
1773                 this.show = false;
1774                 this.submitted = false;
1775                 this.found = true;
1776                 const msg = 'Non hai finito di completare il
questionario!';
1777                 this.openErrorDialog(msg);
1778                 break;
1779             }
1780         }
1781     }
1782 }
1783 }
1784
1785 if (!this.found) {

```

```

1786     let diagnosis = '';
1787     let q1 = this.qForm.get('question1').value;
1788     if (q1 === null || q1.value === null || q1.value === '') {
1789         this.results[0] = 0;
1790         q1 = 0;
1791     }
1792     let q2 = this.qForm.get('question2').value;
1793     if (q2 === null || q2.value === null || q2.value === '') {
1794         this.results[1] = 0;
1795         q2 = 0;
1796     }
1797     // const intense = (this.qForm.get('question1').value * this.
qForm.get('question2').value) * 8;
1798     const intense = (q1 * q2) * 8;
1799     console.log(q2);
1800     let q3 = this.qForm.get('question3').value;
1801     if (q3 === null || q3.value === null || q3.value === '') {
1802         this.results[2] = 0;
1803         q3 = 0;
1804     }
1805     let q4 = this.qForm.get('question4').value;
1806     if (q4 === null || q4.value === null || q4.value === '') {
1807         this.results[3] = 0;
1808         q4 = 0;
1809     }
1810     // const moderate = (this.qForm.get('question3').value * this.
qForm.get('question4').value) * 4;
1811     const moderate = (q3 * q4) * 4;
1812
1813     let q5 = this.qForm.get('question5').value;
1814     if (q5 === null || q5.value === null || q5.value === '') {
1815         this.results[4] = 0;
1816         q5 = 0;
1817     }
1818     let q6 = this.qForm.get('question6').value;
1819     if (q6 === null || q6.value === null || q6.value === '') {
1820         this.results[5] = 0;
1821         q6 = 0;
1822     }
1823
1824     let gradeIntensity = 0;
1825     if (this.results[6] !== null && this.results[6].includes('
INTENSO')) {
1826         gradeIntensity = 3.3;
1827     } else if (this.results[6] !== null && this.results[6].includes
('MODERATO')) {
1828         gradeIntensity = 3;
1829     } else if (this.results[6] !== null && this.results[6].includes
('LENTO')) {

```

```

1830         gradeIntensity = 2.5;
1831     }
1832     if (this.results[6] === null) {
1833         this.results[6] = '';
1834     }
1835
1836     // const walk = (this.qForm.get('question5').value * this.qForm
1837     .get('question6').value) * gradeIntensity;
1838     const walk = (q5 * q6) * gradeIntensity;
1839
1840     const total = intense + moderate + walk;
1841     console.log('total ' + total);
1842
1843     console.log(intense, moderate, walk, total);
1844     if (total < 700) {
1845         diagnosis = 'SEI INATTIVO';
1846     } else if (total >= 700 && total <= 2519) {
1847         diagnosis = 'SEI SUFFICIENTEMENTE ATTIVO';
1848     } else {
1849         diagnosis = 'SEI ATTIVO O MOLTO ATTIVO';
1850     }
1851     console.log(total, diagnosis);
1852     const resultToSend = new IPAQSFResults(this.doctorID, this.
1853     patientSSN, this.results, diagnosis);
1854     this.questionnaireService.sendForm(resultToSend, 'ipaq-sf',
1855     this.uuid);
1856 }
1857
1858 openErrorDialog(message: string): void {
1859     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1860         width: '500px',
1861         height: '270px',
1862         data: message
1863     });
1864     dialogRef.afterClosed().subscribe(result => {
1865         console.log('Finestra chiusa');
1866     });
1867 }
1868
1869 goHome() {
1870     if (this.user === null) {
1871         this.router.navigateByUrl('/startup');
1872     } else {
1873         this.router.navigateByUrl('/landing');
1874     }
1875 }

```

```

1876  /*validateBox1b(): ValidatorFn {
1877      return (control: AbstractControl): { [key: string]: any } | null
      => {
1878          const p = control.value;
1879          console.log(control.value);
1880          if ((this.results[0] === undefined || this.results[0] === 0 ||
this.results[0] === null) &&
1881              (p !== undefined && p !== 0 && p !== null)) {
1882              return {minLengthError: 'Risposta 1b non concorde alla
risposta 1a'};
1883          } else {
1884              console.log("false");
1885          }
1886          return null;
1887      };
1888  }*/
1889  }
1890
1891
1892  import {Component, OnInit, Renderer2} from '@angular/core';
1893  import {FormGroup} from '@angular/forms';
1894  import {QuestionnaireService} from '../services/
questionnaire.service';
1895  import {MatDialog} from '@angular/material/dialog';
1896  import {ActivatedRoute, Router} from '@angular/router';
1897  import {ErrorDialogComponent} from '../user-actions/error-
dialog/error-dialog.component';
1898  import {MatRadioButton} from '@angular/material/radio';
1899  import {Q_LARS, LARSResults} from '../model/questionnaires/
questionnaire-lars';
1900
1901  @Component({
1902      selector: 'app-questionnaire-lars',
1903      templateUrl: './questionnaire-lars.component.html',
1904      styleUrls: ['./questionnaire-lars.component.css']
1905  })
1906  export class QuestionnaireLarsComponent implements OnInit {
1907
1908      qForm: FormGroup;
1909      notFocused = false;
1910      source = Q_LARS;
1911      results = [];
1912      submitted = false;
1913      show = false;
1914      currentCheckedValue = [null];
1915      found = false;
1916
1917      /// toolbar
1918      back = true;

```

```

1919 destination = '/questionnaires';
1920 title = 'LARS';
1921 user = null;
1922 uuid = null;
1923 authorization: boolean;
1924 doctorID: string;
1925 patientSSN: string;
1926
1927 constructor(private questionnaireService: QuestionnaireService,
1928             private ren: Renderer2,
1929             public dialog: MatDialog,
1930             private route: ActivatedRoute,
1931             private router: Router) {}
1932
1933 ngOnInit(): void {
1934     // tslint:disable-next-line:prefer-for-of
1935     for (let i = 0; i < this.source.length; ++i) {
1936         this.results.push(null);
1937     }
1938
1939     this.user = JSON.parse(localStorage.getItem('user'));
1940     this.route.params.subscribe(params => {
1941         this.doctorID = params.doctorID;
1942         this.patientSSN = params.patientSSN;
1943         this.uuid = params.uuid;
1944         this.authorization = this.user.ssn.localeCompare(params.
1945         patientSSN) === 0;
1946         if (params.back.localeCompare('questionnaires') === 0) {
1947             this.destination = 'questionnaires';
1948         } else if (params.back.localeCompare('choose-questionnaire')
1949         === 0) {
1950             this.destination = 'choose-questionnaire';
1951         }
1952     });
1953 }
1954
1955 openDialog(message: string): void {
1956     const dialogRef = this.dialog.open(ErrorDialogComponent, {
1957         width: '500px',
1958         height: '270px',
1959         data: message
1960     });
1961
1962     dialogRef.afterClosed().subscribe(result => {
1963         console.log('Finestra chiusa');
1964     });
1965 }

```

```

1965 setAnswer(button: MatRadioButton, answer: string, position: number)
1966 {
1967     this.results[position - 1] = answer;
1968     console.log(this.results);
1969 }
1970
1971 checkState(el, position: number) {
1972     setTimeout(() => {
1973         if (this.currentCheckedValue[position - 1] && this.
1974             currentCheckedValue[position - 1] === el.value) {
1975             el.checked = false;
1976             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
1977                 focused');
1978             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
1979                 program-focused');
1980             this.currentCheckedValue[position - 1] = null;
1981             this.results[position - 1] = null;
1982         } else {
1983             this.currentCheckedValue[position - 1] = el.value;
1984         }
1985     });
1986 }
1987
1988 onSubmit() {
1989     this.submitted = true;
1990     this.show = true;
1991     this.found = false;
1992
1993     console.log(this.results);
1994     if (this.results.length !== this.source.length) {
1995         const msg = 'Non hai finito di completare il questionario!';
1996         this.openDialog(msg);
1997         this.show = false;
1998         this.submitted = false;
1999     } else {
2000         for (const r of this.results) {
2001             if (r === null) {
2002                 const msg = 'Non hai finito di completare il questionario
2003                 !';
2004                 this.openDialog(msg);
2005                 this.show = false;
2006                 this.submitted = false;
2007                 this.found = true;
2008                 break;
2009             }
2010         }
2011     }
2012
2013     if (!this.found) {
2014         let finalScore = 0;

```

```

2009         // tslint:disable-next-line:prefer-for-of
2010         for (let i = 0; i < this.results.length; ++i) {
2011             finalScore += parseInt(this.results[i].split('-')[0][0],
2012             10);
2013         }
2014         const resultToSend = new LARSResults(this.doctorID, this.
2015         patientSSN, this.results, finalScore);
2016         this.questionnaireService.sendForm(resultToSend, 'lars', this
2017         .uuid);
2018     }
2019 }
2020 }
2021 }
2022 }
2023 }
2024 }
2025 }
2026 }
2027 }
2028 }
2029 import { Component, OnInit } from '@angular/core';
2030 import { FormControl, FormGroup, Validators } from '@angular/forms';
2031 import { QuestionnaireService } from '../services/
2032     questionnaire.service';
2033 import { MatDialog } from '@angular/material/dialog';
2034 import { ActivatedRoute, Router } from '@angular/router';
2035 import { MatOptionSelectionChange } from '@angular/material/core';
2036 import { MatDialogComponent } from '../user-actions/error-
2037     dialog/error-dialog.component';
2038 import { MIAHCDResults, Q_MIAH_CD } from '../model/
2039     questionnaires/questionnaire-miah-cd';
2040
2041 @Component({
2042     selector: 'app-questionnaire-miah-cd',
2043     templateUrl: './questionnaire-miah-cd.component.html',
2044     styleUrls: ['./questionnaire-miah-cd.component.css']
2045 })
2046 export class QuestionnaireMiahCdComponent implements OnInit {
2047
2048     qForm: FormGroup;
2049     dataSource = Q_MIAH_CD;
2050     notFocused = false;
2051     submitted = false;
2052     show = false;
2053     displayedColumns = ['position', 'question', 'value'];
2054     results = [];

```

```

2052 q = 'question';
2053 found = false;
2054
2055 /// toolbar
2056 back = true;
2057 destination = '/questionnaires';
2058 title = 'MIAH-CD';
2059 user = null;
2060 uuid = null;
2061 authorization: boolean;
2062 doctorID: string;
2063 patientSSN: string;
2064 finalScore = 0;
2065 sliderValue1 = 0;
2066 sliderValue6 = 0;
2067
2068 constructor(private questionnaireService: QuestionnaireService,
2069             public dialog: MatDialog,
2070             private route: ActivatedRoute,
2071             private router: Router) { }
2072
2073 ngOnInit(): void {
2074     this.qForm = new FormGroup({
2075         question4: new FormControl({value: ''}, [Validators.required,
2076             Validators.min(0)]),
2077     });
2078     // tslint:disable-next-line:prefer-for-of
2079     for (let i = 0; i < this.dataSource.length; ++i) {
2080         this.results.push(null);
2081     }
2082
2083     this.user = JSON.parse(localStorage.getItem('user'));
2084     this.route.params.subscribe(params => {
2085         this.doctorID = params.doctorID;
2086         this.patientSSN = params.patientSSN;
2087         this.uuid = params.uuid;
2088         this.authorization = this.user.ssn.localeCompare(params.
2089             patientSSN) === 0;
2090         if (params.back.localeCompare('questionnaires') === 0) {
2091             this.destination = 'questionnaires';
2092         } else if (params.back.localeCompare('choose-questionnaire')
2093             === 0) {
2094             this.destination = 'choose-questionnaire';
2095         }
2096     });
2097 }
2098
2099 selectResult($event: MatOptionSelectionChange, value: string,
2100             position: number, answers: string[]) {

```

```
2097 if ($event.isUserInput) {
2098   console.log(value, position, answers);
2099   if (position === 1 || position === 6 || position === 5) {
2100     if (this.results[position - 1] !== null) {
2101       let dec = 1;
2102       if (position === 5) {
2103         dec = 0;
2104       }
2105       answers.forEach(answer => {
2106         if (answer === this.results[position - 1]) {
2107           this.finalScore -= dec;
2108           console.log('Final Score After:', this.finalScore);
2109         }
2110         dec++;
2111       });
2112     }
2113     let inc = 1;
2114     if (position === 5) {
2115       inc = 0;
2116     }
2117     answers.forEach(answer => {
2118       if (answer === value) {
2119         this.finalScore += inc;
2120       }
2121       inc++;
2122     });
2123   } else if (position === 2 || position === 3) {
2124     if (this.results[position - 1] !== null) {
2125       let dec = answers.length - 1;
2126       answers.forEach(answer => {
2127         if (answer === this.results[position - 1]) {
2128           this.finalScore -= dec;
2129           console.log('Final Score After:', this.finalScore);
2130         }
2131         dec--;
2132       });
2133     }
2134     let inc = answers.length - 1;
2135     answers.forEach(answer => {
2136       if (answer === value) {
2137         this.finalScore += inc;
2138       }
2139       inc--;
2140     });
2141   }
2142   this.results[position - 1] = value;
2143
2144   console.log('Risposta: ', value);
2145   console.log('Risultati: ', this.results);
```

```

2146     console.log('Final Score: ', this.finalScore);
2147   }
2148 }
2149
2150 setResult(position: number) {
2151   const question = 'question' + position;
2152   if (position === 4) {
2153     console.log('FORM VAL', this.qForm.get(question).value);
2154     console.log('BEFORE INC VAL', this.finalScore);
2155     if (this.results[position - 1] !== null && this.qForm.get(
2156       question).value >= 0) {
2157       this.finalScore -= parseInt(this.results[position - 1], 10);
2158       this.results[position - 1] = null;
2159     }
2160     if (this.qForm.get(question).value >= 0) {
2161       this.results[position - 1] = this.qForm.get(question).value.
2162       toString();
2163       this.finalScore += this.qForm.get(question).value;
2164       console.log('After inc finalScore: ', this.finalScore);
2165     }
2166   }
2167   if (position === 1) {
2168     if (this.results[position - 1] !== null) {
2169       this.finalScore -= parseInt(this.results[position - 1], 10);
2170       this.results[position - 1] = null;
2171     }
2172     console.log(this.sliderValue1);
2173     this.results[position - 1] = this.sliderValue1;
2174     this.finalScore += this.sliderValue1;
2175     console.log('After inc finalScore: ', this.finalScore);
2176   } else if (position === 6) {
2177     if (this.results[position - 1] !== null) {
2178       this.finalScore -= parseInt(this.results[position - 1], 10);
2179       this.results[position - 1] = null;
2180     }
2181     console.log(this.sliderValue6);
2182     this.results[position - 1] = this.sliderValue6;
2183     this.finalScore += this.sliderValue6;
2184     console.log('After inc finalScore: ', this.finalScore);
2185   }
2186 }
2187
2188 onSubmit() {
2189   this.submitted = true;
2190   this.show = true;
2191   this.found = false;
2192
2193   if (this.results.length !== this.dataSource.length) {
2194     this.show = false;

```

```

2193     this.submitted = false;
2194     this.found = true;
2195     const msg = 'Non hai finito di completare il questionario!';
2196     this.openErrorDialog(msg);
2197   } else {
2198     for (const r of this.results) {
2199       if (r === null) {
2200         this.show = false;
2201         this.submitted = false;
2202         this.found = true;
2203         const msg = 'Non hai finito di completare il questionario
!';
2204         this.openErrorDialog(msg);
2205         break;
2206       }
2207     }
2208
2209     if (!this.found) {
2210
2211       console.log(this.finalScore);
2212       const resultToSend = new MIAHCDResults(this.doctorID, this.
patientSSN, this.results, this.finalScore);
2213       this.questionnaireService.sendForm(resultToSend, 'miah-cd',
this.uuid);
2214     }
2215   }
2216 }
2217
2218 openErrorDialog(message: string): void {
2219   const dialogRef = this.dialog.open(ErrorDialogComponent, {
2220     width: '500px',
2221     height: '270px',
2222     data: message
2223   });
2224
2225   dialogRef.afterClosed().subscribe(result => {
2226     console.log('Finestra chiusa');
2227   });
2228 }
2229
2230 goHome() {
2231   if (this.user === null) {
2232     this.router.navigateByUrl('/startup');
2233   } else {
2234     this.router.navigateByUrl('/landing');
2235   }
2236 }
2237
2238 }

```

```

2239
2240
2241 import { Component, OnInit } from '@angular/core';
2242 import { FormControl, FormGroup, Validators } from '@angular/forms';
2243 import { QuestionnaireService } from '../../../services/
      questionnaire.service';
2244 import { MatDialog } from '@angular/material/dialog';
2245 import { ActivatedRoute, Router } from '@angular/router';
2246 import { MatOptionSelectionChange } from '@angular/material/core';
2247 import { ErrorDialogComponent } from '../../../user-actions/error-
      dialog/error-dialog.component';
2248 import { MIAHUCResults, Q_MIAH_UC } from '../../../model/
      questionnaires/questionnaire-miah-uc';
2249
2250 @Component({
2251   selector: 'app-questionnaire-miah-uc',
2252   templateUrl: './questionnaire-miah-uc.component.html',
2253   styleUrls: ['./questionnaire-miah-uc.component.css']
2254 })
2255 export class QuestionnaireMiahUcComponent implements OnInit {
2256
2257   qForm: FormGroup;
2258   dataSource = Q_MIAH_UC;
2259   notFocused = false;
2260   submitted = false;
2261   show = false;
2262   displayedColumns = ['position', 'question', 'value'];
2263   results = [];
2264   q = 'question';
2265   found = false;
2266
2267   /// toolbar
2268   back = true;
2269   destination = '/questionnaires';
2270   title = 'MIAH-UC';
2271   user = null;
2272   uuid = null;
2273   authorization: boolean;
2274   doctorID: string;
2275   patientSSN: string;
2276   finalScore = 0;
2277   sliderValue1 = 0;
2278   sliderValue5 = 0;
2279
2280   constructor(private questionnaireService: QuestionnaireService,
2281               public dialog: MatDialog,
2282               private route: ActivatedRoute,
2283               private router: Router) { }
2284

```

```

2285 ngOnInit(): void {
2286     this.qForm = new FormGroup({
2287         question3: new FormControl({ value: '' }, [Validators.required,
Validators.min(0)]),
2288     });
2289     // tslint:disable-next-line:prefer-for-of
2290     for (let i = 0; i < this.dataSource.length; ++i) {
2291         this.results.push(null);
2292     }
2293
2294     this.user = JSON.parse(localStorage.getItem('user'));
2295     this.route.params.subscribe(params => {
2296         this.doctorID = params.doctorID;
2297         this.patientSSN = params.patientSSN;
2298         this.uuid = params.uuid;
2299         this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
2300         if (params.back.localeCompare('questionnaires') === 0) {
2301             this.destination = 'questionnaires';
2302         } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
2303             this.destination = 'choose-questionnaire';
2304         }
2305     });
2306 }
2307
2308 selectResult($event: MatOptionSelectionChange, value: string,
position: number, answers: string[]) {
2309     if ($event.isUserInput) {
2310         console.log(value, position, answers);
2311         if (position === 1 || position === 4 || position === 5) {
2312             if (this.results[position - 1] !== null) {
2313                 let dec = 1;
2314                 if (position === 4) {
2315                     dec = 0;
2316                 }
2317                 answers.forEach(answer => {
2318                     if (answer === this.results[position - 1]) {
2319                         this.finalScore -= dec;
2320                         console.log('Final Score After:', this.finalScore);
2321                     }
2322                     dec++;
2323                 });
2324             }
2325             let inc = 1;
2326             if (position === 4) {
2327                 inc = 0;
2328             }
2329             answers.forEach(answer => {

```

```

2330         if (answer === value) {
2331             this.finalScore += inc;
2332         }
2333         inc++;
2334     });
2335 } else if (position === 2) {
2336     if (this.results[position - 1] !== null) {
2337         let dec = answers.length - 1;
2338         answers.forEach(answer => {
2339             if (answer === this.results[position - 1]) {
2340                 this.finalScore -= dec;
2341                 console.log('Final Score After:', this.finalScore);
2342             }
2343             dec--;
2344         });
2345     }
2346     let inc = answers.length - 1;
2347     answers.forEach(answer => {
2348         if (answer === value) {
2349             this.finalScore += inc;
2350         }
2351         inc--;
2352     });
2353 }
2354 this.results[position - 1] = value;
2355
2356 console.log('Risposta: ', value);
2357 console.log('Risultati: ', this.results);
2358 console.log('Final Score: ', this.finalScore);
2359 }
2360 }
2361
2362 setResult(position: number) {
2363     const question = 'question' + position;
2364     if (position === 3) {
2365         console.log('FORM VAL', this.qForm.get(question).value);
2366         console.log('BEFORE INC VAL', this.finalScore);
2367         if (this.results[position - 1] !== null && this.qForm.get(
2368             question).value >= 0) {
2369             this.finalScore -= parseInt(this.results[position - 1], 10);
2370             this.results[position - 1] = null;
2371         }
2372         if (this.qForm.get(question).value >= 0) {
2373             this.results[position - 1] = this.qForm.get(question).value.
2374                 toString();
2375             this.finalScore += this.qForm.get(question).value;
2376             console.log('After inc finalScore: ', this.finalScore);
2377         }
2378     }
2379 }

```

```

2377
2378     if (position === 1) {
2379         if (this.results[position - 1] !== null) {
2380             this.finalScore += parseInt(this.results[position - 1], 10);
2381             this.results[position - 1] = null;
2382         }
2383         console.log(this.sliderValue1);
2384         this.results[position - 1] = this.sliderValue1;
2385         this.finalScore += this.sliderValue1;
2386         console.log('After inc finalScore: ', this.finalScore);
2387     } else if (position === 5) {
2388         if (this.results[position - 1] !== null) {
2389             this.finalScore += parseInt(this.results[position - 1], 10);
2390             this.results[position - 1] = null;
2391         }
2392         console.log(this.sliderValue5);
2393         this.results[position - 1] = this.sliderValue5;
2394         this.finalScore += this.sliderValue5;
2395         console.log('After inc finalScore: ', this.finalScore);
2396     }
2397 }
2398
2399 onSubmit() {
2400     this.submitted = true;
2401     this.show = true;
2402     this.found = false;
2403
2404     if (this.results.length !== this.dataSource.length) {
2405         this.show = false;
2406         this.submitted = false;
2407         this.found = true;
2408         const msg = 'Non hai finito di completare il questionario!';
2409         this.openErrorDialog(msg);
2410     } else {
2411         for (const r of this.results) {
2412             if (r === null) {
2413                 this.show = false;
2414                 this.submitted = false;
2415                 this.found = true;
2416                 const msg = 'Non hai finito di completare il questionario
!';
2417                 this.openErrorDialog(msg);
2418                 break;
2419             }
2420         }
2421
2422         if (!this.found) {
2423             console.log(this.finalScore);
2424

```

```

2425         const resultToSend = new MIAHUCResults(this.doctorID, this.
2426         patientSSN, this.results, this.finalScore);
2427         this.questionnaireService.sendForm(resultToSend, 'miah-uc',
2428         this.uuid);
2429     }
2430 }
2431
2432 openErrorDialog(message: string): void {
2433     const dialogRef = this.dialog.open(ErrorDialogComponent, {
2434         width: '500px',
2435         height: '270px',
2436         data: message
2437     });
2438     dialogRef.afterClosed().subscribe(result => {
2439         console.log('Finestra chiusa');
2440     });
2441 }
2442
2443 goHome() {
2444     if (this.user === null) {
2445         this.router.navigateByUrl('/startup');
2446     } else {
2447         this.router.navigateByUrl('/landing');
2448     }
2449 }
2450 }
2451 }
2452
2453
2454
2455 import { Component, OnInit } from '@angular/core';
2456 import { FormGroup } from '@angular/forms';
2457 import { QuestionnaireService } from '../services/questionnaire.service';
2458 import { Store } from '@ngrx/store';
2459 import { AppState } from '../store/app.states';
2460 import { ActivatedRoute, Router } from '@angular/router';
2461 import { MatDialog } from '@angular/material/dialog';
2462 import { MatOptionSelectionChange } from '@angular/material/core';
2463 import { ErrorDialogComponent } from '../user-actions/error-dialog/error-dialog.component';
2464 import { MMAS8Results, Q_MMAS8 } from '../model/questionnaires/questionnaire-mmas8';
2465 import { User } from '../model/user/User';
2466
2467 @Component({
2468     selector: 'app-questionnaire-mmas8',

```

```

2469   templateUrl: './questionnaire-mm8.component.html',
2470   styleUrls: ['./questionnaire-mm8.component.css']
2471 })
2472 export class QuestionnaireMmas8Component implements OnInit {
2473
2474   qForm: FormGroup;
2475   dataSource = Q_MMAS_8;
2476   notFocused = false;
2477   submitted = false;
2478   show = false;
2479   displayedColumns = ['position', 'question', 'value'];
2480   values: string[] = ['Si', 'No'];
2481   values2q: string[] = ['Mai/molto raramente', 'Raramente', 'A volte', 'Spesso', 'Sempre'];
2482   results = [];
2483
2484   /// toolbar
2485   back = true;
2486   found = false;
2487   title = 'MMAS-8';
2488   destination = '/questionnaires';
2489   user: User = null;
2490   uuid = null;
2491   authorization: boolean;
2492   doctorID: string;
2493   patientSSN: string;
2494   finalScore = 0;
2495
2496   constructor(private questionnaireService: QuestionnaireService,
2497               private store: Store<AppState>,
2498               private route: ActivatedRoute,
2499               public dialog: MatDialog,
2500               private router: Router) { }
2501
2502   ngOnInit(): void {
2503
2504     // tslint:disable-next-line:prefer-for-of
2505     for (let i = 0; i < this.dataSource.length; ++i) {
2506       this.results.push(null);
2507     }
2508
2509     this.user = JSON.parse(localStorage.getItem('user'));
2510     this.route.params.subscribe(params => {
2511       this.doctorID = params.doctorID;
2512       this.patientSSN = params.patientSSN;
2513       this.uuid = params.uuid;
2514       this.authorization = this.user.ssn.localeCompare(params.
2515       patientSSN) === 0;
2516       if (params.back.localeCompare('questionnaires') === 0) {

```

```

2516         this.destination = 'questionnaires';
2517     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
2518         this.destination = 'choose-questionnaire';
2519     }
2520     });
2521 }
2522
2523 selectResult($event: MatOptionSelectionChange, value: string,
position: number) {
2524     if ($event.isUserInput) {
2525         if (position >= 1 && position <= 7 && position !== 5) {
2526             if (this.results[position - 1] !== null) {
2527                 let dec = 0;
2528                 this.values.forEach(answer => {
2529                     if (answer === this.results[position - 1]) {
2530                         this.finalScore -= dec;
2531                         console.log('Final Score After:', this.finalScore);
2532                     }
2533                     dec++;
2534                 });
2535             }
2536             let inc = 0;
2537             this.values.forEach(answer => {
2538                 if (answer === value) {
2539                     this.finalScore += inc;
2540                 }
2541                 inc++;
2542             });
2543         } else if (position === 5) {
2544             if (this.results[position - 1] !== null) {
2545                 let dec = this.values.length - 1;
2546                 this.values.forEach(answer => {
2547                     if (answer === this.results[position - 1]) {
2548                         this.finalScore -= dec;
2549                         console.log('Final Score After:', this.finalScore);
2550                     }
2551                     dec--;
2552                 });
2553             }
2554             let inc = this.values.length - 1;
2555             this.values.forEach(answer => {
2556                 if (answer === value) {
2557                     this.finalScore += inc;
2558                 }
2559                 inc--;
2560             });
2561         } else if (position === 8) {
2562             if (this.results[position - 1] !== null) {

```

```

2563         let dec = this.values2q.length - 1;
2564         console.log('Dec', dec);
2565         this.values2q.forEach(answer => {
2566             if (answer === this.results[position - 1]) {
2567                 this.finalScore -= dec;
2568                 console.log('Final Score After:', this.finalScore);
2569             }
2570             dec--;
2571         });
2572     }
2573     let inc = this.values2q.length - 1;
2574     console.log('Inc', inc);
2575     this.values2q.forEach(answer => {
2576         console.log('answer', answer);
2577         console.log('value', value);
2578         if (answer === value) {
2579             this.finalScore += inc;
2580         }
2581         inc--;
2582     });
2583 }
2584 console.log('Final Score Value set:', this.finalScore);
2585 this.results[position - 1] = value;
2586 console.log(this.results);
2587 }
2588 }
2589
2590 onSubmit() {
2591     this.submitted = true;
2592     this.show = true;
2593     this.found = false;
2594
2595     if (this.results.length !== this.dataSource.length) {
2596         this.show = false;
2597         this.submitted = false;
2598         this.found = true;
2599         const msg = 'Non hai finito di completare il questionario!';
2600         this.openErrorDialog(msg);
2601     } else {
2602         for (const r of this.results) {
2603             if (r === null) {
2604                 this.show = false;
2605                 this.submitted = false;
2606                 this.found = true;
2607                 const msg = 'Non hai finito di completare il questionario!';
2608                 this.openErrorDialog(msg);
2609                 break;
2610             }

```

```

2611     }
2612
2613     if (!this.found) {
2614         console.log(this.finalScore);
2615         console.log(this.patientSSN);
2616         const resultToSend = new MMAS8Results(this.doctorID, this.
patientSSN, this.results, this.finalScore);
2617         this.questionnaireService.sendForm(resultToSend, 'mmas8',
this.uuid);
2618     }
2619 }
2620 }
2621
2622 openErrorDialog(message: string): void {
2623     const dialogRef = this.dialog.open(ErrorDialogComponent, {
2624         width: '500px',
2625         height: '270px',
2626         data: message
2627     });
2628
2629     dialogRef.afterClosed().subscribe(result => {
2630         console.log('Finestra chiusa');
2631     });
2632 }
2633
2634 goHome() {
2635     if (this.user === null) {
2636         this.router.navigateByUrl('/startup');
2637     } else {
2638         this.router.navigateByUrl('/landing');
2639     }
2640 }
2641
2642 }
2643
2644
2645
2646 import { Component, OnInit } from '@angular/core';
2647 import { MatOptionSelectionChange } from '@angular/material/core';
2648 import { FormGroup } from '@angular/forms';
2649 import { PatientSccaiResults, Q_PATIENT_SCCAI } from '../.../model
/questionnaires/questionnaire-patient-sccai';
2650 import { QuestionnaireService } from '../.../services/
questionnaire.service';
2651 import { MatDialog } from '@angular/material/dialog';
2652 import { ActivatedRoute, Router } from '@angular/router';
2653 import { ErrorDialogComponent } from '../.../user-actions/error-
dialog/error-dialog.component';
2654

```

```

2655 @Component({
2656   selector: 'app-questionnaire-patient-sccai',
2657   templateUrl: './questionnaire-patient-sccai.component.html',
2658   styleUrls: ['./questionnaire-patient-sccai.component.css']
2659 })
2660 export class QuestionnairePatientSCCAIComponent implements OnInit {
2661
2662   qForm: FormGroup;
2663   dataSource = Q_PATIENT_SCCAI;
2664   notFocused = false;
2665   submitted = false;
2666   show = false;
2667   displayedColumns = ['position', 'question', 'value'];
2668   results = [];
2669   q = 'question';
2670   found = false;
2671   finalScore = 0;
2672
2673   /// toolbar
2674   back = true;
2675   destination = '/questionnaires';
2676   title = 'PATIENT-SCCAI';
2677   user = null;
2678   uuid = null;
2679   authorization: boolean;
2680   doctorID: string;
2681   patientSSN: string;
2682
2683   constructor(private questionnaireService: QuestionnaireService,
2684               public dialog: MatDialog,
2685               private route: ActivatedRoute,
2686               private router: Router) { }
2687
2688   ngOnInit(): void {
2689     // tslint:disable-next-line:prefer-for-of
2690     for (let i = 0; i < this.dataSource.length; ++i) {
2691       this.results.push(null);
2692     }
2693
2694     this.qForm = new FormGroup({});
2695
2696     this.user = JSON.parse(localStorage.getItem('user'));
2697     this.route.params.subscribe(params => {
2698       this.doctorID = params.doctorID;
2699       this.patientSSN = params.patientSSN;
2700       this.uuid = params.uuid;
2701       this.authorization = this.user.ssn.localeCompare(params.
2702 patientSSN) === 0;
2703       if (params.back.localeCompare('questionnaires') === 0) {

```

```

2703         this.destination = 'questionnaires';
2704     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
2705         this.destination = 'choose-questionnaire';
2706     }
2707     });
2708 }
2709
2710 onSubmit(value: any) {
2711     this.submitted = true;
2712     this.show = true;
2713     this.found = false;
2714
2715     if (this.results.length !== this.dataSource.length) {
2716         this.show = false;
2717         this.submitted = false;
2718         this.found = true;
2719         const msg = 'Non hai finito di completare il questionario!';
2720         this.openErrorDialog(msg);
2721     } else {
2722         for (const r of this.results) {
2723             if (r === null) {
2724                 this.show = false;
2725                 this.submitted = false;
2726                 this.found = true;
2727                 const msg = 'Non hai finito di completare il questionario
!';
2728                 this.openErrorDialog(msg);
2729                 break;
2730             }
2731         }
2732
2733         if (!this.found) {
2734             console.log(this.finalScore);
2735             const resultToSend = new PatientSecaiResults(this.doctorID,
this.patientSSN, this.results, this.finalScore);
2736             this.questionnaireService.sendForm(resultToSend, 'patient-
sccai', this.uuid);
2737         }
2738     }
2739 }
2740
2741 selectResult($event: MatOptionSelectionChange, value: string,
position: number, answers: string[]) {
2742     if ($event.isUserInput) {
2743         console.log(value, position, answers);
2744         if (position === 7) {
2745             if (this.results[position - 1] !== null) {
2746                 console.log('Precedente: ', this.results[position - 1]);

```

```

2747     console.log('Final Score Before: ', this.finalScore);
2748     if (parseInt(this.results[position - 1], 10) >= 7) {
2749         this.finalScore -= 0;
2750     } else if (parseInt(this.results[position - 1], 10) === 6)
2751     {
2752         this.finalScore -= 1;
2753     } else if (parseInt(this.results[position - 1], 10) === 5)
2754     {
2755         this.finalScore -= 2;
2756     } else if (parseInt(this.results[position - 1], 10) === 4)
2757     {
2758         this.finalScore -= 3;
2759     } else if (parseInt(this.results[position - 1], 10) < 4) {
2760         this.finalScore -= 4;
2761     }
2762     console.log('Final Score After: ', this.finalScore);
2763 }
2764 if (parseInt(value, 10) >= 7) {
2765     this.finalScore += 0;
2766 } else if (parseInt(value, 10) === 6) {
2767     this.finalScore += 1;
2768 } else if (parseInt(value, 10) === 5) {
2769     this.finalScore += 2;
2770 } else if (parseInt(value, 10) === 4) {
2771     this.finalScore += 3;
2772 } else if (parseInt(value, 10) < 4) {
2773     this.finalScore += 4;
2774 }
2775 } else {
2776     if (this.results[position - 1] !== null) {
2777         let dec = 0;
2778         answers.forEach(answer => {
2779             if (answer === this.results[position - 1]) {
2780                 this.finalScore -= dec;
2781                 console.log('Final Score After: != 7', this.finalScore)
2782             }
2783         });
2784         dec++;
2785     }
2786 }
2787 let inc = 0;
2788 answers.forEach(answer => {
2789     if (answer === value) {
2790         this.finalScore += inc;
2791     }
2792     inc++;
2793 });
2794 }

```

```

2792         this.results[position - 1] = value;
2793
2794         console.log('Risposta: ', value);
2795         console.log('Risultati: ', this.results);
2796         console.log('Final Score: ', this.finalScore);
2797     }
2798 }
2799
2800
2801 openErrorDialog(message: string): void {
2802     const dialogRef = this.dialog.open(ErrorDialogComponent, {
2803         width: '500px',
2804         height: '270px',
2805         data: message
2806     });
2807
2808     dialogRef.afterClosed().subscribe(result => {
2809         console.log('Finestra chiusa');
2810     });
2811 }
2812
2813 goHome() {
2814     if (this.user === null) {
2815         this.router.navigateByUrl('/startup');
2816     } else {
2817         this.router.navigateByUrl('/landing');
2818     }
2819 }
2820 }
2821
2822
2823 import { Component, OnInit } from '@angular/core';
2824 import { FormGroup } from '@angular/forms';
2825 import { QuestionnaireService } from '../services/questionnaire.service';
2826 import { Store } from '@ngrx/store';
2827 import { AppState } from '../store/app.states';
2828 import { ActivatedRoute, Router } from '@angular/router';
2829 import { MatDialog } from '@angular/material/dialog';
2830 import { MatOptionSelectionChange } from '@angular/material/core';
2831 import { ErrorDialogComponent } from '../user-actions/error-dialog/error-dialog.component';
2832 import { PHQ9Results, Q_PHQ_9 } from '../model/questionnaires/questionnaire-phq9';
2833
2834 @Component({
2835     selector: 'app-questionnaire-phq9',
2836     templateUrl: './questionnaire-phq9.component.html',
2837     styleUrls: ['./questionnaire-phq9.component.css']

```

```

2838 })
2839 export class QuestionnairePhq9Component implements OnInit {
2840
2841   qForm: FormGroup;
2842   dataSource = Q_PHQ_9;
2843   notFocused = false;
2844   submitted = false;
2845   show = false;
2846   displayedColumns = ['position', 'question', 'value'];
2847   values: string[] = ['Mai', 'Alcuni giorni', 'Più di metà dei giorni',
2848     'Quasi tutti i giorni'];
2849   values2q: string[] = ['Per niente difficile', 'Abbastanza difficile',
2850     'Molto difficile', 'Estremamente difficile'];
2851   results = [];
2852
2853   /// toolbar
2854   back = true;
2855   found = false;
2856   title = 'PHQ-9';
2857   destination = '/questionnaires';
2858   user = null;
2859   uuid = null;
2860   authorization: boolean;
2861   doctorID: string;
2862   patientSSN: string;
2863   finalScore = 0;
2864
2865   constructor(private questionnaireService: QuestionnaireService,
2866     private store: Store<AppState>,
2867     private route: ActivatedRoute,
2868     public dialog: MatDialog,
2869     private router: Router) { }
2870
2871   ngOnInit(): void {
2872     // tslint:disable-next-line:prefer-for-of
2873     for (let i = 0; i < this.dataSource.length - 1; ++i) {
2874       this.results.push(null);
2875     }
2876
2877     this.user = JSON.parse(localStorage.getItem('user'));
2878     this.route.params.subscribe(params => {
2879       this.doctorID = params.doctorID;
2880       this.patientSSN = params.patientSSN;
2881       this.uuid = params.uuid;
2882       this.authorization = this.user.ssn.localeCompare(params.patientSSN) === 0;
2883       if (params.back.localeCompare('questionnaires') === 0) {
2884         this.destination = 'questionnaires';
2885       }
2886     });
2887   }
2888 }

```

```

2884     } else if (params.back.localeCompare('choose-questionnaire')
2885     == 0) {
2886         this.destination = 'choose-questionnaire';
2887     }
2888 }));
2889 }
2890
2891 selectResult($event: MatOptionSelectionChange, value: string,
2892 position: number) {
2893     if ($event.isUserInput) {
2894         if (position > 0 && position <= 9) {
2895             if (this.results[position - 1] !== null) {
2896                 let dec = 0;
2897                 this.values.forEach(answer => {
2898                     if (answer === this.results[position - 1]) {
2899                         this.finalScore -= dec;
2900                         console.log('Final Score After:', this.finalScore);
2901                     }
2902                     dec++;
2903                 });
2904             }
2905             let inc = 0;
2906             this.values.forEach(answer => {
2907                 if (answer === value) {
2908                     this.finalScore += inc;
2909                 }
2910                 inc++;
2911             });
2912             console.log('Final Score Value set:', this.finalScore);
2913             this.results[position - 1] = value;
2914         }
2915     }
2916
2917 onSubmit() {
2918     this.submitted = true;
2919     this.show = true;
2920     this.found = false;
2921     console.log(this.results);
2922
2923     if (this.results.length !== this.dataSource.length - 1) {
2924         this.show = false;
2925         this.submitted = false;
2926         this.found = true;
2927         const msg = 'Non hai finito di completare il questionario!';
2928         this.openErrorDialog(msg);
2929     } else {
2930         for (const r of this.results) {
2931             if (r === null) {

```

```

2931         this.show = false;
2932         this.submitted = false;
2933         this.found = true;
2934         const msg = 'Non hai finito di completare il questionario
!';
2935         this.openErrorDialog(msg);
2936         break;
2937     }
2938 }
2939
2940     if (!this.found) {
2941         console.log(this.finalScore);
2942         const resultToSend = new PHQ9Results(this.doctorID, this.
patientSSN, this.results, this.finalScore);
2943         this.questionnaireService.sendForm(resultToSend, 'phq9', this
.uuid);
2944     }
2945 }
2946 }
2947
2948 openErrorDialog(message: string): void {
2949     const dialogRef = this.dialog.open(ErrorDialogComponent, {
2950         width: '500px',
2951         height: '270px',
2952         data: message
2953     });
2954
2955     dialogRef.afterClosed().subscribe(result => {
2956         console.log('Finestra chiusa');
2957     });
2958 }
2959
2960 goHome() {
2961     if (this.user === null) {
2962         this.router.navigateByUrl('/startup');
2963     } else {
2964         this.router.navigateByUrl('/landing');
2965     }
2966 }
2967
2968 }
2969
2970
2971
2972 import { AfterViewInit, Component, OnInit } from '@angular/core';
2973 import { QuestionnaireService } from '../services/
questionnaire.service';
2974 import { MatDialog } from '@angular/material/dialog';
2975 import { ActivatedRoute, Router } from '@angular/router';

```

```

2976 import {fabric} from 'fabric';
2977 import {PrismObject, PRISMResult} from '../../../../../model/
      questionnaires/questionnaire-prism';
2978
2979 @Component({
2980   selector: 'app-questionnaire-prism',
2981   templateUrl: './questionnaire-prism.component.html',
2982   styleUrls: ['./questionnaire-prism.component.css']
2983 })
2984 export class QuestionnairePrismComponent implements OnInit,
      AfterViewInit {
2985
2986   notFocused = false;
2987   submitted = false;
2988   show = false;
2989   displayedColumns = ['position', 'question', 'value'];
2990   results = [];
2991   found = false;
2992   finalScore = 0;
2993
2994   /// toolbar
2995   back = true;
2996   destination = '/questionnaires';
2997   title = 'PRISM';
2998   user = null;
2999   uuid = null;
3000   authorization: boolean;
3001   doctorID: string;
3002   patientSSN: string;
3003   canvas: any;
3004   self: PrismObject;
3005   disease: PrismObject;
3006   distance: number;
3007
3008   constructor(private questionnaireService: QuestionnaireService,
3009               public dialog: MatDialog,
3010               private route: ActivatedRoute,
3011               private router: Router) {}
3012
3013   ngOnInit(): void {
3014     this.user = JSON.parse(localStorage.getItem('user'));
3015     this.route.params.subscribe(params => {
3016       this.doctorID = params.doctorID;
3017       this.patientSSN = params.patientSSN;
3018       this.uuid = params.uuid;
3019       this.authorization = this.user.ssn.localeCompare(params.
3020       patientSSN) === 0;
3021       if (params.back.localeCompare('questionnaires') === 0) {
3022         this.destination = 'questionnaires';

```

```

3022     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
3023         this.destination = 'choose-questionnaire';
3024     }
3025 });
3026
3027     this.self = new PrismObject(500, 200);
3028     this.disease = new PrismObject(40, 40);
3029     // tslint:disable-next-line:max-line-length
3030     this.distance = parseFloat((Math.hypot(this.self.x - this.disease
.x,
3031     this.self.y - this.disease.y) * 0.026458).toFixed(1));
3032
3033     console.log('start distance: ', this.distance);
3034 }
3035
3036 ngAfterViewInit(): void {
3037     const canvas = new fabric.Canvas('canvas');
3038     const disease = new fabric.Circle({
3039         radius: 30, left: 40, top: 40, fill: '#cc0000',
3040         originX: 'center', originY: 'center',
3041         hasControls: false, hasBorders: false, hoverCursor: 'pointer',
moveCursor: 'pointer',
3042         stroke: '#990000', strokeWidth: 2
3043     });
3044
3045     const self = new fabric.Circle({
3046         radius: 50, left: 500, top: 200, fill: '#ffe014', hasControls:
false, hasBorders: false,
3047         originX: 'center', originY: 'center',
3048         evented: false, stroke: '#ffc414', strokeWidth: 2
3049     });
3050
3051     canvas.add(self, disease);
3052     canvas.selection = false;
3053     canvas.on({
3054         'object:moving': e => {
3055             const circle = canvas.getActiveObject();
3056             const x1 = circle.left;
3057             const y1 = circle.top;
3058             const x2 = 500;
3059             const y2 = 200;
3060             const distance = parseFloat((Math.hypot(x2 - x1, y2 - y1) *
0.026458).toFixed(1));
3061             console.log('Distance in cm: ', distance);
3062             const selfObj = new PrismObject(x2, y2);
3063             const diseaseObj = new PrismObject(x1, y1);
3064             this.self = selfObj;
3065             this.disease = diseaseObj;

```

```

3066         this.distance = distance;
3067         console.log('log', this.self, this.disease, this.distance);
3068     }
3069     });
3070 }
3071
3072 goHome() {
3073     if (this.user === null) {
3074         this.router.navigateByUrl('/startup');
3075     } else {
3076         this.router.navigateByUrl('/landing');
3077     }
3078 }
3079
3080 onSubmit() {
3081     this.submitted = true;
3082     this.show = true;
3083     this.found = false;
3084     const resultToSend = new PRISMResult(this.doctorID, this.
3085 patientSSN, this.self, this.disease, this.distance.toFixed(2));
3086     this.questionnaireService.sendForm(resultToSend, 'prism', this.
3087 uuid);
3088 }
3089
3090 import { Component, OnInit } from '@angular/core';
3091 import { FormControl, FormGroup, Validators } from '@angular/forms';
3092 import { QuestionnaireService } from '../services/
3093     questionnaire.service';
3094 import { MatDialog } from '@angular/material/dialog';
3095 import { ActivatedRoute, Router } from '@angular/router';
3096 import { MatOptionSelectionChange } from '@angular/material/core';
3097 import { ErrorDialogComponent } from '../user-actions/error-
3098     dialog/error-dialog.component';
3099 import { PSQIResults, Q_PSQI } from '../model/questionnaires/
3100     questionnaire-psqi';
3101
3102 @Component({
3103     selector: 'app-questionnaire-psqi',
3104     templateUrl: './questionnaire-psqi.component.html',
3105     styleUrls: ['./questionnaire-psqi.component.css']
3106 })
3107 export class QuestionnairePsqiComponent implements OnInit {
3108
3109     qForm: FormGroup;
3110     dataSource = Q_PSQI;
3111     notFocused = false;
3112     submitted = false;
3113     show = false;

```

```

3110 displayedColumns = [ 'position', 'question', 'value' ];
3111 results = [];
3112 q = 'question';
3113 found = false;
3114
3115 /// toolbar
3116 back = true;
3117 destination = '/questionnaires';
3118 title = 'PSQI';
3119 user = null;
3120 uuid = null;
3121 authorization: boolean;
3122 doctorID: string;
3123 patientSSN: string;
3124
3125 constructor(private questionnaireService: QuestionnaireService,
3126             public dialog: MatDialog,
3127             private route: ActivatedRoute,
3128             private router: Router) { }
3129
3130 ngOnInit(): void {
3131     this.qForm = new FormGroup({
3132         question1hour: new FormControl({ value: '' }, [ Validators.
3133             required, Validators.min(0), Validators.max(23) ]),
3134         question1minute: new FormControl({ value: '' }, [ Validators.
3135             required, Validators.min(0), Validators.max(59) ]),
3136         question2: new FormControl({ value: '' }, [ Validators.required,
3137             Validators.min(0) ]),
3138         question3hour: new FormControl({ value: '' }, [ Validators.
3139             required, Validators.min(0), Validators.max(23) ]),
3140         question3minute: new FormControl({ value: '' }, [ Validators.
3141             required, Validators.min(0), Validators.max(59) ]),
3142         question4: new FormControl({ value: '' }, [ Validators.required,
3143             Validators.min(0) ]),
3144         question15: new FormControl({ value: '' }),
3145     });
3146     // tslint:disable-next-line:prefer-for-of
3147     for (let i = 0; i < this.dataSource.length + 1; ++i) {
3148         this.results.push(null);
3149     }
3150
3151     this.user = JSON.parse(localStorage.getItem('user'));
3152     this.route.params.subscribe(params => {
3153         this.doctorID = params.doctorID;
3154         this.patientSSN = params.patientSSN;
3155         this.uuid = params.uuid;
3156         this.authorization = this.user.ssn.localeCompare(params.
3157             patientSSN) === 0;
3158         if (params.back.localeCompare('questionnaires') === 0) {

```

```

3152         this.destination = 'questionnaires';
3153     } else if (params.back.localeCompare('choose-questionnaire')
=== 0) {
3154         this.destination = 'choose-questionnaire';
3155     }
3156     });
3157     this.qForm.get('question15').setValue(null);
3158 }
3159
3160 selectResult($event: MatOptionSelectionChange, value: string,
position: number, answers: string[]) {
3161     if ($event.isUserInput) {
3162         this.results[position - 1] = value;
3163
3164         if (position === 14 && this.results[position - 1] === 'No') {
3165             this.results[position] = null;
3166             this.results[position + 1] = null;
3167             console.log('Setteeddd');
3168         }
3169
3170         console.log('Risposta: ', value);
3171         console.log('Risultati: ', this.results);
3172     }
3173 }
3174
3175 setResult(position: number) {
3176     const question = 'question' + position;
3177     // tslint:disable-next-line:max-line-length
3178     if ((position === 1 || position === 3) && (this.qForm.get(
question + 'hour').value >= 0 && this.qForm.get(question + 'minute
').value >= 0)) {
3179
3180         let res = '';
3181         if (this.qForm.get(question + 'hour').value < 10) {
3182             res += '0';
3183         }
3184         res += this.qForm.get(question + 'hour').value.toString() +
':';
3185         if (this.qForm.get(question + 'minute').value < 10) {
3186             res += '0';
3187         }
3188         res += this.qForm.get(question + 'minute').value.toString();
3189
3190         this.results[position - 1] = res;
3191
3192     } else if (position === 2 || position === 4) {
3193         this.results[position - 1] = this.qForm.get(question).value.
toString();
3194         console.log('Ecco');

```

```

3195     } else if (position === 14 && this.results[position - 1] === 'Si
3196     (specificare)') {
3197         if (this.qForm.get('question15').value !== '' && this.qForm.get
3198         ('question15').value !== null) {
3199             this.results[position] = this.qForm.get('question15').value.
3200             toString();
3201         } else {
3202             this.results[position] = null;
3203         }
3204     }
3205     console.log(this.results);
3206 }
3207
3208 onSubmit() {
3209     this.submitted = true;
3210     this.show = true;
3211     this.found = false;
3212
3213     if (this.results.length !== this.dataSource.length + 1) {
3214         this.show = false;
3215         this.submitted = false;
3216         this.found = true;
3217         const msg = 'Non hai finito di completare il questionario!';
3218         this.openAlertDialog(msg);
3219     } else {
3220         for (let i = 0; i < this.results.length; i++) {
3221             if (i !== 14 && i !== 15) {
3222                 if (this.results[i] === null) {
3223                     this.show = false;
3224                     this.submitted = false;
3225                     this.found = true;
3226                     const msg = 'Non hai finito di completare il
3227                     questionario!';
3228                     this.openAlertDialog(msg);
3229                     break;
3230                 }
3231             }
3232             if (i === 13) {
3233                 if (this.results[i] !== null && this.results[i] !== 'No')
3234                 {
3235                     if (this.results[i + 1] === null || this.results[i + 2]
3236                     === null) {
3237                         this.show = false;
3238                         this.submitted = false;
3239                         this.found = true;
3240                         const msg = 'Non hai finito di completare il
3241                         questionario!';
3242                         this.openAlertDialog(msg);

```

```

3237         break;
3238     }
3239 }
3240 }
3241 }
3242
3243     if (!this.found) {
3244         const resultToSend = new PSQIResults(this.doctorID, this.
patientSSN, this.results);
3245         this.questionnaireService.sendForm(resultToSend, 'psqi',
this.uuid);
3246     }
3247 }
3248 }
3249
3250 openErrorDialog(message: string): void {
3251     const dialogRef = this.dialog.open(ErrorDialogComponent, {
3252         width: '500px',
3253         height: '270px',
3254         data: message
3255     });
3256
3257     dialogRef.afterClosed().subscribe(result => {
3258         console.log('Finestra chiusa');
3259     });
3260 }
3261
3262 goHome() {
3263     if (this.user === null) {
3264         this.router.navigateByUrl('/startup');
3265     } else {
3266         this.router.navigateByUrl('/landing');
3267     }
3268 }
3269 }
3270 }
3271
3272
3273
3274 import {Component, OnInit, Renderer2} from '@angular/core';
3275 import {FormGroup} from '@angular/forms';
3276 import {QuestionnaireService} from '../services/questionnaire.service';
3277 import {MatDialog} from '@angular/material/dialog';
3278 import {ErrorDialogComponent} from '../user-actions/error-dialog/error-dialog.component';
3279 import {MatRadioButton} from '@angular/material/radio';
3280 import {Q_SIBDQ, SIBDQResults} from '../model/questionnaires/questionnaire-sibdq';

```

```

3281 import { ActivatedRoute, Router } from '@angular/router';
3282
3283 @Component({
3284   selector: 'app-questionnaire-sibdq',
3285   templateUrl: './questionnaire-sibdq.component.html',
3286   styleUrls: ['./questionnaire-sibdq.component.css']
3287 })
3288 export class QuestionnaireSibdqComponent implements OnInit {
3289
3290   qForm: FormGroup;
3291   notFocused = false;
3292   source = Q_SIBDQ;
3293   results = [];
3294   submitted = false;
3295   show = false;
3296   currentCheckedValue = [null];
3297   found = false;
3298
3299   /// toolbar
3300   back = true;
3301   destination = '/questionnaires';
3302   title = 'SIBDQ';
3303   user = null;
3304   uuid = null;
3305   authorization: boolean;
3306   doctorID: string;
3307   patientSSN: string;
3308
3309   constructor(private questionnaireService: QuestionnaireService,
3310               private ren: Renderer2,
3311               public dialog: MatDialog,
3312               private route: ActivatedRoute,
3313               private router: Router) { }
3314
3315   ngOnInit(): void {
3316     // tslint:disable-next-line:prefer-for-of
3317     for (let i = 0; i < this.source.length; ++i) {
3318       this.results.push(null);
3319     }
3320
3321     this.user = JSON.parse(localStorage.getItem('user'));
3322     this.route.params.subscribe(params => {
3323       this.doctorID = params.doctorID;
3324       this.patientSSN = params.patientSSN;
3325       this.uuid = params.uuid;
3326       this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
3327       if (params.back.localeCompare('questionnaires') === 0) {
3328         this.destination = 'questionnaires';

```

```

3329         } else if (params.back.localeCompare('choose-questionnaire')
3330         === 0) {
3331             this.destination = 'choose-questionnaire';
3332         }
3333     });
3334 }
3335
3336 openDialog(message: string): void {
3337     const dialogRef = this.dialog.open(ErrorDialogComponent, {
3338         width: '500px',
3339         height: '270px',
3340         data: message
3341     });
3342
3343     dialogRef.afterClosed().subscribe(result => {
3344         console.log('Finestra chiusa');
3345     });
3346 }
3347
3348 setAnswer(button: MatRadioButton, answer: string, position: number)
3349 {
3350     this.results[position - 1] = answer;
3351     console.log(this.results);
3352 }
3353
3354 checkState(el, position: number) {
3355     setTimeout(() => {
3356         if (this.currentCheckedValue[position - 1] && this.
3357         currentCheckedValue[position - 1] === el.value) {
3358             el.checked = false;
3359             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
3360             focused');
3361             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
3362             program-focused');
3363             this.currentCheckedValue[position - 1] = null;
3364             this.results[position - 1] = null;
3365         } else {
3366             this.currentCheckedValue[position - 1] = el.value;
3367         }
3368     });
3369 }
3370
3371 onSubmit() {
3372     this.submitted = true;
3373     this.show = true;
3374     this.found = false;
3375
3376     console.log(this.results);

```

```

3373     if (this.results.length !== this.source.length) {
3374         const msg = 'Non hai finito di completare il questionario!';
3375         this.openDialog(msg);
3376         this.show = false;
3377         this.submitted = false;
3378     } else {
3379         for (const r of this.results) {
3380             if (r === null) {
3381                 const msg = 'Non hai finito di completare il questionario
!';
3382                 this.openDialog(msg);
3383                 this.show = false;
3384                 this.submitted = false;
3385                 this.found = true;
3386                 break;
3387             }
3388         }
3389
3390         if (!this.found) {
3391             const resultToSend = new SIBDQResults(this.doctorID, this.
patientSSN, this.results);
3392             this.questionnaireService.sendForm(resultToSend, 'sibdq',
this.uuid);
3393         }
3394     }
3395 }
3396
3397 goHome() {
3398     if (this.user === null) {
3399         this.router.navigateByUrl('/startup');
3400     } else {
3401         this.router.navigateByUrl('/landing');
3402     }
3403 }
3404 }
3405
3406
3407
3408 import {Component, OnInit, Renderer2} from '@angular/core';
3409 import {FormGroup} from '@angular/forms';
3410 import {Q_TSQM, TSQMResults} from '../../model/questionnaires/
questionnaire-tsqm';
3411 import {ErrorDialogComponent} from '../../user-actions/error-
dialog/error-dialog.component';
3412 import {QuestionnaireService} from '../../services/
questionnaire.service';
3413 import {MatDialog} from '@angular/material/dialog';
3414 import {MatRadioButton} from '@angular/material/radio';
3415 import {ActivatedRoute, Router} from '@angular/router';

```

```

3416
3417 @Component({
3418   selector: 'app-questionnaire-tsqm',
3419   templateUrl: './questionnaire-tsqm.component.html',
3420   styleUrls: ['./questionnaire-tsqm.component.css']
3421 })
3422 export class QuestionnaireTsqmComponent implements OnInit {
3423
3424   qForm: FormGroup;
3425   notFocused = false;
3426   source = Q_TSQM;
3427   results = [];
3428   submitted = false;
3429   show = false;
3430   noDisabledButtons = true;
3431   currentCheckedValue = [null];
3432   found = false;
3433
3434   /// toolbar
3435   back = true;
3436   destination = '/questionnaires';
3437   title = 'TSQM';
3438   user = null;
3439   uuid = null;
3440   authorization: boolean;
3441   doctorID: string;
3442   patientSSN: string;
3443
3444   constructor(private questionnaireService: QuestionnaireService,
3445               private ren: Renderer2,
3446               public dialog: MatDialog,
3447               private route: ActivatedRoute,
3448               private router: Router) { }
3449
3450   ngOnInit(): void {
3451     // tslint:disable-next-line:prefer-for-of
3452     for (let i = 0; i < this.source.length; ++i) {
3453       this.results.push(null);
3454     }
3455
3456     this.user = JSON.parse(localStorage.getItem('user'));
3457     this.route.params.subscribe(params => {
3458       this.doctorID = params.doctorID;
3459       this.patientSSN = params.patientSSN;
3460       this.uuid = params.uuid;
3461       this.authorization = this.user.ssn.localeCompare(params.
patientSSN) === 0;
3462       if (params.back.localeCompare('questionnaires') === 0) {
3463         this.destination = 'questionnaires';

```

```

3464     } else if (params.back.localeCompare('choose-questionnaire')
3465     === 0) {
3466         this.destination = 'choose-questionnaire';
3467     }
3468 }
3469
3470 openFileDialog(message: string): void {
3471     const dialogRef = this.dialog.open(ErrorDialogComponent, {
3472         width: '500px',
3473         height: '270px',
3474         data: message
3475     });
3476
3477     dialogRef.afterClosed().subscribe(result => {
3478         console.log('Finestra chiusa');
3479     });
3480 }
3481
3482 setAnswer(button: MatRadioButton, answer: string, position: number)
3483 {
3484     if (position === 4) {
3485         if (answer === 'Si') {
3486             this.noDisabledButtons = true;
3487         } else if (answer === 'No (in questo caso, passi alla domanda n
3488         .9)') {
3489             this.noDisabledButtons = false;
3490             for (let i = position; i < 8; ++i) {
3491                 this.results[i] = null;
3492             }
3493         }
3494     }
3495
3496     this.results[position - 1] = answer;
3497     console.log(this.results);
3498 }
3499
3500 checkState(el, position: number) {
3501     setTimeout(() => {
3502         if (this.currentCheckedValue[position - 1] && this.
3503         currentCheckedValue[position - 1] === el.value) {
3504             el.checked = false;
3505             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
3506             focused');
3507             this.ren.removeClass(el._elementRef.nativeElement, 'cdk-
3508             program-focused');
3509             this.currentCheckedValue[position - 1] = null;
3510             this.results[position - 1] = null;
3511         } else {

```

```

3507         this.currentCheckedValue[position - 1] = el.value;
3508     }
3509     });
3510 }
3511
3512 onSubmit() {
3513     this.submitted = true;
3514     this.show = true;
3515     this.found = false;
3516
3517     let count = 0;
3518
3519     if (this.results[3] === 'No (in questo caso, passi alla domanda n
3520     .9)') {
3521         for (const r of this.results) {
3522             if (r === null) {
3523                 count++;
3524             }
3525         }
3526
3527         if (count > 4 || count < 4) {
3528             this.submitted = false;
3529             this.show = false;
3530             this.found = true;
3531             const msg = 'Non hai finito di completare il questionario!';
3532             this.openDialog(msg);
3533         } else {
3534             this.found = false; // può essere inviato
3535         }
3536     } else {
3537         for (const r of this.results) {
3538             if (r === null) {
3539                 this.found = true;
3540                 break;
3541             }
3542         }
3543         if (this.found) {
3544             this.submitted = false;
3545             this.show = false;
3546             const msg = 'Non hai finito di completare il questionario!';
3547             this.openDialog(msg);
3548         }
3549     }
3550
3551     if (!this.found) {
3552         console.log(this.doctorID + ' ' + this.patientSSN + ' ' + this.
results);

```

```

3553     const resultToSend = new TSQMResults(this.doctorID, this.
patientSSN, this.results);
3554     this.questionnaireService.sendForm(resultToSend, 'tsqm', this.
uuid);
3555   }
3556
3557   }
3558
3559   goHome() {
3560     if (this.user === null) {
3561       this.router.navigateByUrl('/startup');
3562     } else {
3563       this.router.navigateByUrl('/landing');
3564     }
3565   }
3566 }
3567
3568
3569
3570 import {Component, OnInit} from '@angular/core';
3571 import {FormControl, FormGroup, Validators} from '@angular/forms';
3572 import {Q_WPAI, WpaiResults} from '../../../../../model/questionnaires/
questionnaire-wpai';
3573 import {QuestionnaireService} from '../../../../../services/
questionnaire.service';
3574 import {MatDialog} from '@angular/material/dialog';
3575 import {MatOptionSelectionChange} from '@angular/material/core';
3576 import {ErrorDialogComponent} from '../../../../../user-actions/error-
dialog/error-dialog.component';
3577 import {ActivatedRoute, Router} from '@angular/router';
3578
3579 @Component({
3580   selector: 'app-questionnaire-wpai',
3581   templateUrl: './questionnaire-wpai.component.html',
3582   styleUrls: ['./questionnaire-wpai.component.css']
3583 })
3584 export class QuestionnaireWpaiComponent implements OnInit {
3585
3586   qForm: FormGroup;
3587   dataSource = Q_WPAI;
3588   notFocused = false;
3589   submitted = false;
3590   show = false;
3591   displayedColumns = ['position', 'question', 'value'];
3592   values: string[] = ['1', '2', '3', '4', '5', '6', '7', '8', '9',
'10'];
3593   results = [];
3594   answers: string[] = ['SI', 'NO'];
3595   isDisabled: boolean;

```

```

3596   isDisabled5 = false;
3597   q = 'question';
3598   found = false;
3599
3600   /// toolbar
3601   back = true;
3602   destination = '/questionnaires';
3603   title = 'WPAI';
3604   user = null;
3605   uuid = null;
3606   authorization: boolean;
3607   doctorID: string;
3608   patientSSN: string;
3609
3610   constructor(private questionnaireService: QuestionnaireService,
3611               public dialog: MatDialog,
3612               private route: ActivatedRoute,
3613               private router: Router) { }
3614
3615   ngOnInit(): void {
3616     this.qForm = new FormGroup({
3617       question2: new FormControl({ value: '' }, [Validators.required,
3618         Validators.min(0)]),
3619       question3: new FormControl({ value: '' }, [Validators.required,
3620         Validators.min(0)]),
3621       question4: new FormControl({ value: '' }, [Validators.required,
3622         Validators.min(0)])
3623     });
3624     // tslint:disable-next-line:prefer-for-of
3625     for (let i = 0; i < this.dataSource.length; ++i) {
3626       this.results.push(null);
3627     }
3628
3629     this.user = JSON.parse(localStorage.getItem('user'));
3630     this.route.params.subscribe(params => {
3631       this.doctorID = params.doctorID;
3632       this.patientSSN = params.patientSSN;
3633       this.uuid = params.uuid;
3634       this.authorization = this.user.ssn.localeCompare(params.
3635         patientSSN) === 0;
3636       if (params.back.localeCompare('questionnaires') === 0) {
3637         this.destination = 'questionnaires';
3638       } else if (params.back.localeCompare('choose-questionnaire')
3639         === 0) {
3640         this.destination = 'choose-questionnaire';
3641       }
3642     });
3643   }

```

```

3640 selectResult($event: MatOptionSelectionChange, value: string,
3641 position: number) {
3642   if ($event.isUserInput) {
3643     this.results[position - 1] = value;
3644     console.log(this.results);
3645     this.isDisabled = this.results[0] === 'NO';
3646     console.log(this.isDisabled);
3647
3648     if (this.results[0] === 'NO') {
3649       this.qForm.get('question2').reset();
3650       this.qForm.get('question2').disable();
3651       this.qForm.get('question3').reset();
3652       this.qForm.get('question3').disable();
3653       this.qForm.get('question4').reset();
3654       this.qForm.get('question4').disable();
3655       this.qForm.get('question5').reset();
3656       this.qForm.get('question5').disable();
3657
3658       for (let i = 1; i < 5; ++i) {
3659         this.results[i] = null;
3660       }
3661       console.log(this.results);
3662     } else {
3663       this.qForm.get('question2').enable();
3664       this.qForm.get('question3').enable();
3665       this.qForm.get('question4').enable();
3666       this.qForm.get('question5').enable();
3667
3668       console.log(this.results);
3669     }
3670   }
3671 }
3672
3673
3674 setDisabled5(position: number) {
3675   if (this.qForm.get('question4').value === 0) {
3676     this.isDisabled5 = true;
3677     this.setResult(position);
3678   } else {
3679     this.isDisabled5 = false;
3680     this.setResult(position);
3681   }
3682 }
3683
3684 setResult(position: number) {
3685   const question = 'question' + position;
3686   this.results[position - 1] = this.qForm.get(question).value.
3687   toString();

```

```

3687     console.log(question);
3688     console.log(this.qForm.get(question).value.toString());
3689     console.log(this.results);
3690
3691     if (position === 4) {
3692         if (this.qForm.get(question).value === 0) {
3693             this.results[position] = null;
3694         }
3695     }
3696 }
3697
3698 onSubmit() {
3699     this.submitted = true;
3700     this.show = true;
3701     this.found = false;
3702     let questionnaireComplete = true;
3703     console.log(this.results);
3704     //CASO IN CUI SELEZIONO NO ALLA 1
3705     if(this.results[0] === 'NO' && this.results[5]!==null) {
3706         const resultToSend = new WpaiResults(this.doctorID, this.
3707 patientSSN, this.results);
3708         this.questionnaireService.sendForm(resultToSend, 'wpai', this
3709 .uuid);
3710
3711     } else {
3712
3713         for (let i = 0; i < this.results.length; ++i){
3714             if (i !== 4 && this.results[i] === null) {
3715                 questionnaireComplete = false;
3716                 this.show = false;
3717                 this.submitted = false;
3718                 this.found = true;
3719                 const msg = 'Non hai finito di completare il questionario!';
3720                 this.openDialog(msg);
3721                 break;
3722             }
3723         }
3724     }
3725
3726     /*if (this.results.length !== this.dataSource.length) {
3727         this.show = false;
3728         this.submitted = false;
3729         this.found = true;
3730         const msg = 'Non hai finito di completare il questionario!';
3731         this.openDialog(msg);
3732     } else {*/
3733     if(questionnaireComplete === true){
3734         let count = 0;
3735         if (this.results[0] === 'NO') {
3736             for (const r of this.results) {

```