A Mobile Application for Milano Ventilatore Meccanico: A First Prototype

Silvia Bonfanti[1][a], Angelo Gargantini[1][b] and Luca Novelli[2][c]
[1]*University of Bergamo, Bergamo, Italy*
[2]*Pulmonary medicine Unit, ASST Papa Giovanni XXIII, Bergamo, Italy*

Keywords: Mobile Application, Milano Ventilatore Meccanico, Telemedicine, Mechanical Ventilator, Application Prototype.

Abstract: In hospitals the need to have devices connected and accessible remotely is increasing, in order to continuously monitor patients. This need also arose for mechanical ventilators. In this paper, we introduce the first prototype of a mobile application to connect remotely to Milano Ventilatore Meccanico, a mechanical ventilator developed during COVID-19 pandemic. We show the process adopted to design and develop the first prototype in Android.

1 INTRODUCTION

Nowadays, the use of applications to remotely control patients' health is increasing. During COVID-19 pandemic, a group of researchers has developed an open source ventilator called Milano Ventilatore Meccanico (MVM)[1]. The MVM project started from the idea of the physicist Cristiano Galbiati, who was also the leader. In only 42 days from the initial prototype production to the demonstration of performances, the FDA (Food and Drug Administration) declared that the MVM falls within the scope of the Emergency Use Authorization (EUA) for ventilators and, during the following months, it has obtained the Health Canada and the CE marking as well. Thanks to these achievements, the MVM can be sold and used in the USA, Canada, and Europe. One disadvantage of this ventilator is that the patient connected cannot be monitored and controlled remotely. This functionality has been omitted because, based on international regulations, the ventilator must guarantee a cybersecurity standard that was not possible to satisfy during its development due to time constraints related to the pandemic. Before the COVID-19 pandemic, remote control of ventilators was limited; however, changes caused by the COVID-19 crisis and the limited amount of physicians, motivated new designs of remote-control ventilators, despite system setup complexity and strict government regulation requirements (Barrow et al., 2022).

In this paper, we show how we have implemented a prototype of remote controller for the MVM. In particular, after having introduced the basic functionalities of MVM (see Sect. 2), we explain the methodology adopted in order to develop the prototype by applying an Agile approach (see Sect. 3). In Sect. 4 we have collected the existing applications related to mechanical ventilators, and in Sect. 5 we have listed the MVM mobile application requirements. The prototypes are presented in Sect. 6, while the Android prototype is shown in Sect. 7. Sect. 8 concludes the paper and we list some future improvements to the application.

2 MILANO VENTILATORE MECCANICO

MVM is an electro-mechanical ventilator and it is intended to provide pressure-regulated ventilation support for patients that are in the Intensive Care Unit (ICU). The ventilation is controlled by the pressure and both compressed oxygen and medical air are required for its operation (Bombarda et al., 2022). Before starting the ventilation the MVM controller passed through three phases: start-up in which the

[a] https://orcid.org/0000-0001-9679-4551
[b] https://orcid.org/0000-0002-4035-0131
[c] https://orcid.org/0000-0002-2705-248X
[1] http://mvm.care

Bonfanti, S., Gargantini, A. and Novelli, L.
A Mobile Application for Milano Ventilatore Meccanico: A First Prototype.
DOI: 10.5220/0011668400003414
In *Proceedings of the 16th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2023) - Volume 5: HEALTHINF*, pages 314-321
ISBN: 978-989-758-631-6; ISSN: 2184-4305

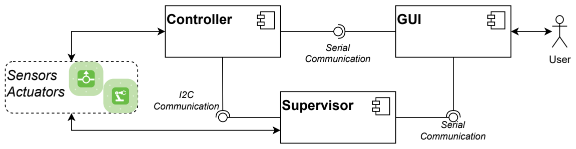


Figure 1: Current high-level software architecture.

controller is initialized with default parameters; self-test which ensures that the hardware is fully functional; and ventilation off in which the controller is ready for ventilation when requested. MVM provides two types of ventilation: Pressure Controlled Ventilation (PCV) and Pressure Support Ventilation (PSV). PCV is used when the patient is not able to start breathing on his own. The respiratory cycle is set by the physician and kept constant during the ventilation, while the pressure changes between target inspiratory pressure and positive end-expiratory pressure. A new breath starts after a breathing cycle is over, and it is controlled by the respiratory rate (RR) and the ratio between inspiratory and expiratory (I/E). Sometimes the user can spontaneously initiate a breath when a sudden drop in pressure is detected within the trigger window during expiration. When the patient is able to control his breathing, but still needs support, PSV mode is the most appropriate mode because MVM partially takes over the work of breathing. A new breath starts when the ventilator detects a sudden drop in pressure, while the expiration starts when the patient's inspiratory flow drops below a set fraction of the peak flow. If a new breath is not detected within the apnea delay (a time window set by the physician), the ventilator automatically moves to PCV mode because it is assumed that the patient is not able to breathe alone. Two valves allow the air to enter/exit from the patient, i.e., an input valve and an output valve. When the ventilator is in the inspiration phase, the input valve is opened and the output valve is closed, while in the expiration phase the input valve is closed and the output valve is opened. When the ventilator is not running, or in case of emergencies, valves are set to safe mode: the input valve is closed and the output valve is opened. During ventilation, three special operations can be performed by the physician: inspiratory pause, expiratory pause, and recruitment maneuver.

Current high-level software architecture is shown in Fig. 1. It is composed of three components: GUI (Graphical User Interface), controller, and supervisor. The GUI (see Fig. 2), a touch screen, allows the physician to control patient parameters and interact with the ventilator to set ventilation parameters; the controller receives input from the GUI and interacts directly with the hardware reading patient's data and issuing commands; the supervisor monitors the overall behavior of the ventilator and intervenes in case of errors.

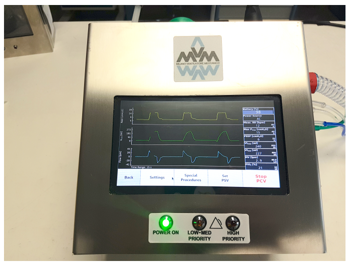


Figure 2: MVM Graphical User Interface on board.

3 METHODOLOGY

In order to develop the MVM mobile application, we have adopted an Agile approach to keep development cycles small and incremental. Moreover, since the application is new, and we did not find any similar application to monitor and manage the ventilator remotely, we use a prototyping approach. Instead of directly developing the final application, we build several prototypes. Here, we describe how we have designed and developed the final Android prototype by following these steps:

1. analysis of existing applications for ventilation;
2. analysis of the existing GUI;
3. writing of the requirements specification document for the mobile application;
4. prototyping on paper (POP);
5. prototyping using a software: Justinmind;
6. working prototype implemented in Android;

During all these steps, a pulmonologist has been consulted to discuss the functionalities that the MVM application should provide and to get suggestions for improvement.

The main goal of the first step is to extract the advantages and disadvantages of existing applications, in order to identify those functionalities that should be provided by the MVM application. The obtained results are shown in Sect. 4.

The second step consists in analyzing the GUI currently used to interact with the ventilator. We have analyzed the user manual and the requirements specification from which we have extracted the functionalities that can be performed also from the mobile application to allow the physician to interact remotely.

In the third step, we formally defined the mobile application requirements by integrating the functionalities available in the GUI and the tips from the physician. In particular, the physician helped us to improve the application security in order to avoid patient harm and the usability of the application. The application requirements are explained in Sect. 5.

Then, the prototyping design started. The prototype has been designed following the prototyping on paper (POP) approach, which consists in drawing on paper shapes representing screens, buttons, text boxes, and other items of the user interface. This is the simplest method to prototype applications and allows the designer to have a general view of what will be the final result. The second approach adopted to prototype is using dedicated software: Justinmind[2]. Justinmind allows us to prototype mobile and web applications analogous to realistic applications. After that, we started the implementation on Android Studio to get a working prototype. It does not have all the functionalities as defined in the requirements specification document, but it is a starting point to develop the complete MVM mobile application.

4 EXISTING APPLICATIONS FOR VENTILATION

Before starting the design of the MVM mobile application, we analyzed existing applications related to mechanical ventilators (Agudelo and Sánchez, 2020). We have found seven applications.

Four of them are for educational purposes, they provide ventilator setting parameters based on patient data given as input by the expert: Basics of Mechanical Ventilation (Unknown, 2014), VentilO (IUCPQ, 2020), VentICalc (van Beukering et al., 2020) and Mechanical Ventilation Expert (Streltsov, 2022). The other three applications simulate or are connected to the ventilator, for each of them we have extracted advantages and disadvantages as summarized in Table 1. MyVenus (Lab, 2022; Battista, 2016) is used for patients that suffer from chronic respiratory failure or acute respiratory diseases and are mechanically ventilated at home. Using this device the physician can remotely monitor, using a web interface, the ventilation parameters e.g. respiratory rate, inspiration time, peak flow, and tidal volume. Moreover, it is able to detect apnea and patient tube disconnection. TruVent App (Trucorp, 2022) is a training application. The trainer sets patient characteristics and the learner monitor and performs actions based on patient status, which can be modified continuously by the trainer in order to test different clinical conditions. Hamilton Connect App (medical, 2022) allows the physician to access selected ventilation data in order to monitor patients remotely. Moreover, it offers a demo mode that allows exploring all the application functionalities. None of the presented applications is able to control remotely the ventilator in order to monitor and also control patient clinical status.

[2]https://www.justinmind.com/

5 APPLICATION REQUIREMENTS

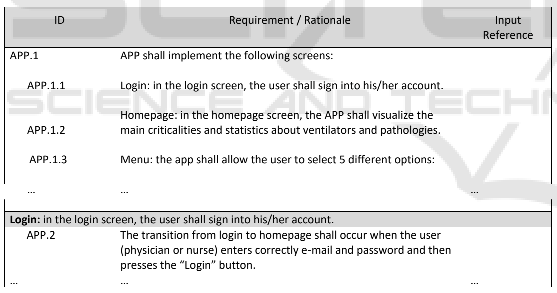
The possibility of remote management that this application allows, reduces the response time of specialized personnel given the high demand for ventilators and specifically in the cases of patients who totally depend on the use of this equipment and require greater attention and care by intensive care physician (Miño et al., 2021).

To provide safe care to ventilated patients, the number of healthcare professionals who are allowed to adjust the ventilator should be limited. Informing the clinician early about potential complications or patient tolerance of ventilator changes can provide coordinated care and improve patient outcomes. Every time an adjustment is made on the ventilator, there must be a notification to the critical care nurse in charge in order that the settings can be reviewed, and an appropriate clinical observation is guaranteed for the safety of the patient. It is crucial a local control and careful verification of remote prescriptions by qualified health professionals. Furthermore, any parameter and alarm prescription change on ventilatory devices should be clearly recorded, documented, and communicated to the entire healthcare team. A multiparameter monitoring of the patient and a collaborative and trained interprofessional team are necessary conditions for safe and effective mechanical ventilation. It might be reasonable to define shared protocols and algorithms in individual ICUs to avoid potential confusion from competing loci of control and information, including alarm signals, introduced by the remote-control system. A careful study of the management of the clinical risk deriving from the remote control of mechanical ventilation and related cybersecurity will be necessary to define the possible clinical implications and obtain the authorizations of the Healthcare regulatory Agencies (Williams LM, 2022; Branson et al., 2016; AAMI, 2020). An accurate evaluation of risks associated with a medical device remote control in the tele-critical care context is strictly recommended.

After having analyzed existing applications and the existing MVM GUI, we have defined the MVM application requirements by writing the Software Requirement Specification document. Fig. 3 shows an

Table 1: Existing applications for ventilation.

Advantages	Disadvantages
MyVenus	
• continuous monitoring of home mechanical ventilation • visualization of treatment at the patient's home via web interface for caregivers • web interface optimized for different devices with numerical parameters and waveforms • apnea detection and patient tube disconnections	• dark and basic layouts • only a web interface • not connected directly to the ventilator, but a specific device is required • not yet on the market
TruVent App	
• performs realistic medical simulation training without the need for expensive/high-fidelity mannequins • the instructor can adjust patient characteristics • intuitive interface • complex scenarios can be run • shows ventilation alarms	• not connected to ventilators, it is an interactive remote learning ventilation app • concentration of information on one single screen • needs clinical judgment
Hamilton Connect App	
• test the app even without the ventilator when in demo mode • management of connected devices • respiratory data wherever you want • freeze and change the time scale for a closer view • customizable view: layouts that suit the physician's needs	• not intended to replace the real-time display of data on the ventilator • dark layouts



ID	Requirement / Rationale	Input Reference
APP.1	APP shall implement the following screens:	
APP.1.1	Login: in the login screen, the user shall sign into his/her account.	
APP.1.2	Homepage: in the homepage screen, the APP shall visualize the main criticalities and statistics about ventilators and pathologies.	
APP.1.3	Menu: the app shall allow the user to select 5 different options:	
...	...	...
Login: in the login screen, the user shall sign into his/her account.		
APP.2	The transition from login to homepage shall occur when the user (physician or nurse) enters correctly e-mail and password and then presses the "Login" button.	
...	...	...

Figure 3: Excerpt of Software Requirement Specification document.

excerpt of the requirements document, we have defined each requirement and assigned an identifier in order to track it through all the documents produced during the application development (like e.g. in testing documentation).

We have identified nine views: Login, Homepage, New patient, PSV mode, PCV mode, Ventilator list, Monitor and Alarms setting.

Login. After logging in using the username and password provided, the system allows different operations based on user type: the nurse/medical assistant can monitor the connected ventilators and check if alarms have been raised by the ventilator; the physician, in addition to the features provided for the nurse, can connect/disconnect new patients and set ventilation parameters.

Homepage. From the homepage the user can view the list of ventilators alarms based on their criticality. By clicking on the single alarm he/she is automatically redirected to the monitor screen. On the home page, some statistics related to the state of ventilators (available or connected) and patients' pathologies are shown. The following screens are reachable from the menu on the homepage: New patient, Ventilator list, and Monitor.

New Patient. The user can insert patient information through New patient screen. The required information is an ID (usually the fiscal code), name, surname, date of birth, height (in cm), gender, and patient disease. Then, based on these data, the application automatically computes the predicted body weight (PBW), the required parameters for the ventilation. All the data are stored in a database by clicking the save button. Then the physician (this function is not available for nurse/medical assistant) selects the ventilator through a drop-down menu showing all the available ventilators. Then by clicking on the corresponding button, the physician inserts the ventilator parameters for both PCV and PSV modes.

PCV Mode. The physician inserts all the parameters required for running the ventilator in PCV mode: respiratory rate (RR), inspiratory/expiratory ratio (I:E), target inspiratory pressure (P_{insp}) and inhale trigger sensitivity (ITS). Then all the parameters are stored in a database. To start and stop the ventilation, the physician can use the appropriate buttons. For security purposes, these operations must be authorized through an authorization code displayed on the ventilator. This guarantee that someone is close to the patient when performing these critical operations, nevertheless the physician can do them remotely.

PSV Mode. The physician inserts all the parameters required for running the ventilator in PSV mode: target inspiratory pressure (P_{insp}), inhale trigger sensitivity (ITS), expiratory trigger sensitivity (ETS), apnea delay and the apnea backup parameters (RR, I:E and P_{insp}) when automatically the ventilator moves from PSV to PCV because the patient is not able to breathe on his/her own. As required for PCV mode, start and stop operations must be authorized through an authorization code displayed on the ventilator.

Ventilator List. This screen shows all the ventilators registered in the application. For each of them, the application shows if it is connected to the patient or not, if it is ventilating or not, and if alarms are present. By clicking on the ventilator, the user is redirected to the monitor screen.

Monitor. The monitor screen shows the real-time measured parameters and waveforms. The waveforms displayed on the screen are pressure in the airways (PAW), tidal volume (V_{tidal}), and the instantaneous flow. Then the following real-time parameters are shown: RR, PEEP, measured minute volume V_e, P_{insp}, V_{tidal}, the measured fraction of inspired oxygen (FiO_2). Then, only for the physician, it is possible to perform inspiratory and expiratory pauses and recruitment maneuver, which must be authorized by a code displayed on the ventilator.

Alarms Setting. The physician inserts the alarm parameters ranges: minimum and maximum RR, minimum and maximum P_{insp}, minimum and maximum V_{tidal}, minimum and maximum PEEP. All these values are stored in the database and used during the ventilation to raise alarms when measured parameters are out of range.

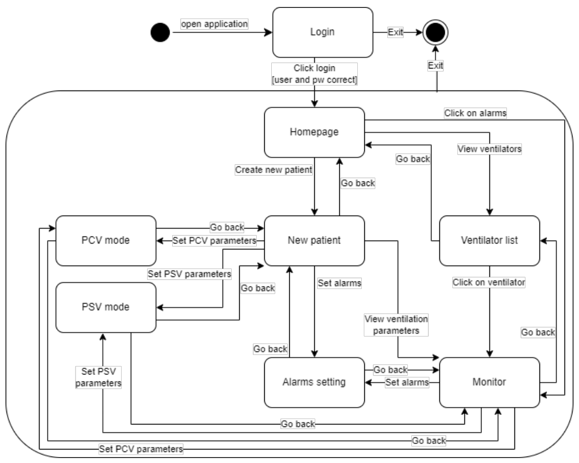


Figure 4: State machine.

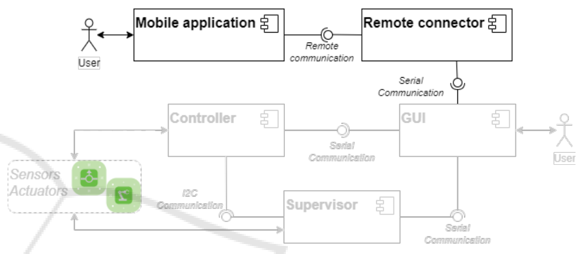


Figure 5: New high-level software architecture.

An overview of the transition between application screens is given in Fig. 4.

To mitigate risks associated to remote control, we have included in the application some procedures that require an authorization code shown on the GUI of the ventilator. This is necessary to guarantee that during critical operations, like start/stop ventilation and pauses, nevertheless the physician performs them remotely, a qualified person is close to the patient to monitor his/her health state. Moreover, an authorization code is required whenever connecting to the ventilator in case the physician wants to update ventilation parameters or connect a new patient. It is not required only if the physician intends to view measured parameters without performing any action.

5.1 Architecture

The existing MVM architecture (see Fig. 1) does not support mobile application integration because, for cybersecurity problems, all the remote connection modules have been removed. Fig. 5 shows the new architecture in which new components are added in order to allow communication between the ventilator and the mobile application.

The new architecture requires a new module integrated into the hardware where GUI is implemented. This module read and sends data from the ventilator

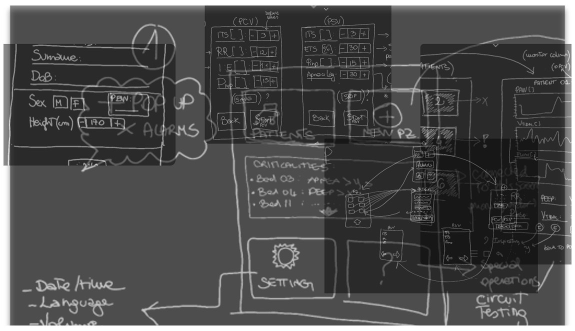


Figure 6: Prototyping on paper.

to the mobile application and vice versa. When implementing this module, the cyber security risk must be considered in order to avoid systems vulnerabilities (e.g. ventilator parameters are changed by an external application not authorized).

6 APPLICATION PROTOTYPE

As written in Section 3, we have followed two steps for the application prototype: prototyping on paper (POP) and prototyping using the software.

Fig. 6 shows the POP result. This initial prototype helped us to have an immediate overview of the entire application, by understanding the information displayed by the application and the information required to the user.

The second prototype has been implemented using Justinmind tool (see Fig. 7). The obtained prototype is clickable and fully-functional, without writing a single line of code. This step required less effort than implementing in Android, but allowed us to have a usable application showable to the physician from which we started to analyze the characteristics, advantages, and disadvantages of the implemented prototype.

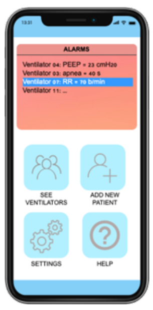
Given the prototype, we got important feedback from the physician. The most important and critical was the lack of direct control of the ventilator when performing critical operations like start and stop ventilation. For this reason, as already explained in Sect. 5, we have decided to introduce an authorization code shown on the ventilator, that must be communicated to the physician and used by him/her to confirm critical operations. This procedure guarantee that someone belonging to the medical staff is close to the ventilator and monitors the patient's status.

Another suggestion was about the functionalities available based on the role of the app user. In the prototype implemented in Justinmind, there was no difference between the app interface used by a physician or by a nurse/medical assistant, all the operations were allowed to everyone. This was not safe for patients because the competencies of medical staff are different, and not everybody is able to correctly configure the ventilator and start/stop patient ventilation. Given this, we have associated a role to each user, and based on it different operations are allowed.

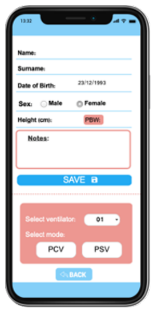
Performing different operations on the ventilator can be critical, and for safety purposes, it is very important to track who performs and when operations are performed. At the moment this functionality is not expected to be implemented in the first version of the application, but it will be taken into consideration for the next.

7 ANDROID IMPLEMENTATION

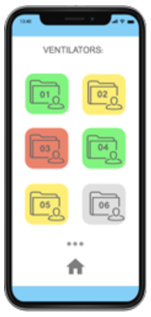
After having implemented the prototype using the two approaches presented above, we have implemented the first prototype in Android Studio as shown in Fig. 8. The first screen is for authentication (see Fig. 8a). At the moment, in this prototype, we have not distinguished between two roles as explained in the requirements section. The home page (see Fig. 8b) shows all the alarms raised by connected ventilators, and clicking on each alarm it is possible to directly access the ventilation screen in order to check measured parameters. At the bottom of the screen are shown how many ventilators are free and the type of disease for which patients need ventilator support. The main menu, different from the prototype implemented using Justinmind tool, is accessible from a drop-down menu, from where the user can access the following main functionalities: connect a new patient to the ventilator and see all the ventilators. When connecting a new patient (see Fig. 8c), some personal data are required, then the physician must specify the type of disease and select the ventilator to which the patient is connected among the free ones. Then the physician must set the PSV and PCV parameters (see Fig. 8f and Fig. 8e) to start the ventilation. In order to guarantee patient safety when performing a start and stop ventilation, an authorization code is required to authorize the activity. Moreover, alarms threshold are set using the alarms interface (see Fig. 8g). When the physician opens the screen to see all the ventilators (see Fig. 8d) is it possible to see for each a colored led: gray if the ventilator is disconnected, green if there are no alarms, yellow if there are warnings on the ventilator, red if there are critical alarm raised by the ventilator. In this prototype, all the connections with the real ventilators are omitted, because the ventilator does not have a module to interact remotely.



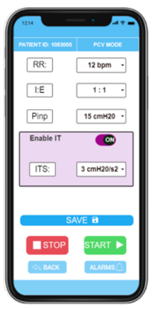
(a) Home.



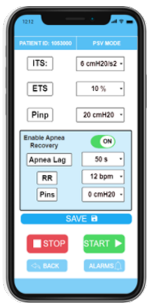
(b) New Patient.



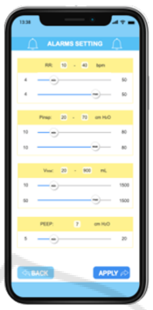
(c) Ventilator List.



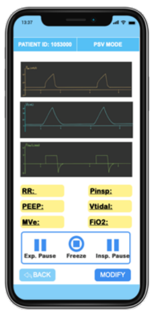
(d) PCV mode.



(e) PSV mode.

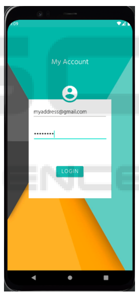


(f) Alarms settings.

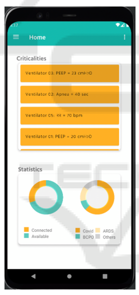


(g) Ventilator monitor.

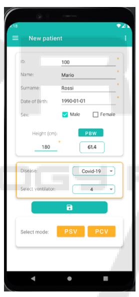
Figure 7: Prototype implemented using Justinmind tool.



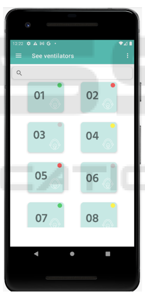
(a) Login.



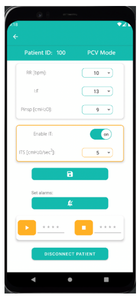
(b) Home.



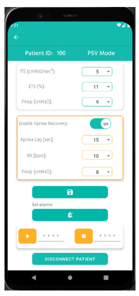
(c) New Patient.



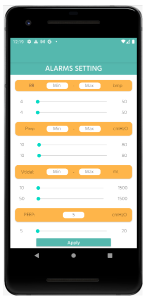
(d) Ventilator List.



(e) PCV mode.



(f) PSV mode.



(g) Alarms settings.

Figure 8: First prototype in Android.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we have shown the methodology adopted in order to develop a first prototype of the mobile application to monitor and control patients connected to the ventilator. Starting from the GUI, already integrated into MVM, we have defined requirements and we have implemented two prototypes, the first using the approach of prototyping on paper and the second prototype having been developed using Justinmind tool. Starting from the prototype, we have integrated physician comments, especially we have included that critical operations must be confirmed by a code displayed on the device (this implies that someone is near the ventilator, but at the same time the physician can perform actions in order to treat the patient). This work is just preliminary, and it has been performed in order to check the feasibility. Many other features must be implemented as future work to get the final working application. For example, depending on the role of the user only patient monitoring is available, or also ventilator control. The most critical part is to connect the application with the ventilator because this requires changes also on the MVM as shown in the new architecture. While, in order to make the application available to the largest number of users, it should be developed using a cross-platform development environment.

ACKNOWLEDGEMENTS

We would like to thank Alice Lucchini, Dounia Ahbar, Tatiana Vasilache, and Martina Brambilla for the preliminary work done for this project.

REFERENCES

AAMI (2020). Emergency use guidance for remote control of medical devices.

Agudelo, J. C. M. and Sánchez, M. B. S. (2020). Medical learning tool for ventilator weaning protocols. *Revista EIA*, 17(34).

Barrow, M., Restuccia, F., Gobulukoglu, M., Rossi, E., and Kastner, R. (2022). A remote control system for emergency ventilators during SARS-CoV-2. *IEEE Embedded Systems Letters*, 14(1):43–46.

Battista, L. (2016). A new system for continuous and remote monitoring of patients receiving home mechanical ventilation. *Review of Scientific Instruments*, 87(9):095105.

Bombarda, A., Bonfanti, S., Galbiati, C., Gargantini, A., Pelliccione, P., Riccobene, E., and Wada, M. (2022). Guidelines for the development of a critical software under emergency. *Information and Software Technology*, 152:107061.

Branson, R., Godwin, T., Hargett, J., Papadakos, P., Rodriquez, D., Stampor, L., and Strickland, S. (2016). Safe initiation and managementof mechanical ventilation. *American Association for Respiratory Care and UniversityHealthSystem Consortium's*.

IUCPQ (2020). VentilO.

Lab, B. (2022). MyVenus.

medical, H. (2022). Hamilton Connect App.

Miño, C., Flor, O., Quiroga, J., and Cuaycal, A. (2021). Remote variable monitoring app for mechanical ventilators used in COVID-19. In *Artificial Intelligence, Computer and Software Engineering Advances*, pages 303–315. Springer International Publishing.

Streltsov, V. (2022). Mechanical Ventilation Expert.

Trucorp (2022). TruVent App.

Unknown (2014). Basics of Mechanical Ventilation.

van Beukering, S., de Ruijter, P., Sreekantan, S., der Hoeven, J. V., and Workum, J. (2020). VentICalc.

Williams LM, S. S. (2022). *Ventilator Safety*. StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing.