

MedECC Telemedicine App and Universal Ventilator Controller

Bryant Liu and Jeremy Ford
UCSD CSE145 / CSE237d
Spring 2020

1. Abstract

COVID-19 is an ongoing global pandemic which is straining global medical systems. Previous development of cheap and easy-to-manufacture makeshift ventilators has helped to address mechanical ventilator shortages, however there is still a lack of frontline healthcare professionals. Many are not situated where they are most needed, so we wish to remove the location barrier with a telemedicine mobile application. Our mobile application enables a larger number of healthcare workers to remotely and efficiently monitor their patients as well as provide a standardized control interface for makeshift ventilators. Frontline healthcare workers can assign caregivers to remotely monitor patients and alert when needed, therefore distributing the workload. For scalability and ease of development the application was ported to a native app from a web app, which was written in Angular for frontend and C# for the backend. Our application will help to relieve the pressures placed on healthcare providers and will help to ensure the safety of makeshift ventilators.

2. Introduction

The first reported case of the novel coronavirus disease 2019 (COVID-19) was announced on December 31, 2019, in Wuhan, China. Two weeks later Chinese reported the first COVID-19 death. The following week brought reports of the first confirmed case in the United States. By March 2020, COVID-19 had rapidly spread to nearly every country and was declared a global pandemic by the World Health Organization [1]. Now, six months following the initial reporting of COVID-19, there more than 7,400,000 global cases and 400,000 global deaths, with more than a quarter of the cases and deaths occurring in the United States [2]. This pandemic has placed an unprecedented strain on both rural and urban healthcare systems, and has overloaded testing capacities, intensive care beds, medical equipment and supplies, and healthcare workforces [3,4].

Governments, organizations, and individuals around the world have stepped forward in response to the COVID-19 crisis. Governments have opened their stockpile of emergency supplies to aid in the recovery, but this stockpile has proven insufficient [5]. Many of the world's largest manufacturers have shifted their production toward medical devices and medical supplies, which has helped to alleviate some of the shortage [6]. Smaller groups and individuals have been contributing to open-source medical devices, such as ventilators which can be quickly and cheaply constructed from commonly available materials [7].

As the number of COVID-19 cases in the United States and globally continues to grow there may be a growing demand for additional ventilators [8]. Open-source makeshift ventilators may be an invaluable resource in treating patients. One drawback of the makeshift ventilator designs that are being proposed is a wide-variety of mechanisms and control systems in place. Some ventilators have only very rudimentary control, such as a single knob used to adjust the rate at which a motor pumps an ambu-bag type respirator [9]. Other ventilators are controlled by much more sophisticated control systems, similar to those implemented in FDA-cleared ventilators currently used in hospitals. The lack of a standardized control interface poses a safety risk with the use of these devices.

The highly contagious nature of COVID-19 places physicians, nurses, and other health care staff, who have close contact with infected patients, at an increased risk of the disease. Additionally, nearly a quarter of healthcare workers are above the age of 55, making them especially at risk of complications related to COVID-19 [10]. Providing remote-care capability may help to reduce the risk to these healthcare workers. As well, a telemedicine application may help to ease the burden on healthcare workers by allowing volunteers to assist with bedside care of patients, freeing physicians and nurses to focus on the most critical cases.

The Medical Electronic Control Center (MedECC) mobile telemedicine application is designed to address the concerns with ventilator controls as well as the risk to healthcare workers. The MedECC project is a web application that provides healthcare workers and volunteers with a simple-to-use interface to track and manage their patient populations, and a universal ventilator controller that interfaces with the plethora of makeshift ventilators being designed by teams around the globe.

3. Technical Material

3.1. Unit and Regression Testing on Arduino Hardware

The Arduino Due with 32-bit ARM Cortex-3 has been selected as the target hardware for ventilator control. Communication occurs between the user's cell phone and the Arduino Due via a standard 1/8" barrel connector. The Arduino Due takes incoming commands and actuates some control on the target ventilator. The software and firmware for this system is being developed in house by our hardware development team.

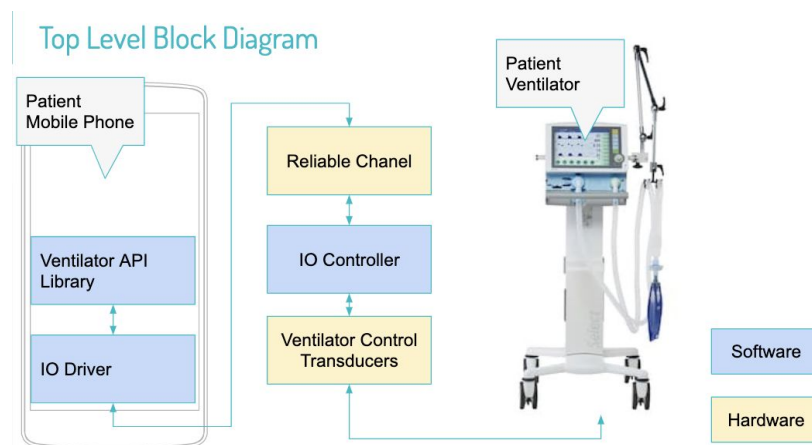


Figure 1. Top Level Block Diagram of Ventilator Controller

The Food and Drug Administration (FDA) requires extensive verification and validation for all software that is used to automate any part of a medical device [11]. The universal ventilator controller portion of MedECC must comply with this regulation. As such, it was necessary for the ventilator control protocol and implementation to be thoroughly tested, and integrated with a unit testing framework.

Unit testing is common in many software development systems, but is less commonly used with embedded systems [12]. In many embedded applications, unit testing of code is performed on a workstation - the code is written, compiled, and tested on a workstation computer and then the validated code is compiled onto the target hardware. This method of testing does not meet the FDA's requirements. In order to achieve unit testing on the Arduino hardware, we have implemented the AUnit library and the Arduino Real-Time extension.

3.2. Telemedicine Web App Development

The telemedicine application consists of many parts, the biggest ones are the frontend and the backend. The frontend of the application consists of Angular [13] and the backend of the application is built with a Dotnet server[14]. Angular is a popular type-scripted web framework developed by Google in 2016, it is commonly used in industry just like React. Angular uses a hierarchy of components as its primary architecture which makes the application relatively modular compared to other solutions. Angular is written in typescript, which is an extension of Javascript. Some Angular applications are: Youtube, Gmail, Paypal, etc. Dotnet server is the chosen backend solution to our application, it is a server side application framework built and maintained by Microsoft and written with C#. Its main function is to allow websites to be dynamically updated.

Angular Material[15] is used to aid development of a modern and consistent Angular application; it is an Angular UI component library that is also built by Google. Angular Material allows faster development of applications as it provides many of the component's styling and hierarchy, such as lists, forms, buttons, etc. We have incorporated the use of Angular Material in almost every page of our application to improve styling consistency and shorten development time.

Within our application, we have a patient live stream feature, which is handled by Daily.co[16] on the backend. Since it is not feasible for us to create a scalable live streaming service due to lack of hardware and fiscal resources. Daily.co is an easy to use API for live video chat, it also scales automatically, which helps us to be able to handle any load/traffic.

For the design aspect of the web application, Figma[17] is used by our designers to provide a clear visual for our developers to implement. Figma is a web-based design and prototyping tool, built with vector based contents. Users can use and view it collaboratively like google docs. All of our page views are designed on Figma then programmatically implemented in Angular. Figma makes it easy for our developers to implement designs as all the positioning and colors are automatically generated, and can then be copy and pasted into the code.

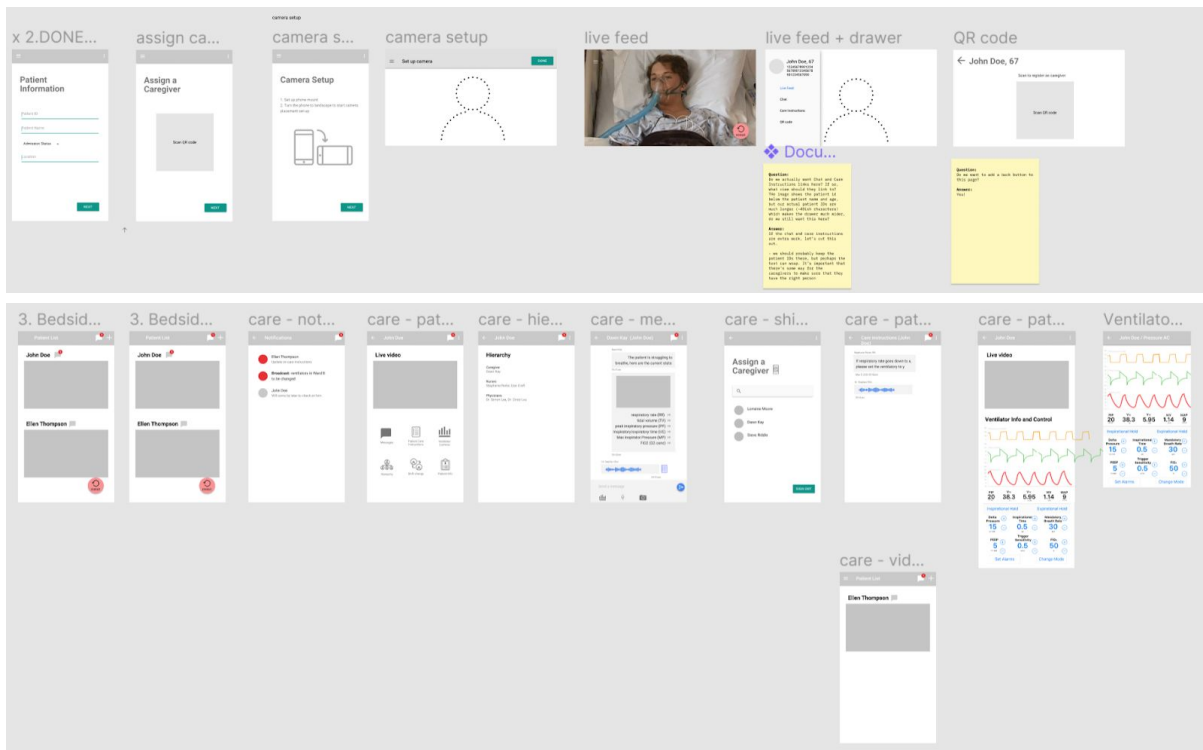


Figure 2. Examples of Figma designs for the web application

As Angular builds all parts of the application as components, routing[18] is necessary to connect each component together. Imagine building a house, Angular components are the bricks and routing is the mortar. Routing in Angular works essentially the same as the one in a web browser, allowing the user to navigate through components as well as querying data ubiquitously. Routing allows our application to pass crucial information reliability and securely across pages, such as name and ids of patients and caregivers. The Dotnet server uses the routing to communicate data between what the user sees and the database. Routing also lets the application to hook on Daily.co, making the patient live stream possible.

Integration between different developers is done through the use of git[19] version control through github. As each developer is working on a different part of the project, it only makes sense that everyone can work simultaneously and without breaking the other's code. Git allows developers to easily share code efficiently and provide a medium for collaboration. Github contains a hierarchical structure for code approval, new changes that are meant to be incorporated into the code base must be approved by other developers through a pull request, which works like a ticketing system. Code under the pull request can be commented by authorized approvers and edits can be made by the author to satisfy those comments, afterwards the code may be merged to the master code base. Git is a standard version control that is used ubiquitously by programmers.

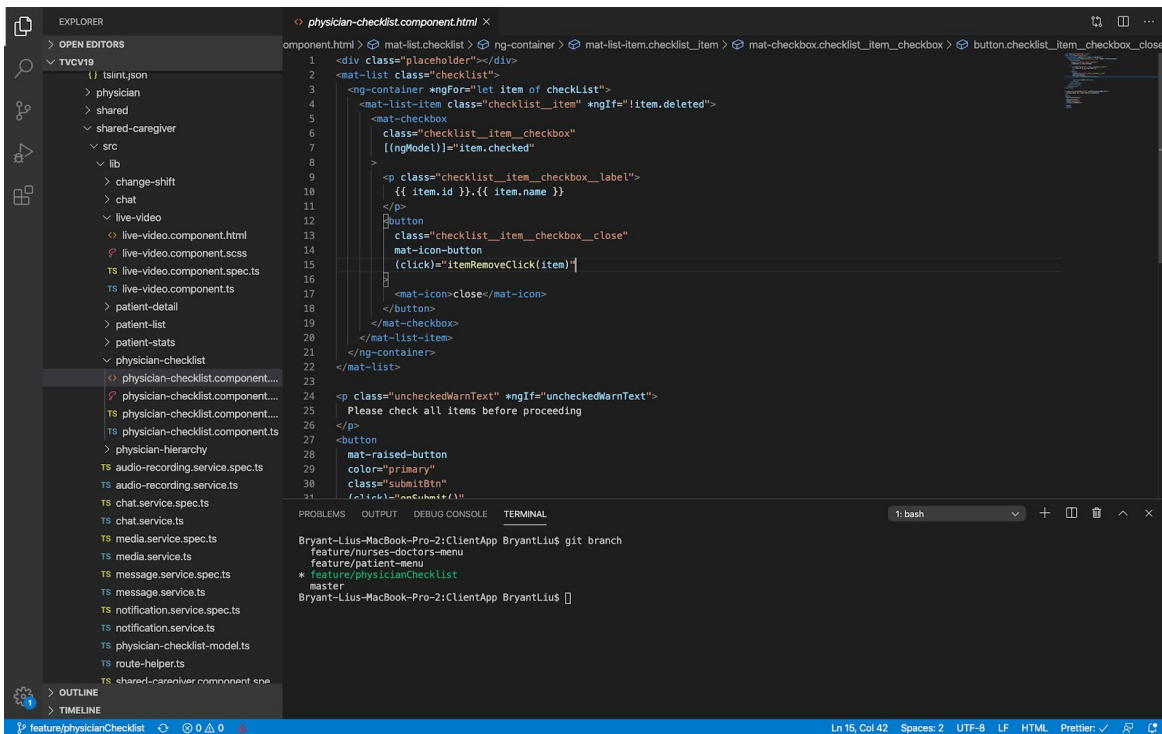


Figure 3. Development Environment of Angular code with git branching on the bottom

In the future the web application is planned to be ported over to a native application[20] for both iOS and Android. Our goal is for the telemedicine application to not only be used for the current COVID crisis but also for areas with a shortage of caregivers and dire times such as natural disasters, pandemics, etc.

4. Milestones

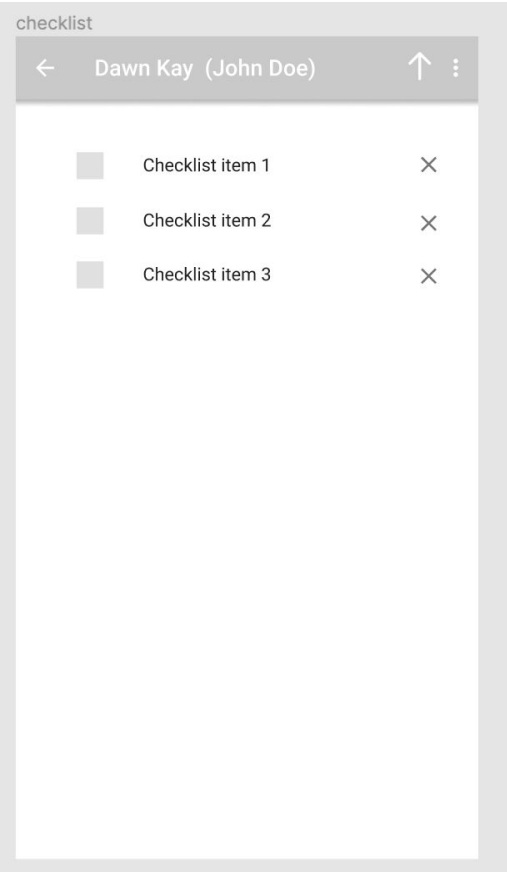
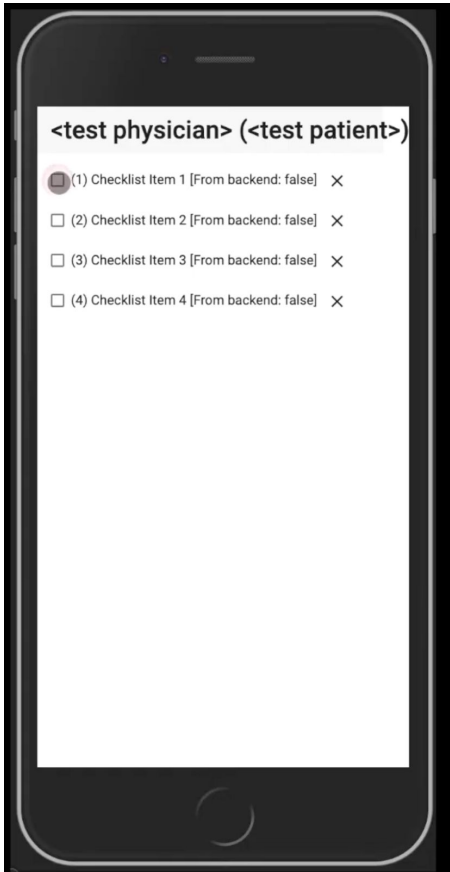
This section addresses our achievements with this project, as well as milestones which were not completed, and problems we faced while working on the project.

4.1. Completed Milestones

4.1.1. Complete First Angular Page View (Milestone 1)

Physician Checklist (version 1):

Live Demo: <https://www.youtube.com/watch?v=oltIGrrKvvM>

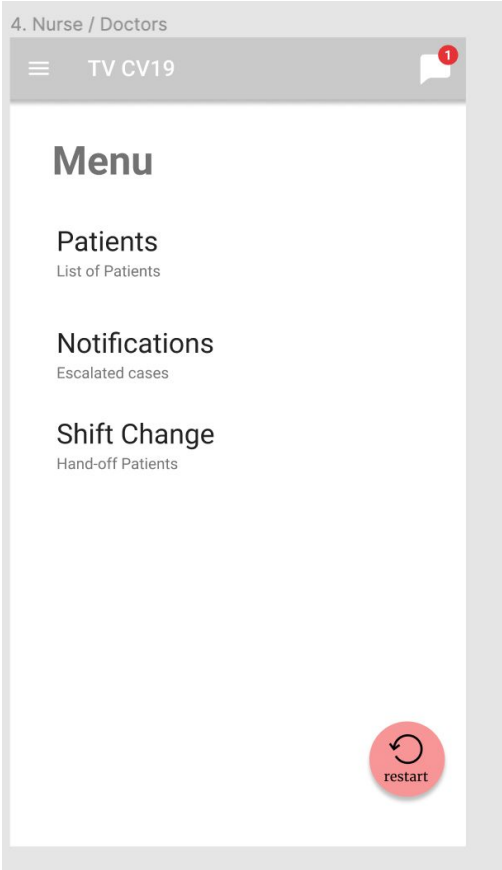
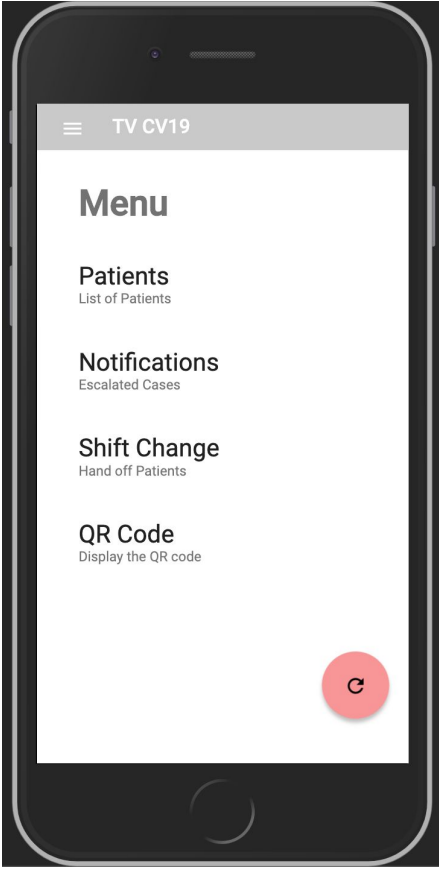
Figma Requirement	Implementation (version 1)
	

This page has services set up for checklist items, as well as modules for items on the checklist. All of the information on the page is templated, which can be substituted at any time. This includes functionality to remove checklist items.

4.1.2. Complete Two Additional Page Views (Milestone 2)



Carer(Nurses/Doctor) Menu:

Live Demo: <https://youtu.be/yd2dO4V9jkk>.

Figma Requirement	Implementation
	

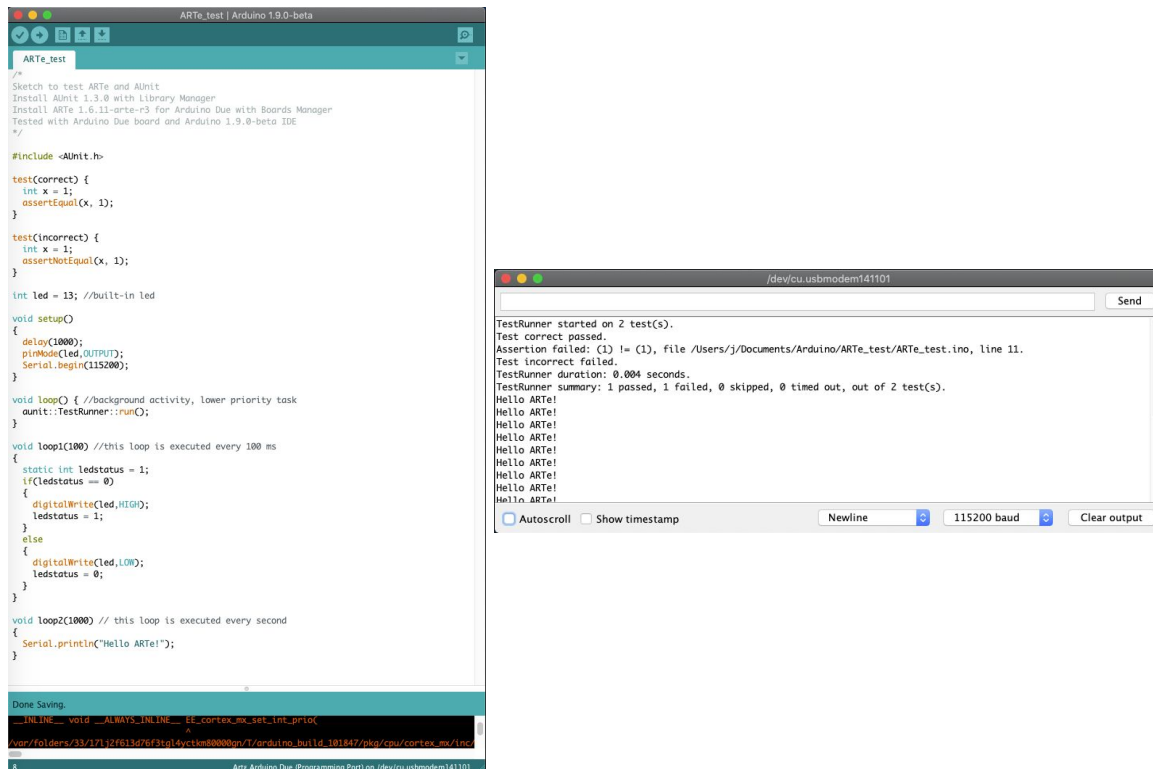
This page view is created for caregivers to have fast and clear access to all of their important information. The caregivers are able to see their list of patients as well see the notifications about them. Shift changing is also here for caregivers to change their shifts. The caregiver's own QR code is also here for others to scan. The routing for these links are completed.

Patient Age:

<h3>Patient Information</h3> <p>Patient Name *</p> <p>Location *</p> <p>Age *</p> <p>NEXT</p>	<h3>Patient List</h3> <p>Rick Sanchez Age: 73</p>  <p>Ric Flair Age: 65</p> 
---	---

This is the added age parameter for patient information when admitting the patient, patient's age is now available for doctors, nurses and caregivers to see in order to prescribe a more accurate and relevant diagnostic and treatment plan

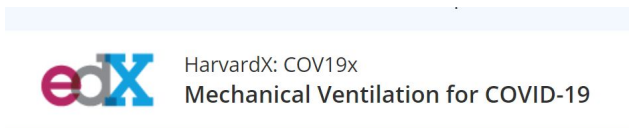
4.1.3. Integrate Unit Test Framework with Arduino Extension ARTE (Milestone 3)



These are screenshots of the Arduino IDE showing an example unit test and output.

4.1.4. Complete Three Additional Angular Page Views (Milestone 4)

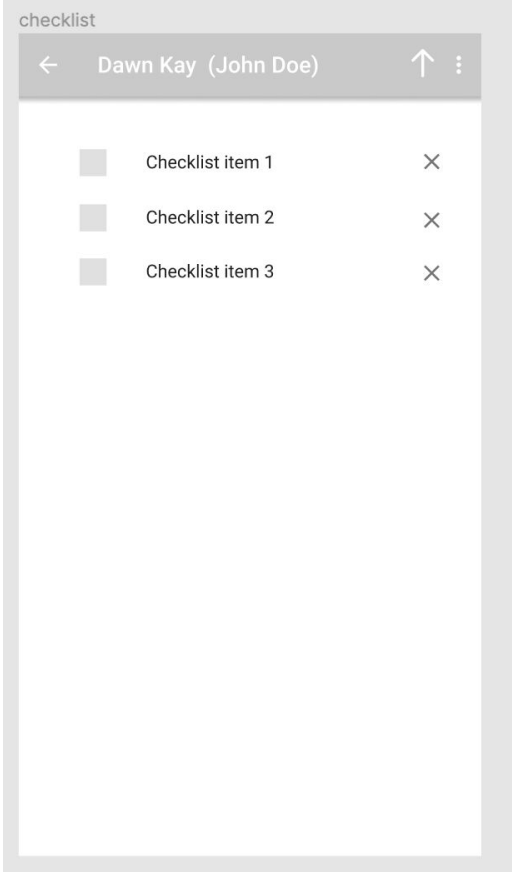
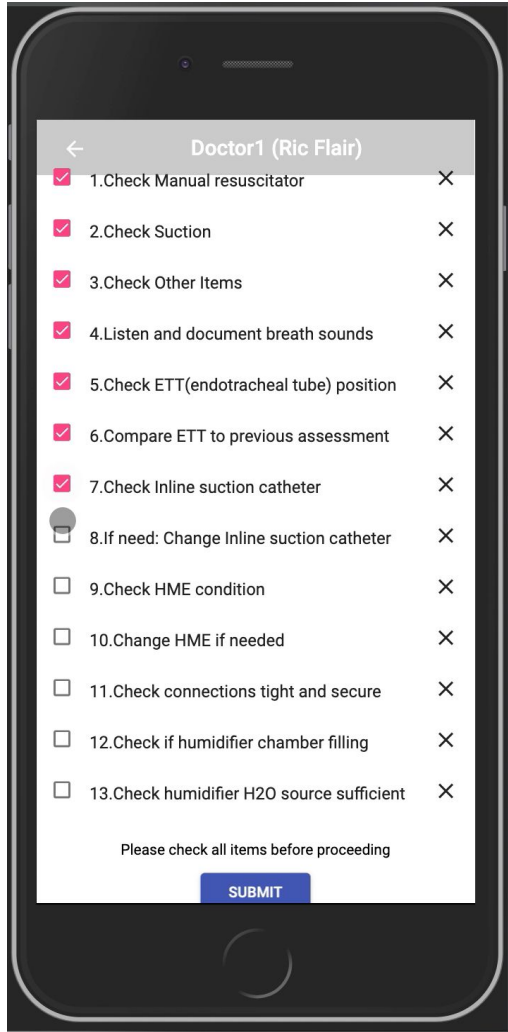
Caring for COVID Patients Course:



- > Introduction to the Course
- > Basic Introduction to Mechanical Ventilation
- ✓ Advanced Topics in Mechanical Ventilation
 - Setting the Ventilator (25 min)
 - Ventilation in ARDS (30 min)
 - Ventilation in Obstructive Lung Disease (20 min)
 - Waveform Analysis (30 min)
 - Feedback
- ✓ Mechanical Ventilation in COVID-19 Patients
 - COVID-19 (20 min)

This is a HarvardX course regarding ventilation for COVID patients, which we took, so we will know what to put inside the checklist page that has been created. We documented some checklists to be shown to doctors for improvements and validation. This step took a very long time, so we decided to use it in place as a page view in the milestone.



Updated Patient Checklist:

Figma Requirement	Implementation
 <p>The Figma design shows a checklist titled 'checklist' for 'Dawn Kay (John Doe)'. It features a header with a back arrow, the patient name, an up arrow, and a menu icon. Below the header are three checklist items, each with a grey square checkbox and an 'X' mark on the right.</p>	 <p>The mobile app implementation shows a checklist for 'Doctor1 (Ric Flair)'. The header includes a back arrow, the doctor name, and a menu icon. The checklist contains 13 items, each with a checkbox and an 'X' mark. The first three items are checked with red checkmarks. Item 8 has a grey circle next to its checkbox. At the bottom, there is a blue 'SUBMIT' button and a message: 'Please check all items before proceeding'.</p>

We built a new checklist page, based on change in scope and new instructions. The previous checklist page was completely redone. The checklist items are designed in a modular way, which allows the creation of nested checklist items if needed. In the new checklist, the doctor and patient name is auto updated to correspond to the patient the checklist is for and that patient's caregiver.

Live demo: <https://youtu.be/LV6ZMcSNk00>

Patient Live Stream:

Figma Requirement	Implementation
	

Building a patient live stream and allowing it to be full screen. This page uses the Daily.co API to show the live video.

Live demo: <https://youtu.be/VV7E9x2Q6Lc>

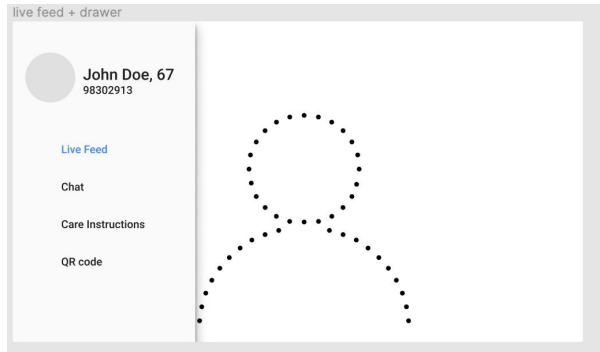
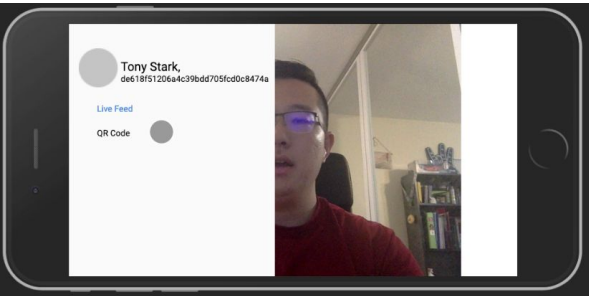
4.1.5. Complete Three Additional Angular Page Views (Milestone 5)

Patient GUID:



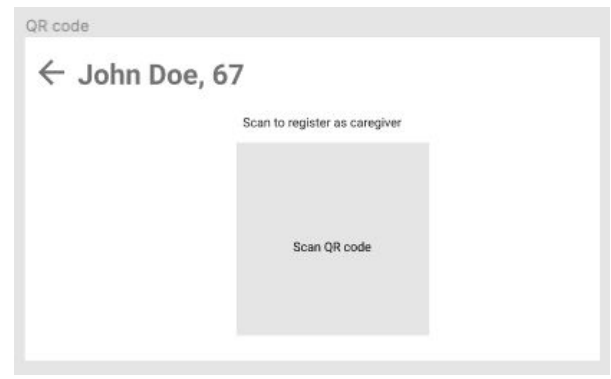

We've implemented GUID, which generates a new unique ID for each patient, preventing attempts to create patient rooms under the same ID. This patient id will appear in the patient menu. This is a backend feature that removed the issues we had before regarding patients with the same id on the backend.

Patient Menu:

Figma Requirement	Implementation
	

This is the patient menu, which contains the details of the patient during the live stream. The live feed will close the menu and show the live stream again. The QR code will lead to the patient's QR code allowing it to be scanned. The patient name and id are both properly linked up with the backend and updated to the patient.

Patient QR Code:

Figma Requirement	Implementation
	

This page provides a quick link to the patient's QR code from the patient menu, allowing new caregivers to be added.

4.2. Incomplete Milestones

4.2.1. Integrate Angular/webapp code with native code

This milestone was removed from the project because the Angular/Webapp code was not in a state that was ready for integration with native code. We decided to focus on the Angular/webapp code, so that it could later be integrated with native code.

4.2.2. Integrate Raspberry Pi based makeshift ventilator into existing app framework

This milestone was removed from the project because it was determined that the team had an excess of embedded system developers and the web development was lagging. Additionally, the hardware requirements are continually evolving, and there is a non-trivial lead-time associated with getting new components, making scheduling of these tasks difficult. We decided to focus on Angular/Webapp development.

4.3. Problems Faced

There were many problems faced while working on this project. First, and foremost, working on an embedded project while under social distancing, as required due to COVID was a challenge on its own. Additionally, we originally planned to focus more on the ventilator controls rather than the telemedicine app, but the infrastructure was not in place. Focusing on the Angular application required us to learn Angular and Typescript, which was a time consuming task. Had we not been working remotely, and had we had previous experience with Angular, we likely could have achieved much more than we ultimately did.

5. Conclusion

As the current global pandemic of COVID-19 drags on, the demand on global medical systems will exceed supply in many areas. We believe our MedECC project will widen the support for healthcare workers in need. Our telemedicine application allows frontline healthcare workers to be able to assign

caregivers to remotely monitor patients and alert help when needed, therefore distributing the workload. We have developed the telemedicine application in modular languages such as Angular and DotNet, thus allowing our developers the freedom to work independently as well as easily incorporating more help.

In terms of the future, this pandemic showed us how many aspects of our society can be done virtually, we believe the telemedicine application can be used for healthcare systems in areas with a shortage of healthcare workers and other crises such as natural disasters, pandemics, etc. This will reduce response time as well as lower capital required by charity efforts to help with disasters.

References

1. World Health Organization. Rolling Updates on Coronavirus Disease (COVID-19). 01 Jun 2020. <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/events-as-they-happen>
2. Worldometer. COVID-19 Coronavirus Pandemic. 10 Jun 2020. <https://www.worldometers.info/coronavirus/>
3. PEW Charitable Trust. Coronavirus Threatens Strained Rural Health Care System. 17 Mar 2020. <https://www.pewtrusts.org/en/research-and-analysis/blogs/stateline/2020/03/17/coronavirus-threatens-strained-rural-health-care-system>
4. N. Smith, M. Fraser. American Journal of Public Health. Straining the System: Novel Coronavirus (COVID-19) and Preparedness for Concomitant Disasters. 08 Apr 2020. <https://ajph.aphapublications.org/doi/full/10.2105/AJPH.2020.305618>
5. The Center for Public Integrity. The Government's Secret Ventilator Stockpile is Nowhere Near Enough to Fight the Coronavirus. 24 March 2020. <https://publicintegrity.org/health/coronavirus-and-inequality/the-governments-secret-ventilator-stockpile-is-nowhere-near-enough-to-fight-the-coronavirus/>
6. E. Modic. Today's Medical Developments. Manufacturing: Producing, Design, delivering, Helping During COVID-19. <https://www.todaysmedicaldevelopments.com/article/covid-19-manufacturing-medical-device-capacity-capabilities/>
7. J. Pearce. F1000 Research. A Review of Open-Source Ventilators for COVID-19 and Future Pandemics. 20 Apr 2020 <https://f1000research.com/articles/9-218>
8. M. Ranney, V. Griffeth, & A. Jha. New England Journal of Medicine. The Need for Ventilators and Personal Protective Equipment During the COVID-19 Pandemic. 30 Apr 2020. <https://www.nejm.org/doi/full/10.1056/NEJMp2006141>
9. L. Teschler. Test & Measurement Tips. More Dangerous than the Virus? Converting Manual Resuscitators to Ventilators. 30 Mar 2020. <https://www.testandmeasurementtips.com/more-dangerous-than-the-virus-converting-manual-resuscitators-to-ventilators/>
10. H. Ehrlich, M. McKenney, & A. Elkbuli. The American Journal of Emergency Medicine. Protecting our Healthcare Workers during the COVID-19 Pandemic. 17 Apr. 2020. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7162741/>
11. Food and Drug Administration. General Principles of Software Validation; Final Guidance for Industry and FDA Staff. 22 Jan 2002. <https://www.fda.gov/media/73141/download>

12. Parasoft. Practical Unit Testing for Embedded Systems - Part 1. 2010.
<http://www.public.asu.edu/~atrow/ser456/articles/PracticalUnitTesting.pdf>
13. Google. Angular. 2020.
<https://angular.io/>
14. Microsoft. ASP.NET. 2020.
dotnet.microsoft.com/apps/aspnet
15. Tutorials Point. Angular Material Tutorial, 2020.
www.tutorialspoint.com/angular_material/index.htm
16. "Daily.Co | 1-Click Video Chat API." Daily.Co, 2020,
www.daily.co
17. Gonzalez, Robbie. "Figma Wants Designers to Collaborate Google-Docs Style." Wired, 26 July 2017.
www.wired.com/story/figma-updates
18. Ahmed. "A Complete Guide To Routing In Angular." Smashing Magazine, 28 Nov. 2018.
www.smashingmagazine.com/2018/11/a-complete-guide-to-routing-in-angular
19. "Git." Git, 2020.
<https://git-scm.com/>
20. Rouse, Margaret. "Native App." SearchSoftwareQuality, 29 Mar. 2018.
<https://searchsoftwarequality.techtarget.com/definition/native-application-native-app#:~:text=A%20native%20application%20is%20a,device%2Dspecific%20hardware%20and%20software>