An Android App for Training New Doctors in Mechanical Ventilation

Andrea Bombarda[1][a], Sara Noto Millefiori[1], Michela Penzo[1], Luca Novelli[2][b], and Angelo Gargantini[1][c]

[1]*Department of Management, Information and Production Engineering, University of Bergamo, Bergamo, Italy*
[2]*Pulmonary medicine Unit, ASST Papa Giovanni XXIII, Bergamo, Italy*
{*andrea.bombarda, angelo.gargantini*}*@unibg.it, lnovelli@asst-pg23.it,*
{*s.notomillefiori, m.penzo1*}*@studenti.unibg.it*

Keywords: Mobile Application, Mechanical Ventilation, Application Prototype, Training

Abstract: Mechanical ventilation is essential for critically ill patients, as recently demonstrated by the COVID-19 pandemic. The experience of physicians in correctly selecting ventilation parameters and values plays a crucial role in ensuring the best possible outcome. In order to aid physicians in setting up a mechanical ventilator, several brands have implemented in their products an adaptive ventilation mode called Adaptive Support Ventilation (ASV). This mode automatically selects pressure and respiratory rate to require the patient the minimum breathing effort possible. However, physicians are generally skeptical about adopting this ventilation mode, as they prefer to have all parameters under their control. Nevertheless, we believe that comprehending how ASV works is paramount important, to understanding the patterns used, and possibly exploiting them while manually setting mechanical ventilators. For this reason, in this paper, we present `Ventilation App`, an Android app for training new physicians in mechanical ventilation. It allows the simulation of a ventilation process for a patient unable to breathe and gives feedback to the user by exploiting the same operating principles of the ASV mode. Thanks to the feedback received by a collaborating physician, we believe that our app can be useful for allowing physicians-in-training to acquire proficiency in mechanical ventilation.

1 INTRODUCTION

Mechanical ventilation plays a crucial role in the treatment of critically ill patients, and the proficiency of physicians in this area is vital to provide optimal patient care. However, acquiring the necessary skills and knowledge in mechanical ventilation can be a challenging task, as it requires a deep understanding of complex respiratory physiology and the ability to interpret and adjust ventilation parameters effectively. To solve this issue, automatic ventilation strategies such as the Adaptive Support Ventilation (ASV) modes have been proposed. In this way, the mechanical ventilator automatically defines the ventilation parameters and values based on the patient's respiratory mechanics, and by computing them by using the Otis' curve (Fernández et al., 2013). However, the complexity of ASV can give rise to concerns and skepticism among healthcare professionals. Physicians may question its efficacy and reliability, especially when compared to traditional modes of ventilation that they are more familiar with and that allow them to manually manage the ventilation parameters. Additionally, most clinicians may be hesitant to fully embrace ASV due to a lack of comprehensive training or limited exposure to its implementation in real-world clinical settings.

To bridge this gap and facilitate the learning process for aspiring physicians, in this paper, we introduce an Android application aimed at enhancing training in mechanical ventilation by taking advantage of the mechanisms used by the ASV strategy, used by ventilators when automatically deciding the ventilation strategy. It is based on the principle of learning by doing, i.e., it lets the user manually set the ventilation parameters after having decided on the physiological and mechanical characteristics of the patient. Then, the app simulates the mechanical ventilation and gives feedback to the user by using the measured effort and its distance from the optimal, i.e., the minimum one computed using the Otis equation as automatically done by mechanical ventilators employing the ASV mode. In this way, physicians-in-

[a] https://orcid.org/0000-0003-4244-9319
[b] https://orcid.org/0000-0002-2705-248X
[c] https://orcid.org/0000-0002-4035-0131

training can understand the ventilation patterns used by the ASV mode and, possibly, exploit them while manually setting mechanical ventilators.

With this paper, our contribution is twofold. On the one hand, we contribute our Android app allowing physicians to train with mechanical ventilation. On the other hand, we apply the paradigm of learning-by-doing in the context of healthcare, especially devoted to increasing the understanding of physicians in automatic procedures that are normally avoided for lack of confidence. This approach can be generalized to other fields, beyond mechanical ventilation, where simulators are available.

The remainder of the paper is structured as follows. Sect. 2 presents the background on mechanical ventilation, including operating principles and modes. Moreover, it includes a brief analysis of existing Android apps for training new physicians in terms of mechanical ventilation. In Sect. 3, we report the requirements and the architecture of the application we propose in this paper. Sect. 4 presents the implemented app and explains its functioning. Finally, Sect. 5 reports related work on apps for training physicians in different fields, while Sect. 6 reports future work directions and concludes the paper.

2 BACKGROUND

In this section, we report some background on the basis of mechanical ventilation, together with an analysis of already existing apps for training physicians in using mechanical ventilators.

2.1 Mechanical ventilation

Artificial mechanical ventilation replaces (or integrates) the activity of the inspiratory muscles to ensure a sufficient supply of gas to the lungs. It is a necessary tool in every modern intensive care unit (ICU). The type and intensity of ventilation support required by a patient vary over the course of treatment. Modern mechanical ventilators are versatile and adapt to patient needs. They support patients who cannot breathe or those who can still trigger a mechanical cycle by a spontaneous inspiratory effort. Present-day mechanical ventilators are complex machines, consisting of many specialized components and featuring several ventilation modes, which have to be set depending on the conditions of the patients. There are numerous conditions in which it is necessary to resort to mechanical ventilation, for example in all those conditions in which spontaneous natural breathing is not possible (e.g., in the case of acute respiratory distress syndrome - ARDS - or chronic obstructive pulmonary disease - COPD).

Ventilation modes Depending on the patient and the type of ventilator available, several ventilation modes (i.e., specific combinations of breathing patterns, control types, and operational algorithms) are available in the products on the market, and the most common are those based on pressure or volume control. Most of them operate in an open-loop way.

For what concerns volume-control ventilation modes, the two most common modes are Mandatory Minute Ventilation (MMV) and Controlled Mandatory Ventilation (CMV). With the former, the ventilator provides a predetermined minute ventilation (i.e., a pre-defined volume of air) when the patient's spontaneous breathing effort becomes inadequate (e.g., because of an apnea). When this occurs, the mandatory frequency is increased automatically to compensate for the decrease in minute ventilation caused by the apnea and to ensure the desired minute ventilation. With the latter, the ventilator controls both the patient's tidal volume (i.e., the volume of air inspired/expired in each breath) and respiratory rate. In practice, the ventilator controls the patient's minute volume. This means that a patient cannot change the ventilator frequency or breathe spontaneously.

Instead, the two main pressure-control ventilation modes are Pressure Controlled Ventilation (PCV) and Pressure Support Ventilation (PSV). In PCV mode, the ventilator controls the timing of the breathing cycle and regulates the pressure applied to the patient. PCV mode is used in the acute phase of the disease when patients are deeply sedated or paralyzed, and cannot breathe themselves. On the other hand, PSV is an assisted ventilatory mode that is patient-triggered, pressure-limited, and flow-cycled. The main use of this mode is for the weaning of the patient from mechanical ventilation because it gradually unloads the work of breathing.

Ventilation parameters Depending on the ventilation mode, different parameters have to be set by the physicians. In the following, a brief explanation of the main ventilation parameters we exploit in the work presented in this paper is given.

The respiratory rate (RR) indicates the number of breaths per minute the patient has to take. For each breath, the ratio between the inspiratory and expiratory times is set through the parameter I:E. During the inspiration phase, the ventilator reaches the maximum pressure P_{max}, which has to be set accurately in order not to damage the patient's lung with too high pressure. On the other hand, during the expiration

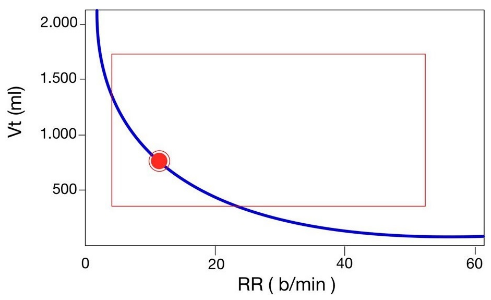


Figure 1: Example of the Otis curve for the minimal WOB. The red dot indicates the optimal target ventilation point

phase, the ventilator sets its pressure to the positive end-expiratory pressure (PEEP), which is greater than zero to prevent the collapse of the airway.

2.2 Adaptive Support Ventilation

Adaptive support ventilation (ASV) is a mechanical ventilation mode that operates in a closed-loop system, adjusting the ventilator's output for each breath. It utilizes the patient's measured mechanical characteristics (e.g., airway resistance and compliance) and breathing effort to determine the appropriate level of ventilatory support. The primary objective of ASV is to deliver a predetermined level of alveolar ventilation while minimizing the combined work of breathing (WOB) for both the patient and the ventilator. ASV can provide complete or partial ventilatory support and is applicable during mechanical ventilation's initiation, maintenance, or weaning stages. The ventilation pattern is automatically decided by the ventilator for reducing the WOB based on the Otis equation

$$RR(t) = \frac{\sqrt{1 + 2 \cdot a \cdot RC_e \cdot \frac{V_{min} - RR(t-1) \cdot V_D}{V_D}} - 1}{a \cdot RC_e}$$

where RC_e is the expiratory time constant (defined through the resistance R and compliance C of the patient's airway), V_{min} is the target minute ventilation, V_D represents the dead space volume (computed as $2.2 \cdot IBW$, where IBW is the ideal body weight), a is a factor depending on the flow waveform (e.g., it is 0.329 for sinusoidal flows), and RR is the respiratory rate, i.e., the number of breaths per minute. Based on Otis's observations, if the ventilation pattern (respiratory rate and tidal volume) satisfies the proposed equation, then the effort requested to the patient for breathing is reduced to the minimum.

The ASV mode uses the Otis equation and considers the curve in Fig. 1 for automatically adapting the respiratory rate (RR) and the tidal volume ($V_t = {}^{V_{min}}\!/_{RR}$), i.e., the volume inspired or expired during a single respiratory cycle. Based on this curve, every ventilation pattern that guarantees to stay on the blue line is optimal and lowers the patient's effort to the minimum possible.

Note that the Otis curve identifies some safety limits (see the red rectangle in Fig. 1), which are used by ventilators when working in the ASV mode in order to raise alarms and avoid dangerous conditions. In terms of respiratory rate (x-axis in Fig. 1), the lower limit is 5 respiratory cycles per minute, while the upper bound is computed as ${}^{20}\!/_{(R \cdot C)}$ and, in any case, the respiratory rate cannot exceed 60 breaths per minute. On the other hand, in terms of tidal volume V_t (y-axis in Fig. 1), the safety limits depend on the patient's IBW. Indeed, the minimum tidal volume can be $4.4\ {}^{ml}\!/_{kg} \cdot IBW$, while the maximum is $22\ {}^{ml}\!/_{kg} \cdot IBW$ and, in any case, the pressure applied by the ventilator in order to achieve the intended tidal volume has to be lower than $45cmH_2O$.

Despite being automatic, the ASV mode is normally avoided by physicians who prefer to manually set ventilation parameters in order to keep them under control. However, we believe that it is essential for every doctor to understand how ASV works. In this way, physicians can learn some of the patterns that are normally used by ASV and integrate them into their manually defined settings. This is the rationale behind the application we propose in this paper.

2.3 Existing apps for training in mechanical ventilation

Before starting the design of our app, we analyzed existing applications related to training in mechanical ventilation. With this analysis, we found four apps. In the following, we give a brief overview of their features and the differences between them and the application we propose in this paper.

TruVent App (TruCorp, 2023) is a training application in which the trainer and the physician interact. The former sets patient characteristics, while the latter monitors and performs actions based on patient status, which the trainer can continuously modify by simulating different clinical conditions. However, the TruVent App is not based on ASV nor uses the Otis curve to give users feedback, as we do in our app. Similarly, Ventsim (Ventsim, 2023) simulates interactions between a patient and a ventilator and generates waveforms. Different modes of ventilation are implemented, and clinical scenarios can be tested by changing the ventilator and patient settings. However, as TruVent App, it does not offer a way for the physician to understand whether the settings are those leading to the minimum breathing effort for the patient.

VentilO (IUCPQ, 2023) aims at training physicians in reasoning on how different values for the ventilation parameters may influence the patient's condition. Nevertheless, VentilO only implements volume-based ventilation modes and, thus, the pressure-controlled ones we want to deal with are not supported, nor training for users is given in those cases.

Finally, OpenPediatrics (OpenPediatrics, 2023) provides a ventilator simulator embedding the most recent pediatric and adult ventilation guidelines, as well as COVID-19 information and COVID-19 patient cases. As for the TruVent app, the ventilator simulator provided by OpenPediatrics is not based on ASV nor uses the Otis curve to give users feedback, as we do in our proposed app.

We can conclude that, to the best of our knowledge, no Android app with the same operating principles as those we embed in `Ventilation App` exists.

3 APP DESIGN

At the beginning of the development process, we held several meetings with physicians and experts to have a deep understanding of how mechanical ventilation works, especially in the context of the ASV mode, and of the most perceived difficulties by physicians-in-training when approaching this ventilation strategy. Then, together with physicians and experts, and thanks to the experience gained in the past by groups working on the development of mechanical ventilators (e.g., for treating patients affected by COVID-19 in ICUs) (Abba et al., 2021; Bombarda et al., 2021; Bombarda et al., 2022; Pearce, 2020), we defined the functional requirements of the Android app for training new doctors in mechanical ventilation. In this way, we have been able to identify critical features and isolate those that were not of interest for the scenario we wanted to deal with, i.e., that of the ASV mode. Then, the app architecture has been defined according to the identified requirements.

3.1 Functional requirements

During the process of the development of our Android app, we started by individuating the requirements, which were then formalized in the Software Requirements Specification document. For each requirement, we have identified a description (sometimes with a rationale, when it was provided by the experts we contacted) and an ID. In the Software Requirements Specification, we have identified 4 different modes, and, for each of them, we have defined the set of needed requirements. In the following, we describe each mode in detail.

Startup mode: In startup mode, the Android app shows the logo and offers the user the possibility to start using the app (see *Settings mode*) or to view information on its operating principles (see *Info mode*).

Info mode: In the info mode, the Android app reports a message explaining its operating principle, including the thresholds used for defining when the breathing effort is low, medium, or high.

Settings mode: In the settings mode, the user is requested to set all the required parameters for the simulation, i.e., both those of the patient and those of the ventilator. For what concerns the patient, the app asks the user to insert the gender, age, weight, height (used to calculate the IBW), airway resistance, and compliance. Then, for what concerns the ventilator, the user must insert the basic ventilation parameters, i.e., the PEEP value, the P_{max}, the I:E fraction, and the respiratory rate. When in this mode, the user may save the data previously set into an embedded database, in order to load them in another simulation, or load the data from those previously saved. When all the parameters are set, the user may proceed by starting the *Simulation mode* or by closing the app.

Simulation mode: The simulation mode is the core mode of the Android application. By using all the data provided by the user during the *Patient parameters set mode*, it simulates the patient's breathing when the mechanical ventilator works in PCV mode, i.e., when the patient is not able to breathe autonomously. During the simulation, two plots are shown: one that reports the pressure over time, and one showing the flow over time. In order to allow users to understand the effect and level of correctness of the settings, during simulation mode, a semaphore is shown: the green light is highlighted when the measured effort is equal to the minimum possible (see the Otis' equation in Sect. 2.2) with a tolerance of ± 150, the yellow light is highlighted when the measured effort is distant from the optimal one from ± 150 to ± 300, otherwise, the red light is shown. These threshold values have been discussed and approved by physicians and have been derived from the examples given in the operator's manual of the Hamilton Galileo ventilator[1], which embeds the ASV mode.

As previously explained in Sect. 2.2, safety limits are identified by the ASV mode. The red light is shown even if the chosen ventilation pattern does not imply a high effort for the patient, but it is outside those safety limits. When in simulation mode, the user may change some configuration parameters (see *Settings mode*), freeze the simulation (see *Freeze mode*), or stop the simulation and close the app.

[1]https://bit.ly/3Zh4PIy

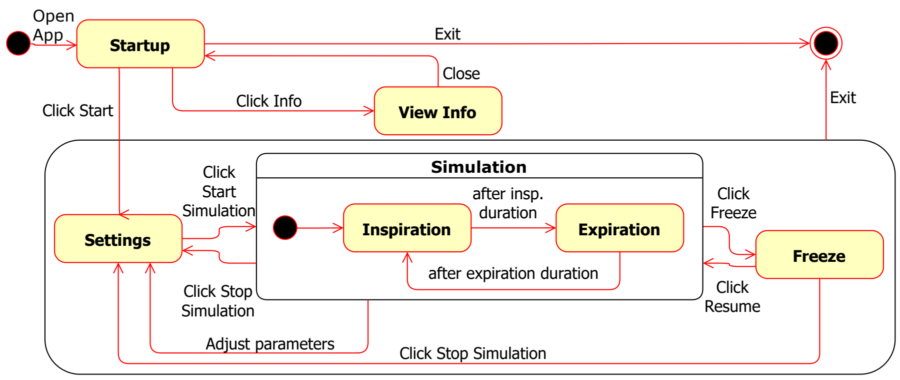


Figure 2: State machine

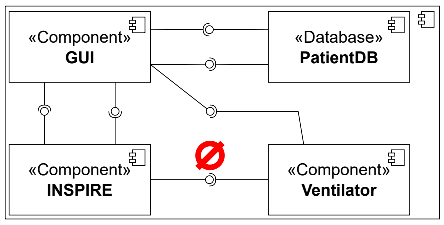


Figure 3: High-level software architecture

Freeze mode: The freeze mode can be activated by the user during the simulation. When the freeze mode is active, the simulation pauses and the user can visualize the plot details in order to better understand the patient's condition. Users may exit from the freeze mode by restarting the simulation (see *Simulation mode*) or by stopping the simulation and setting new patient parameters (see *Settings mode*).

A general overview of the transitions between the application modes and the functionalities is given in the state machine in Fig. 2, while a diagram showing the interaction between actors and software artifacts is available in our open-source repository at https://github.com/foselab/VentilationApp.

3.2 App architecture

The high-level architecture of the training app is reported in Fig. 3. It is composed of 4 main components. The `INSPIRE` component implements the patient simulation logic. It handles the electronic equivalent of human lungs (van Diepen et al., 2021), composed by the resistance R and compliance C set by the user through the `GUI`, and when a defined pressure is applied by the ventilator, it computes the current flow and pressure in the patient's airway. The `Ventilator` component reads the ventilator's parameters from the `GUI` and simulates a simple mechanical ventilator working in PCV mode (see Sect. 2.1). It sets the pressure (between the $PEEP$ and P_{max} values) and sends its value to the `INSPIRE` component, through a ZeroMQ[2] communication channel, which executes a single simulation step. The `PatientDB` stores the patient and ventilator parameters saved by the user in previous simulations. It is also used by the `GUI` for loading previously saved data. Finally, the `GUI` component shows the user all the Android app screens, including the two plots and the semaphore that signals the effort. Through the `GUI`, the user can start/stop/freeze the simulation, set the configuration parameters for the ventilator and those of the patient, load them from the `PatientDB`, or decide to store the data of the current simulation run into the database.

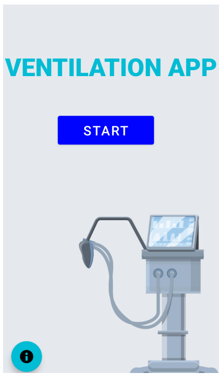
Extension points: The architecture has been chosen to promote the future extensibility of the app. New ventilatory strategies may be implemented by simply modifying the `Ventilator` component. Similarly, considering that the accuracy of the simulation depends on the patient models, more complex and realistic ones (including other parameters beyond R and C) can be easily added in future releases, as the `INSPIRE` component can simulate any kind of circuit representing the respiratory system.

4 APP PROTOTYPE

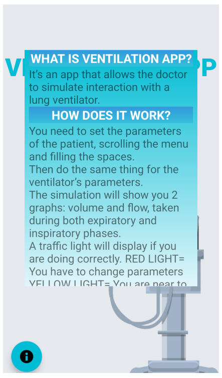
In this section, we describe the prototype of our training `Ventilation App`, available and open source at https://github.com/foselab/VentilationApp. It has been developed for mobile devices, with a responsive design, in order to make it suitable both for smartphones and tablets. Fig. 4 shows the Android application screens, which we better explain in the following.

The app starts with the splash screen shown in Fig. 4a, which contains two buttons, i.e., one for accessing the info section and one for starting the training. When the info button is clicked, the app

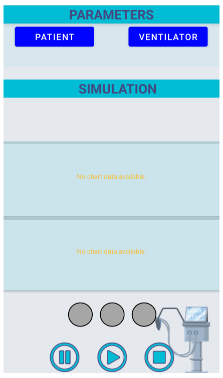
[2]https://zeromq.org/



(a) Splash screen


WHAT IS VENTILATION APP?
It's an app that allows the doctor to simulate interaction with a lung ventilator.
HOW DOES IT WORK?
You need to set the parameters of the patient, scrolling the menu and filling the spaces.
Then do the same thing for the ventilator's parameters.
The simulation will show you 2 graphs: volume and flow, taken during both expiratory and inspiratory phases.
A traffic light will display if you are doing correctly. RED LIGHT= You have to change parameters YELLOW LIGHT= You are near to

(b) Info screen



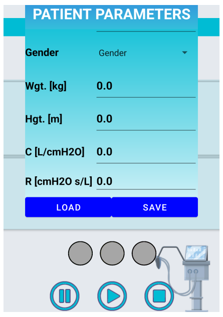
(c) Main window

Figure 4: Startup and View Info mode in the First prototype in Android

shows information about its purpose and functioning (Fig. 4b). Instead, when the start button is clicked, the app moves to the main window (shown in Fig. 4c).

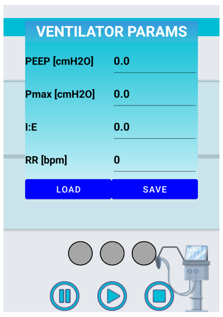
The main window contains the core functionalities of our `Ventilation App`. It contains real-time charts reporting the simulation outcomes, namely the airflow in the patient's airways and the pressure measured at the patient's mouth. This information is used by physicians to understand potentially dangerous patient conditions, such as the auto-PEEP (i.e., when the pressure in the patient's airways increases breath after breath, due to a too-short expiration time). In addition, two buttons allowing the setting of patient characteristics and ventilation settings are present. For what concerns the patient (Fig. 5a), the user is requested to insert the gender, age, weight, height (used to calculate the IBW), airway resistance, and compliance. Then, in terms of ventilation parameters (Fig. 5b), the user must set the value of the PEEP, the P_{max}, the I:E ratio, and the respiratory rate.

As introduced in Sect. 3, `Ventilation App` allows the user to load both patient configurations and ventilation parameters from an embedded database (see Fig.3). In the case of patients it can be done by clicking on the load button in Fig. 5a. In this way, the list of patients (including age, gender, height, weight, resistance, and compliance) is shown (Fig. 5c). The user may then decide to load the desired patient by clicking on the "check" button or to delete one of the patients by clicking on the trash bin icon. If a patient is chosen, all the fields in the patient configuration (Fig. 5a) are automatically filled with the values read from the DB. Similarly, as described for the patient parameters, the user can load the data related to the ventilator configuration with a specific load button (Fig. 5d). Therefore, a dedicated list is shown with values of PEEP, PMax, I:E, and RR. Also in this case, it is possible to select or delete the data with two dif-


PATIENT PARAMETERS

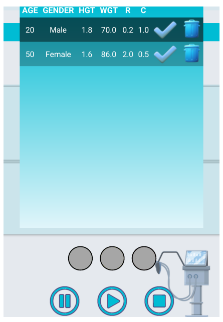
Gender	Gender
Wgt. [kg]	0.0
Hgt. [m]	0.0
C [L/cmH2O]	0.0
R [cmH2O s/L]	0.0

(a) Patient's settings


VENTILATOR PARAMS

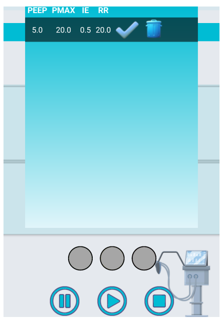
PEEP [cmH2O]	0.0
Pmax [cmH2O]	0.0
I:E	0.0
RR [bpm]	0

(b) Ventilation settings



AGE	GENDER	HGT	WGT	R	C		
20	Male	1.8	70.0	0.2	1.0	✓	🗑
50	Female	1.6	86.0	2.0	0.5	✓	🗑

(c) Patient selection from DB



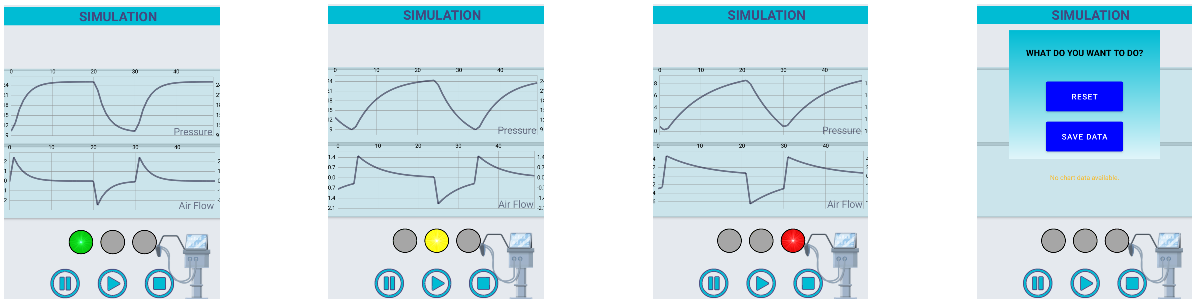
PEEP	PMAX	IE	RR		
5.0	20.0	0.5	20.0	✓	🗑

(d) Ventilation params selection from DB

Figure 5: Settings mode in the first prototype in Android

ferent buttons ("check" or trash bin icon). When both the patient's parameters and the ventilator settings are filled, the user can start the simulation by clicking the play button in the main window (see Fig. 4c).

Thus, `Ventilation App` starts simulating a patient with the desired conditions who is undergoing a mechanical ventilation procedure. During the simulation, at every breath, our app computes the WOB for the patient and, based on it, lights up one of the three semaphore lights, as explained in Sect. 3. If the measured effort is low and near the target one, the green light is shown, as in Fig. 6a. Instead, if the mea-



(a) Simulation with optimal settings (b) Simulation with non-optimal settings (c) Simulation with wrong settings (d) Stop screen

Figure 6: Simulation mode and Stop screen in the first prototype in Android

sured effort is higher but still acceptable, the yellow light is turned on, as in Fig. 6b. Finally, if the chosen ventilation strategy is beyond the safety limits or the measured effort is considered too high, the red light is shown (see Fig. 6c). Indeed, from Fig. 6a to Fig. 6c (where the chosen respiratory rate is too high and the pressure cannot reach the set maximum value) we can see that the pressure curve tends to become more flat, hence implying higher effort for the patient.

Having received the feedback on the chosen settings and analyzed the relevant patient's data through the pressure and flow curves, the user may decide to freeze the ventilation (e.g., to take a screenshot) with the pause button available in the main window (Fig. 4c). Finally, when the simulation is completed, the user may stop it by clicking on the stop button in the main window (Fig. 4c). In that case, the pop-up in Fig. 6d is shown. It lets the user choose whether to stop and reset the simulation or to save the current settings (both those of the patient and of the ventilator) in order to load them for following simulations.

5 RELATED WORK

As reported in Sect. 2.3 no other Android apps offering the same functionalities we offer with our training app exist, to the best of our knowledge. However, the effort spent by developers in designing and implementing simulation apps for physicians demonstrates the importance of such tools (Kunkler, 2006), especially in the most critical scenarios such as emergency medicine (Reznek, 2002). A survey conducted by the authors of (Wallace et al., 2012) reports that over 85% of the interviewed participants, including medical students and physicians, reported using a mobile-computing device for their training. This is the reason why we also decided to opt for a mobile app. Indeed, especially due to the increasing complexity and preciseness of patient models, simulators can be useful tools in determining a physician's understanding and use of best practices, management of patient complications, appropriate use of instruments and tools, and overall competence in performing procedures. Also in other medical domains beyond mechanical ventilation, simulators have been used, for example, to train doctors in the management of temporary pacemakers (Crowe et al., 2013), in bedside cardiac examinations (Takashina et al., 1997), in surgeries (Phé et al., 2016), in chronic kidney disease treatments (Markossian et al., 2021) and also during radiology procedures (L. E. Wood, 2018; Toderis et al., 2022). Using case-based simulation training software (as the app we propose in this paper) has been shown to improve physicians' skills even more than simple textbook reviews (Schwid, 2001; Couto et al., 2015).

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have shown the methodology adopted to develop the first prototype of a mobile application to make the training of new physicians in mechanical ventilation easier and more effective. It exploits the principles of the minimum WOB, described by the Otis equation, and used by ventilators implementing the ASV mode to give users feedback not only on the correctness of the set ventilation parameters, but also on their optimality, to reduce the effort required to patients when under mechanical ventilation. Starting from the discussion of the idea with pneumologists and the analysis of the already existing apps, we have first defined the requirements and the architecture, and then implemented the initial prototype presented in this paper. This prototype is just preliminary work that we have used to verify the feasibility of the idea and to stimulate discussion with experts in mechanical ventilation. In the future, we may conduct further analysis with physicians-in-training in order to refine `Ventilation App`, and to investigate potential ethical and legal implications of using it for medical education.

Thanks to the feedback received by the experts we collaborated with, we have identified some key future work. The first future work direction will be focused on having pre-defined scenarios in which the evolution of the patient status is described depending on the actions taken by the physicians. For now, it can be done only manually by saving into the database some relevant patient data and, then, changing them on-the-fly. Nevertheless, we believe that integrating some formalism automatically describing the patient's evolution depending on environmental conditions and ventilation choices, such as Markov Decision Processes (Lakkaraju and Rudin, 2016), or including an AI component, is feasible and worthwhile. Another future work direction is related to the expandability of the lung simulator on which our Android app is based. Indeed, it would be very appreciated by physicians to have different patient models, with different granularities, and modeling patients having different diseases. We believe that it can be easily done by exploiting the INSPIRE framework, which is on the basis of `Ventilation App` and supports the simulation of complex circuits. Finally, more ventilation modes beyond PCV can be implemented, in order to let new physicians train under disparate patient conditions.

ACKNOWLEDGEMENTS

This work has been partially funded by PNRR - ANTHEM (AdvaNced Technologies for Human-centrEd Medicine) - Grant PNC0000003 – CUP: B53C22006700001 - Spoke 1 - Pilot 1.4. We would like to thank Eleonora Vitali for the preliminary work done for this project.

REFERENCES

Abba, A., Accorsi, C., et al. (2021). The novel mechanical ventilator milano for the COVID-19 pandemic. *Physics of Fluids*, 33(3):037122.

Bombarda, A., Bonfanti, S., Galbiati, C., Gargantini, A., Pelliccione, P., Riccobene, E., and Wada, M. (2021). Lessons learned from the development of a mechanical ventilator for COVID-19. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE.

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.

Couto, T. B., Farhat, S. C., et al. (2015). High-fidelity simulation versus case-based discussion for teaching medical students in brazil about pediatric emergencies. *Clinics*, 70(6):393–399.

Crowe, M. E., Hayes, C. T., and Hassan, Z.-U. (2013). Using software-based simulation for resident physician training in the management of temporary pacemakers. *Simulation in Healthcare: The Journal of the Society for Simulation in Healthcare*, 8(2):109–113.

Fernández, J., Miguelena, D., Mulett, H., Godoy, J., and Martinón-Torres, F. (2013). Adaptive support ventilation: State of the art review. *Indian Journal of Critical Care Medicine*, 17(1):16–22.

IUCPQ (2023). VentilO Android App. https://play.google.com/store/apps/details?id=ca.qc.iucpq.ventilo.

Kunkler, K. (2006). The role of medical simulation: an overview. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2(3).

L. E. Wood, M. M. Picard, M. K. (2018). App review: The radiology assistant 2.0. *Journal of Digital Imaging*.

Lakkaraju, H. and Rudin, C. (2016). Learning cost-effective treatment regimes using markov decision processes.

Markossian, T. W., Boyda, J., et al. (2021). A mobile app to support self-management of chronic kidney disease: Development study. *JMIR Human Factors*, 8(4):e29197.

OpenPediatrics (2023). Openpediatrics. https://www.openpediatrics.org/simulators.

Pearce, J. M. (2020). A review of open source ventilators for COVID-19 and future pandemics. *F1000Research*, 9:218.

Phé, V., Cattarino, S., et al. (2016). Outcomes of a virtual-reality simulator-training programme on basic surgical skills in robot-assisted laparoscopic surgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 13(2):e1740.

Reznek, M. (2002). Virtual reality and simulation: Training the future emergency physician. *Academic Emergency Medicine*, 9(1):78–87.

Schwid, H. A. (2001). Components of an effective medical simulation software solution. *Simulation & Gaming*, 32(2):240–249.

Takashina, T., Shimizu, M., and Katayama, H. (1997). A new cardiology patient simulator. *Cardiology*, 88(5):408–413.

Toderis, L., Vo, A., et al. (2022). Development of a mobile training app to assist radiographers' diagnostic assessments. *Health Informatics Journal*, 28(2).

TruCorp (2023). Truvent app. https://trucorp.com/product/truvent-app/.

van Diepen, A., Bakkes, T. H. G. F., et al. (2021). A model-based approach to generating annotated pressure support waveforms. *Journal of Clinical Monitoring and Computing*.

Ventsim (2023). Ventsim. https://ventsim.cc/#/.

Wallace, S., Clark, M., and White, J. (2012). 'it's on my iPhone': attitudes to the use of mobile computing devices in medical education, a mixed-methods study. *BMJ Open*, 2(4):e001099.