AED-System

Run Instructions

```
cd src/
make all
```

Requirements

Linux:

QtSvg library

```
sudo apt install libqt5svg5-dev
```

---

macOS:

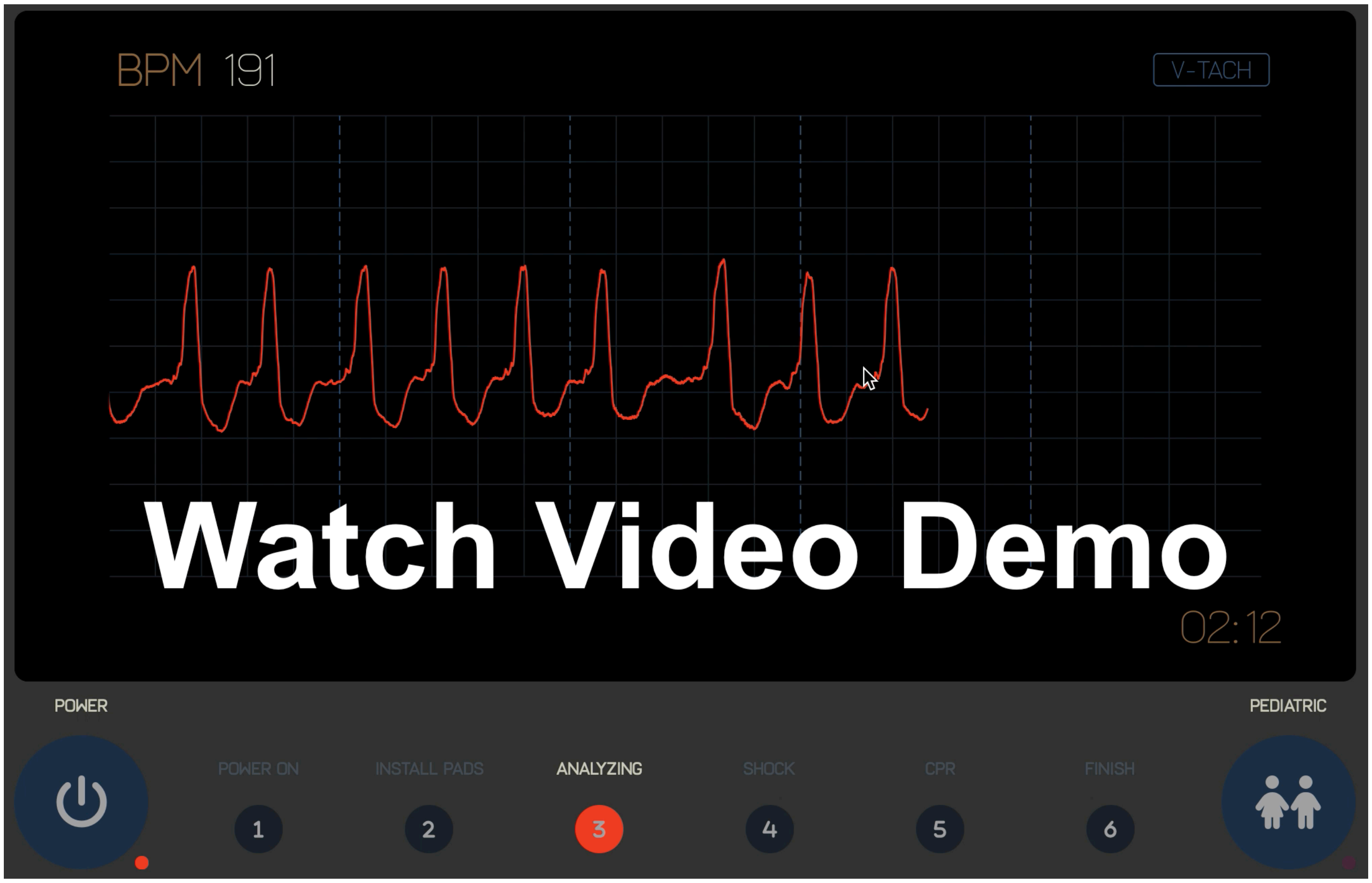
Qt5 framework

```
brew install qt@5
```

File directory

```
AED-System/
├── README.md
├── graphics/
│   ├── layers.png
│   ├── screen_artboard.png
│   ├── screen_wireframe.png
│   └── screens.ai # Source file used to create the SVG screen.svg file
├── assets/
│   └── fonts
│       ├── fa-solid-900.ttf # Icon font
│       └── frank-light.ttf
└── src/
    ├── AEDController.cpp
    ├── AEDController.h # Controller class for all the different stages
    ├── Makefile
    ├── Pads.cpp
    ├── Pads.h # Provides sensor data to controller for rhythm, heart rate and more
    ├── QAEDScreen.cpp
    ├── QAEDScreen.h # Extends the SVG class, provides method to easily use screen
    ├── QCustomIconsFont.h # Imports custom font
    ├── QIconButton.cpp
    ├── QIconButton.h # Extends QPushButton but uses font icon
    ├── QScreenSettings.cpp
    ├── QScreenSettings.h # Admin panel class, controls Pads sensor data
    ├── QSvgWidget.cpp
    ├── QSvgWidget.h # Directly manipulates svg files to show/hide their elements
    ├── Stage1.cpp
    ├── Stage1.h # Initializes AED, shows initial instructions
    ├── Stage2.cpp
    ├── Stage2.h # Pad installation and instructions
    ├── Stage3.cpp
    ├── Stage3.h # Heart rhythm analysis and ECG
    ├── Stage4.cpp
    ├── Stage4.h # Shocks patient
    ├── Stage5.cpp
    ├── Stage5.h # Administers and guides user through CPR
    ├── Stage6.cpp
    ├── Stage6.h # Turns off AED
    ├── StageManager.cpp
    ├── StageManager.h # Abstract class that's extended by Stage1-6 classes
    ├── Team17AED.pro # Qt project config file
    ├── aed_stages.h # enums with all the different Stages and their states
    ├── defs.h # global defines/settings
    ├── main.cpp # Sets up the GUI, main window and runs the application
    └── screen.svg # SVG file containing all the graphics for the screen
```

Video Demo & Testing



https://youtu.be/-twgo3yUh5U

Design Patterns

Overview

This was created using CPP with the Qt framework. It is an AED device that simulates the functionality of an automated electronic defibrillator device.

We have independent stage manager objects which manage the functionality of each stage as well as individual objects such as a debugging menu and a GUI to reflect correct feedback throughout the stages of using an AED. On the back end all stage `Stage1 - Stage6` classes are managed by `AEDController` where each stage is on its own thread, making it a concurrent system.

OOP Design Patterns

Observer

It implements the Observer state design which allows a one to many dependency relationship when one object changing updates the rest of the system.

For example in our project the status of pad attachment will change variables which are recognized by multiple other objects which in turn have their own functions responding to the status of pad attachment.

This allows us to accurately simulate the effect of certain variables which influence core functionality of the AED such as pediatric/adult patient, pad attachment status, heart beat analysis result etc.

State

We also implement a State design pattern which modifies the functions of our objects as there state changes. For example our stage objects will perform safety checks based on the status of certain elements such as if the patient is being touched by the rescuer, which will cause the shock in stage 4 to be delayed until they are not being touched. As specific states during AED defibrillation can interfere with resuscitation this is an essential feature to allow objects to verify the correctness of all states before it can go through with the necessary functions, as well as vice versa with AED objects pausing functions or outputting error/ instruction messages if the state is not correct.

Controller

Our program implements the Controller design pattern as well using the AEDcontroller object which acts as a centralized control for the rest of the objects in our source code.

While the other objects oversee the individual segments of the AED use case the AEDcontroller is responsible for the creation of an interface which unifies these individual stages. It has functions which are responsible for creating the interface as well as switching between stages.

Use cases

Use Case 1: Power on the AED for use

Primary actor:

- User

Preconditions:

- The AED is turned off

Main success scenario:

1. User presses the power button
2. AED turns on
3. AED performs a self test
4. AED displays “UNIT OK” message
5. AED sends voice prompt “UNIT OK”

Postconditions:

- The device is turned on and ready to use

Extensions:

- **2a**. AED does not turn on

- **2a1**. The user must replace all batteries at the same time. If the device does not turn on after, remove it from service
- **4a**. AED screen displays and voice prompts the user with “CHANGE BATTERIES” message
- **4a1**. The user must replace all 10 batteries and press the button in the battery well only after installation of new batteries
- **4b**. AED screen displays and voice prompts the user with “UNIT FAILED” message indicating the device has failed its power up self test and is not usable for victim care
- **4b1**. The user performs a manual test by pressing and holding the Power button for more than 5 seconds. If the unit fails the test again, remove it from service

---

Use Case 2: User Places the Electrodes

Primary actor:

- User

Preconditions:

- The AED is power on and ready for use

Main success scenario:

1. AED displays and voice prompts user with “STAY CALM” message
2. AED displays and voice prompts user with “CHECK RESPONSIVENESS” message while the related graphic indicator light flashes
3. User shakes the victim and asks if they are “OK?”
4. AED displays and voice prompts user with “CALL FOR HELP” message while the related graphic indicator light flashes
5. User calls someone for help
6. AED displays and voice prompts user with “OPEN AIRWAYS” message
7. User opens patients airways
8. AED displays and voice prompts user with “CHECK FOR BREATHING” message
9. User checks to see if patient is breathing
10. AED displays and voice prompts user with “EXPOSE BARE CHEST” messages
11. AED displays and voice prompts user with “ATTACH DEFIB PADS TO PATIENT’S BARE CHEST” messages while the related graphic indicator light flashes
12. User selects the appropriate set of electrode pads (Adult or Child) and connects the electrode cable with the electrode connector
13. Places the pads on the patient's bare chest as guided by the electrode pads package instructions

14. AED displays and voice prompts user with “ADULT PADS” or “PEDIATRIC PADS”, depend on which ones it detects are connected and adjusts defibrillation energy settings accordingly

Postconditions:

- The electrode pads are set properly and ready for use

Extensions:

- 14a. AED displays and voice prompts user with “CHECK ELECTRODE PADS” message
- 14a1. The user rechecks the connection and attempts to reattach the electrode pads

---

Use Case 3: AED Performs Heart Rhythm Analysis

Primary actor:

- User

Preconditions:

- The electrode pads are set properly and not defective

Main success scenario:

1. User selects the Heart Rhythm Analysis mode
2. AED displays and voice prompts user with “DON’T TOUCH PATIENT, ANALYZING” message while the related graph’s indicator light flashes
3. AED starts evaluating the heart rhythm
4. AED completes the heart rhythm analysis
5. AED displays its analyze advice

Postconditions:

- A heart rhythm analysis is made

Extensions:

- 3a. AED screen displays and voice prompts the user with “ANALYSIS HALTED. KEEP PATIENT STILL” message
- 3a1. User stops any ongoing CPR and keeps the victim as motionless as possible
- 5a. AED displays and voice prompts user with "SHOCK ADVISED" message if a shockable rhythm is detected (ventricular fibrillation or ventricular tachycardia)
- 5a1. AED continues to shock delivery stage
- 5b. AED displays and voice prompts user with “NO SHOCK ADVISED” message if a rhythm that is not treatable by defibrillation is detected

[illegible]

- 5b1. AED continues to CPR stage

Use Case 4: AED Performs Shock Delivery

Primary actor:

- User

Preconditions:

- A shock is advised by the AED device

Main success scenario:

1. AED displays and voice prompts user with “STAND CLEAR” message to ensure no one is touching the patient
2. User presses the shock button as guided by the on-screen instructions
3. AED displays and voice prompts user with “SHOCK WILL BE DELIVERED IN THREE (TWO), (ONE)” while the red heart’s indicator light flashes
4. AED delivers the shock
5. AED displays and voice prompts user with “SHOCK DELIVERED” message

Postconditions:

- A shock is delivered to the victim

Extensions:

- 4a. AED displays and voice prompts user with “NO SHOCK DELIVERED” message
- 4a1. An error condition was detected, no shock was delivered
- 5a. AED displays and voice prompts user with “n SHOCKS DELIVERED” message
- 5a1. A total of “n” shocks have been delivered since the Fully Automatic AED Plus was turned on

---

Use Case 5: CPR

Primary actor:

- User

Preconditions:

- The AED advises no shock needed or a shock is delivered

Main success scenario:

1. AED screen displays and voice prompts the user with “START CPR” message while the related graph’s indicator light flashes
2. [illegible]

2. User follows on-screen guidance for chest compressions (2 breaths for every 30 compressions)
3. AED monitors and provides real-time CPR feedback
4. After 2 minutes of CPR, the AED screen displays and voice prompts the user with “STOP CPR” message

Postconditions:

- Successfully performed CPR on the victim

Extensions:

- 3a. AED screen displays and voice prompts the user with "CONTINUE CPR" message if CPR is needed or the device fails to detect chest compressions >= ¾ of an inch deep
- 3a1. User continues to administer CPR to patient with same compression level
- 3b. AED screen displays and voice prompts the user with "PUSH HARDER" message if the device detects CPR compressions are consistently < 2 inches deep
- 3b1. User administers CPR with higher compression level
- 3b2. AED screen displays and voice prompts the user with "GOOD COMPRESSIONS” message if after showing to push harder, user has delivered chest compressions >= 2 inches deep

---

Use Case 6: Power off

Primary actor:

- User

Preconditions:

- The AED is on

Main success scenario:

1. User presses the power button
2. The screen turned off

Postconditions:

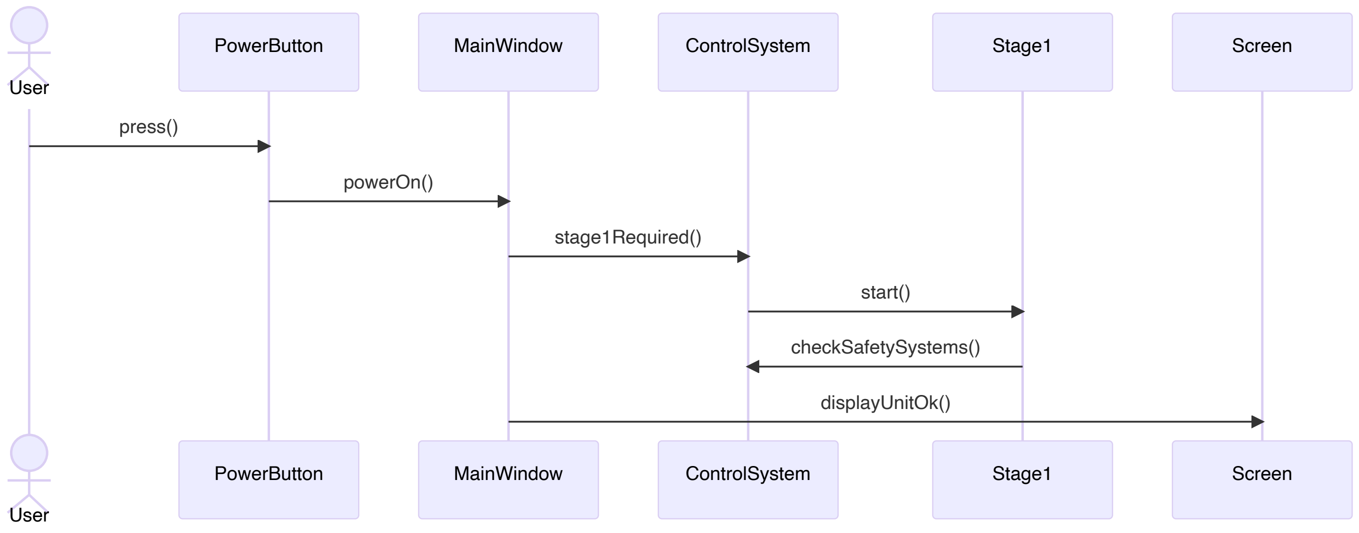
- The AED is turned off

Diagrams

Sequence Diagrams

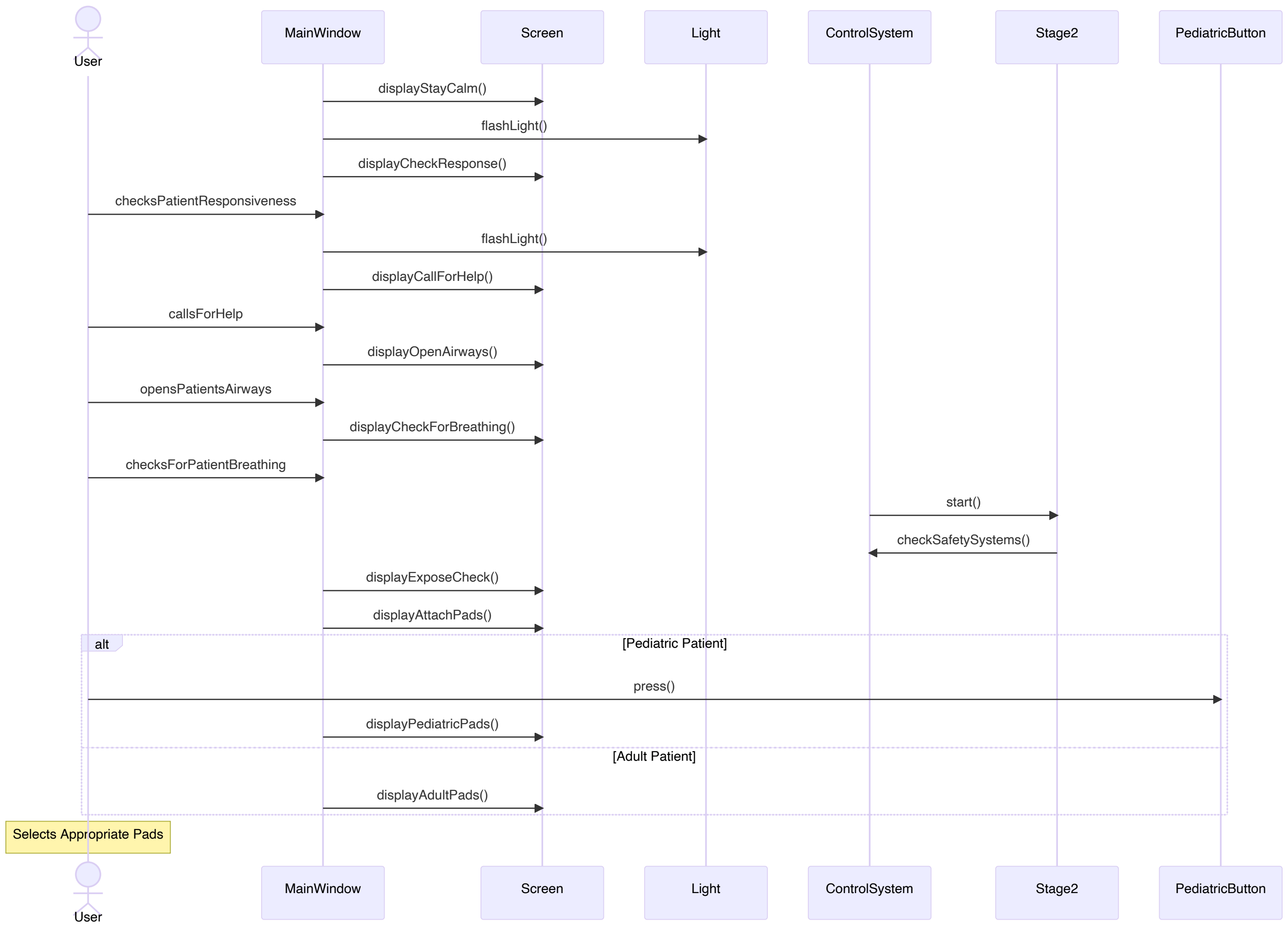
UC1: Power On AED

This shows the case of the user truning on the AED. The user presses the power button, then the system checks to make sure the safety conditions are met and displays the "UNIT OKAY".



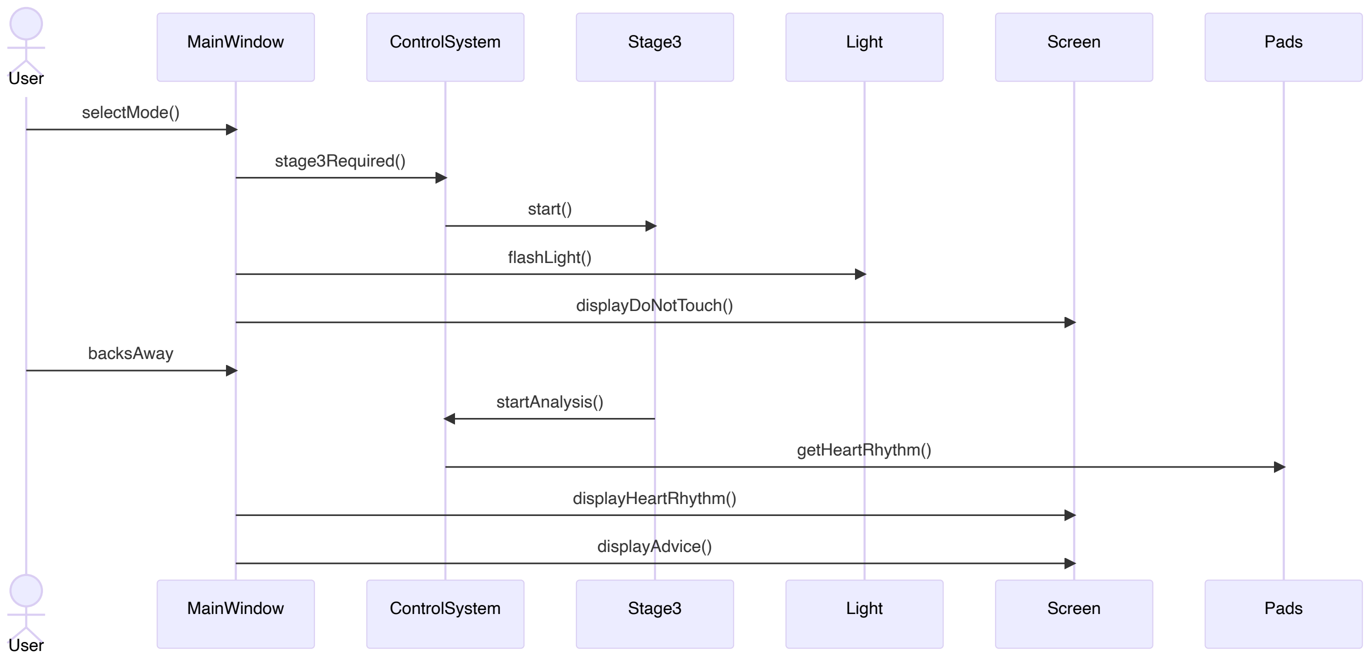
UC2: Electrodes Placement

This shows the case of the user placing the electrode pads on the patient. The device instructs the user, telling them what actions they should perform in order to safely place the electrodes. The system checks to make sure the pads are connected, and adjusts defibrillation energy settings depending on the pads the use selected.


User
MainWindow
Screen
Light
ControlSystem
Stage2
PediatricButton
displayStayCalm()
flashLight()
displayCheckResponse()
checksPatientResponsiveness
flashLight()
displayCallForHelp()
callsForHelp
displayOpenAirways()
opensPatientsAirways
displayCheckForBreathing()
checksForPatientBreathing
start()
checkSafetySystems()
displayExposeCheck()
displayAttachPads()
alt
[Pediatric Patient]
press()
displayPediatricPads()
[Adult Patient]
displayAdultPads()
Selects Appropriate Pads
User
MainWindow
Screen
Light
ControlSystem
Stage2
PediatricButton

UC3: Heart Rythm Analysis

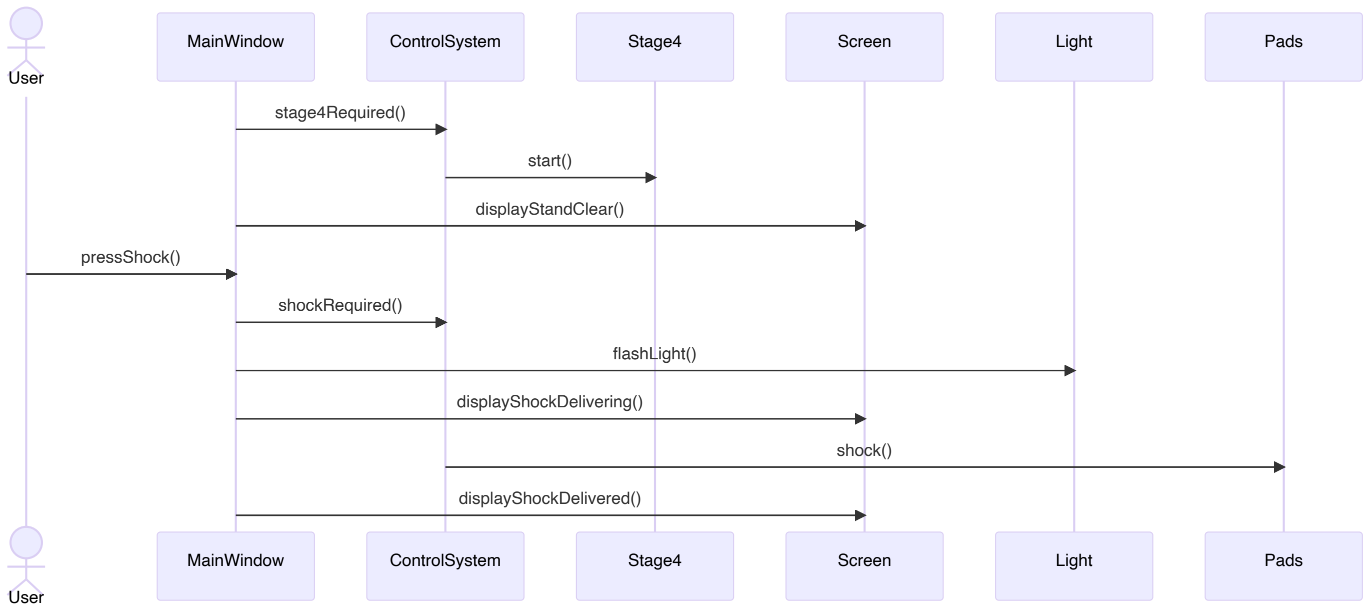
This shows the case of the AED performing a heart rhythm analysis. The system detects and displays the patients rhythm, then provides the user with advice on how to proceed.



UC4: Shock Delivery

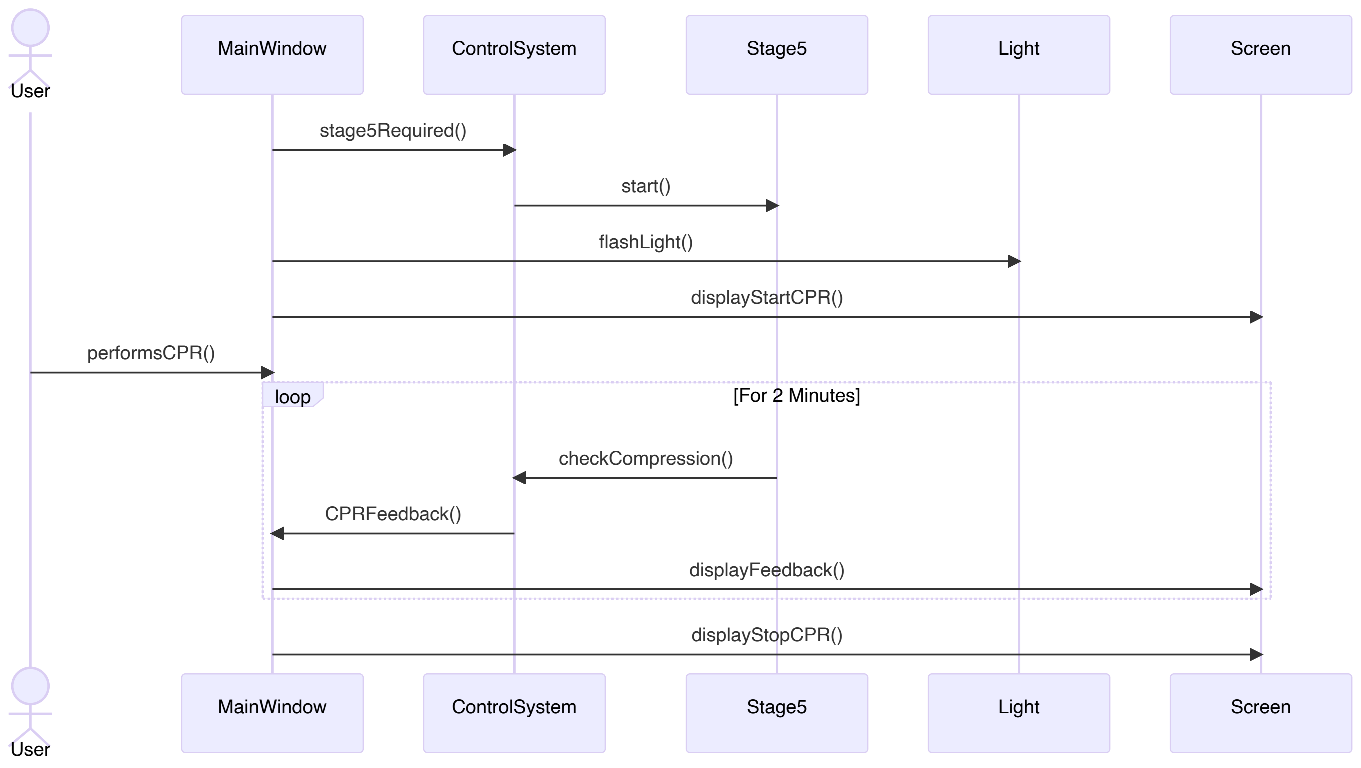
UC4: Shock Delivery

This shows the case of the AED delivering a shock to the patient. The system detected a shockable rhythm, then delivers a shock(s) to the patient in an attempt to restore a normal heartbeat.



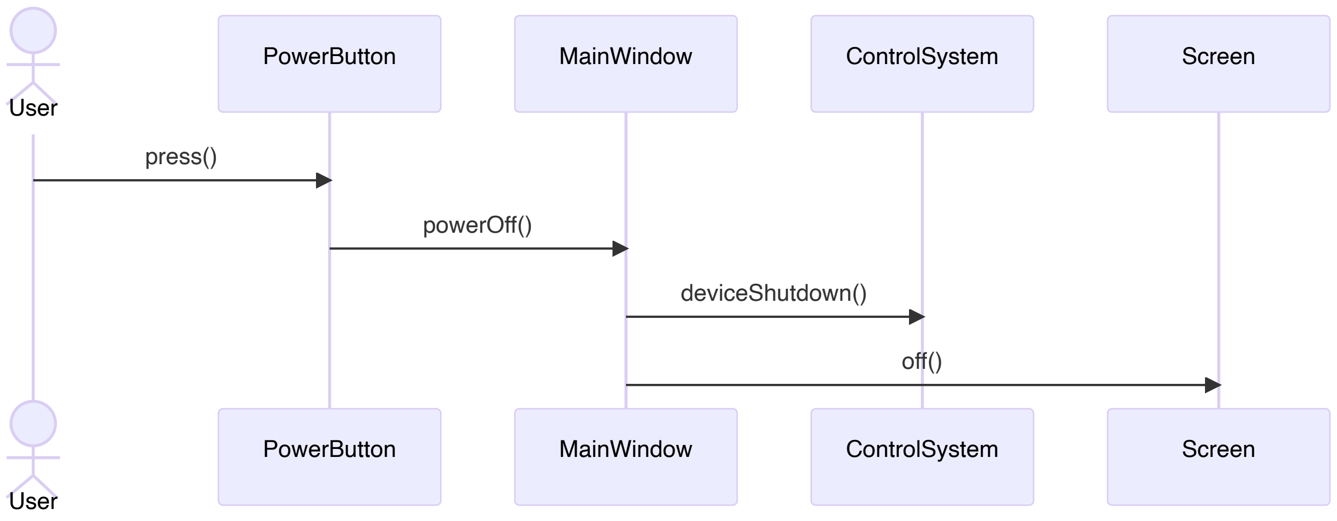
UC5: CPR

This shows the case of the AED guiding the user through CPR. The system displays the instructions to the screen, then monitors and provides real-time CPR feedback to the user for 2 minutes.



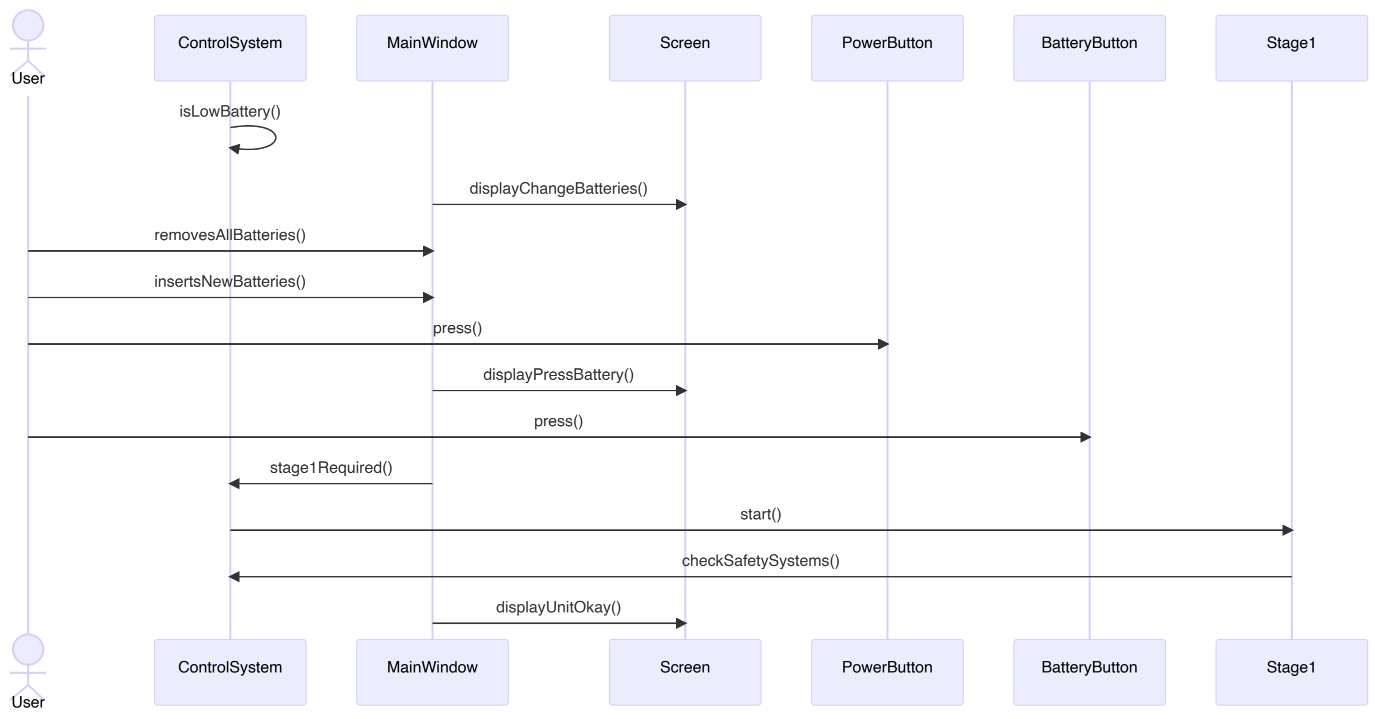
UC6: Power Off AED

This shows the case of the user truning off the AED. The user presses the power button, then the system then turns off.



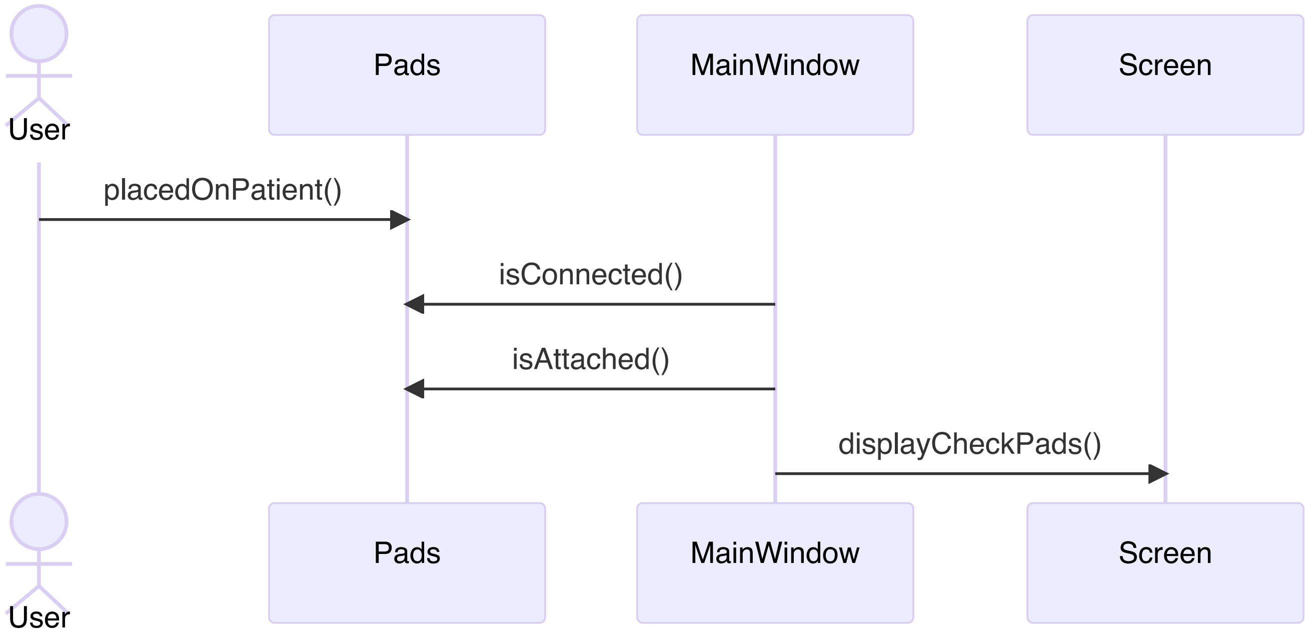
Safety Scenario 1: Low Battery

This shows the safety scenario for when the AED battery is low. The system detects that its battery is low. It prompts the user to change the batteries and persoms a safety check afterwards to ensure proper functionality and sufficient battery power.

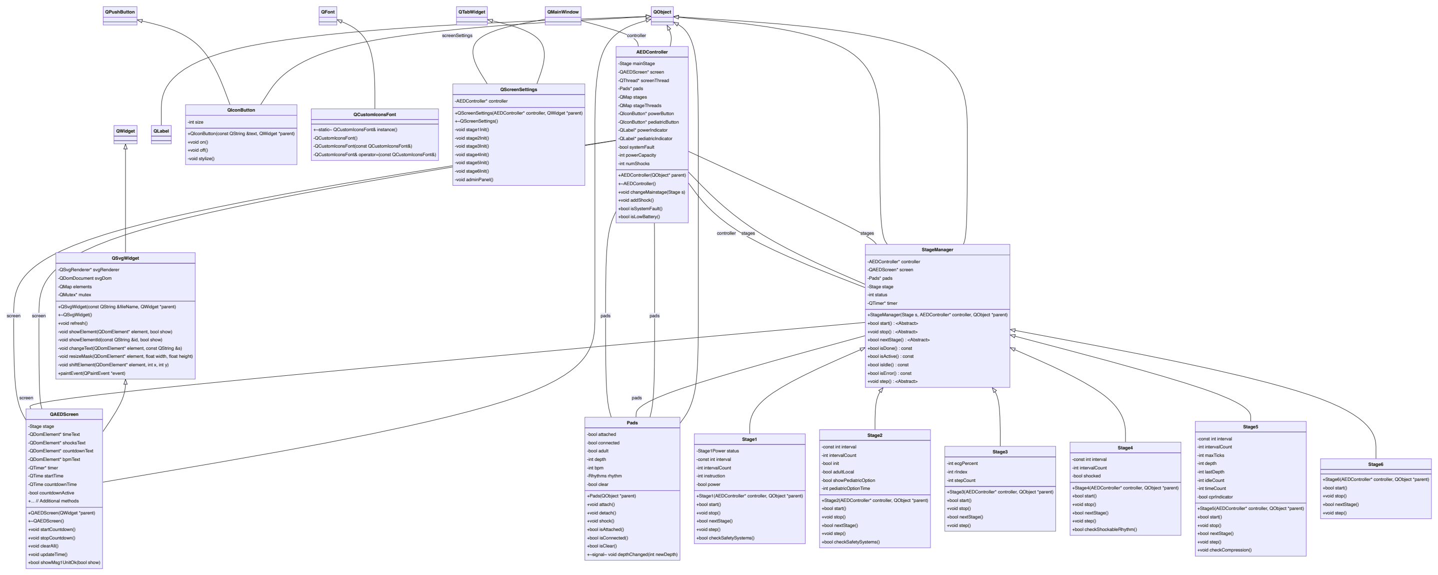


Safety Scenario 2: Pads Error

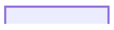
This shows the safety scenario for when the electrode pads are not connected/attached correctly. The system detects that the pads are not properly connected/attached then prompts the user to "CHECK ELECTRODE PADS".

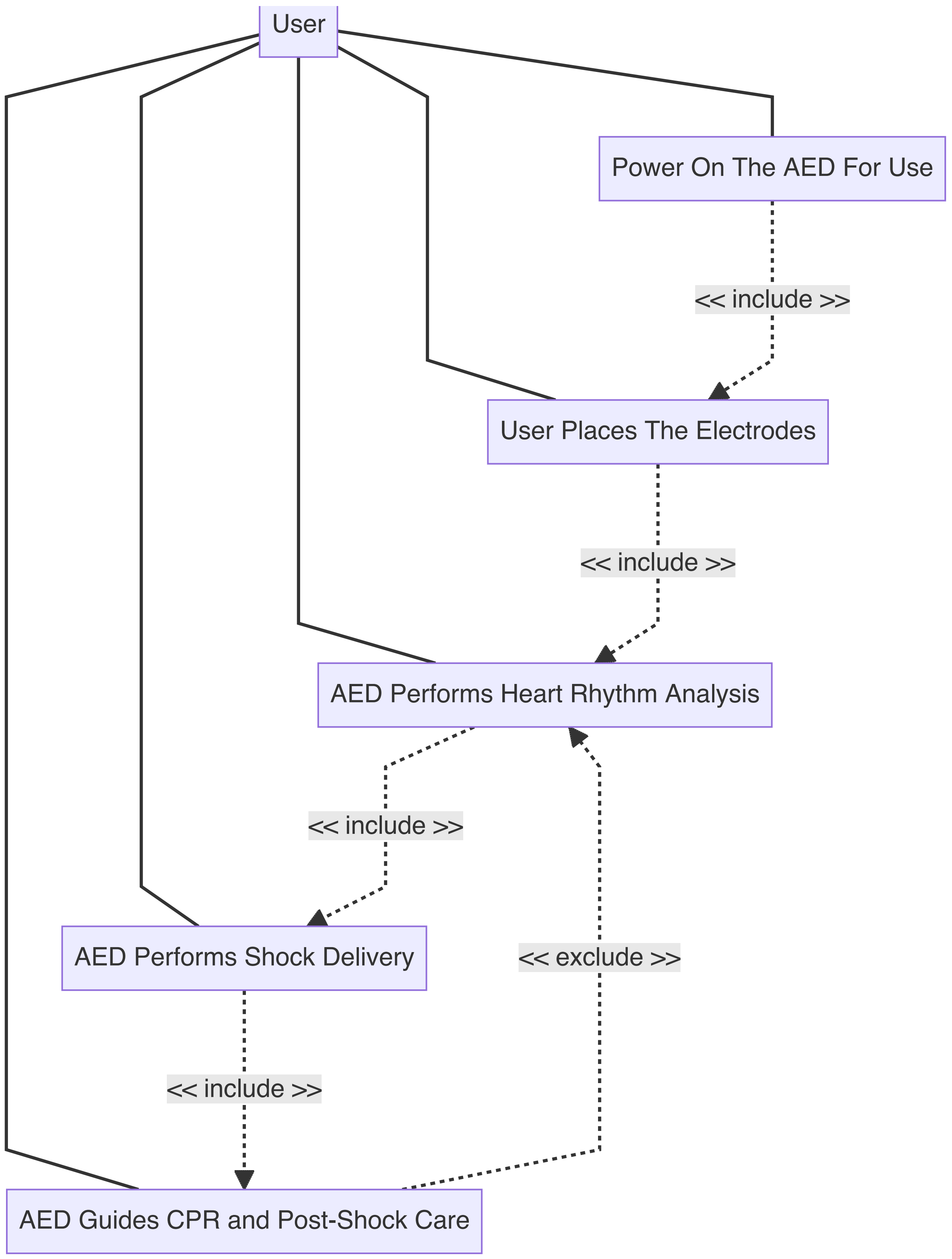


UML Class Diagrams

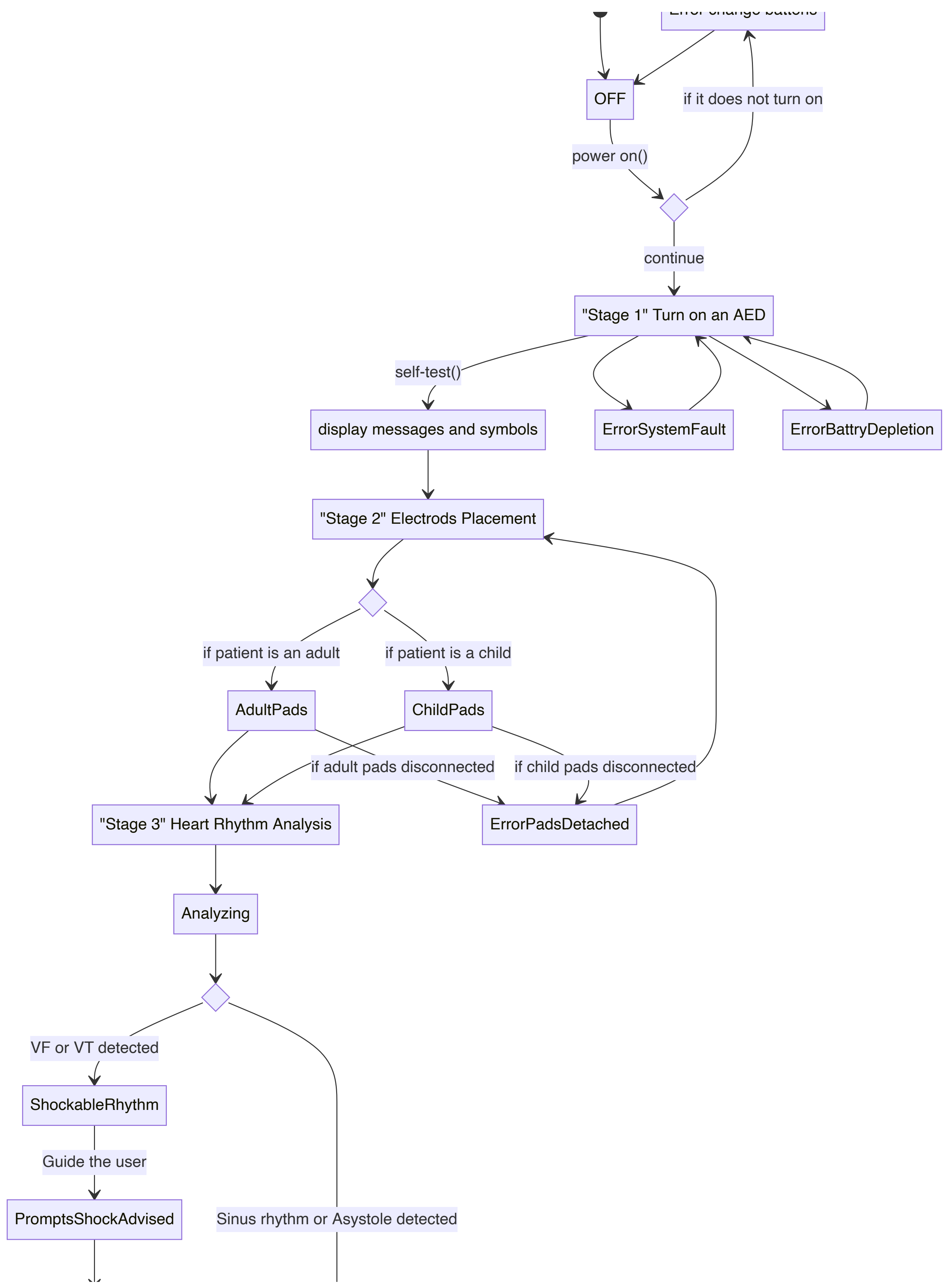


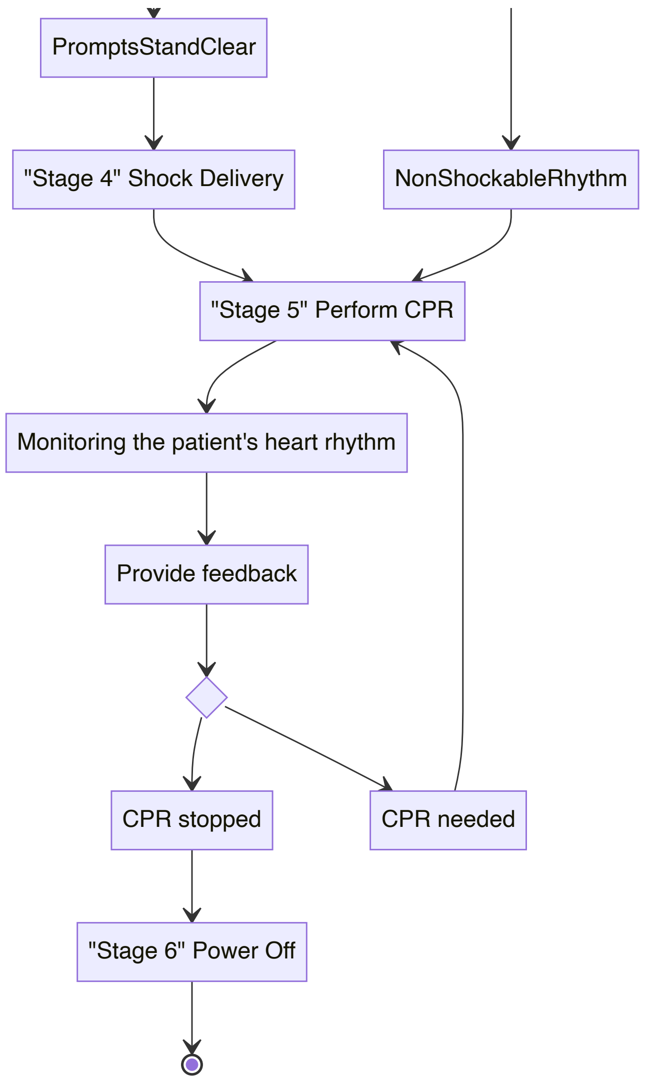
Use Case Diagram




User
Power On The AED For Use
<< include >>
User Places The Electrodes
<< include >>
AED Performs Heart Rhythm Analysis
<< include >>
<< exclude >>
AED Performs Shock Delivery
<< include >>
AED Guides CPR and Post-Shock Care

State Diagram


Error change battons
OFF
if it does not turn on
power on()
continue
"Stage 1" Turn on an AED
self-test()
display messages and symbols
ErrorSystemFault
ErrorBattryDepletion
"Stage 2" Electrods Placement
if patient is an adult
if patient is a child
AdultPads
ChildPads
if adult pads disconnected
if child pads disconnected
"Stage 3" Heart Rhythm Analysis
ErrorPadsDetached
Analyzing
VF or VT detected
ShockableRhythm
Guide the user
PromptsShockAdvised
Sinus rhythm or Asystole detected



Traceability Matrix

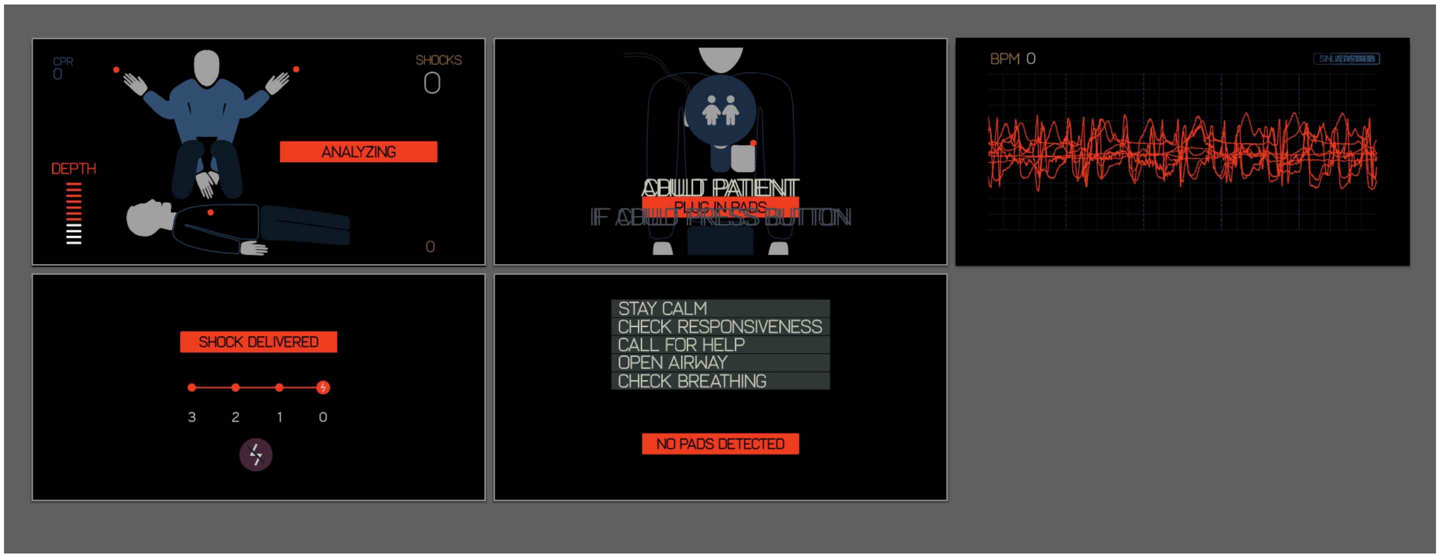
ID	Requirement	Related Use Case and Stage	Fulfilled By (Function)	Description	Test
R1	Power on and self-test	Use Case 1: Power on `Stage1.cpp`	`Stage1::checkSafetySystems()`	Verifies the AED powers on and performs a self-test, displaying the correct status messages.	Run the program on to observe the UI - Turn on AED - press power button - verify "UNIT OK" message displayed
R2	Battery check and low battery warning	Use Case 1 (Ext 2a) `Stage1.cpp`	`Stage1::checkSafetySystems()` & `AEDController::isLowBattery()`	Checks battery status; alerts for replacement if needed.	- AED is turned on - observe low battery warning
R3	Handle system failure during power on	Use Case 1 (Ext 4a) `Stage1.cpp`	`Stage1::checkSafetySystems()` & `AEDController::isSystemFault()`	Handles scenarios where the AED fails its power-up self-test.	- AED is turned on - simulate system failure, verify handling
R4	Guide electrode placement	Use Case 2 `Stage2.cpp`	`Stage2::step()`	Guides the user through electrode placement, including selecting adult or paediatric pads.	- follow instructions for electrodes placement - Verify guidance messages
	Detect	Use Case 2	`Stage2::checkSafetySystems()` &	Detects and alerts when	- Place electrodes with poor contact

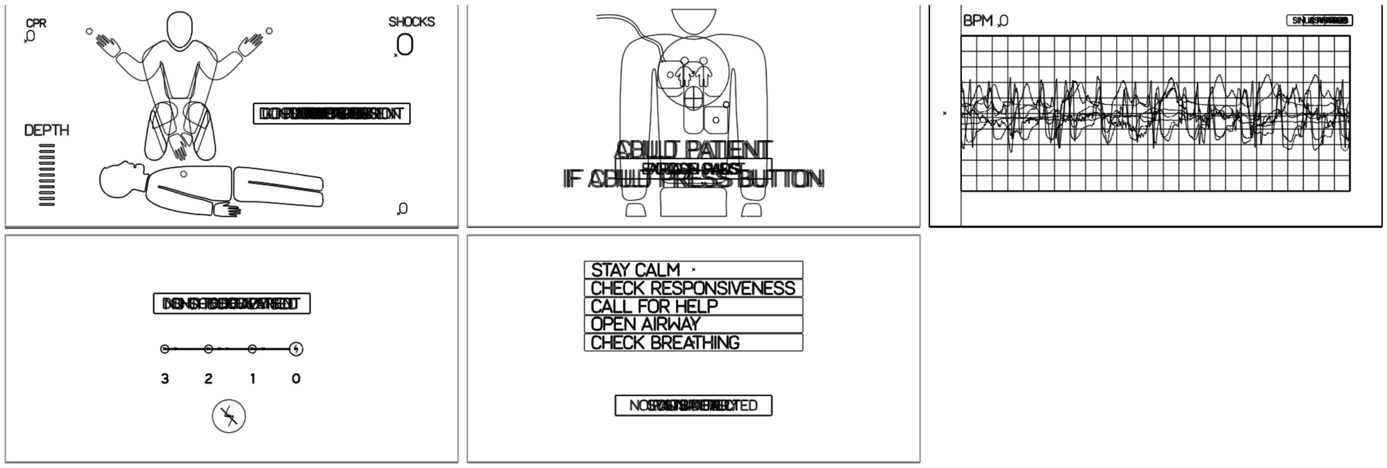
R5	electrode pad errors	(Ext 8a) `Stage2.cpp`	`Pads::isConnected()` & `Pads::isAttached()`	electrode pads are not making good contact.	verify error alert displayed
R6	Perform heart rhythm analysis	Use Case 3 `Stage3.cpp`	`Stage3::step()`	Analyzes the patient's heart rhythm and displays the analysis advice.	-Select Heart Rhythm Analysis mode, verify analysis completion and advice
R7	Handle analysis interruption	Use Case 3 (Ext 3a) `Stage3.cpp`	`Stage3::checkSafetySystems()`	Manages interruptions during heart rhythm analysis, prompting to keep the patient still.	Intrerrupt analysis, verify prompt to keep patient still displayed
R8	Advise and deliver shock	Use Case 4 `Stage4.cpp`	`Stage4::step()` & `AEDController::addShock()`	Manages the process of advising and delivering a shock, including handling errors.	-simulate shock advised, verify "STAND CLEAR" message displayed. -simulate sock delivery, verify "SHOCK DELIVERED" message displayed
R9	Guide CPR and post-shock care	Use Case 5 `Stage5.cpp`	`Stage5::step()`	Provides guidance for CPR and post-shock care, including real-time feedback.	-Follow CPR guidance messages -verify real-time CPR feedback
R10	Detect and instruct on CPR quality	Use Case 5 (Extensions 3a, 3b, 3c) `Stage5.cpp`	`Stage5::step()` & `Pads::getDepth()`	Monitors CPR quality and provides instructions to improve, such as "PUSH HARDER".	-Perform CPR, monitor quality, verify feedback messages
R11	Communicate no shock advised	Use Case 3 (Ext 5b) `Stage3.cpp`	`Stage4::step()`	Communicates when no shock is advised after rhythm analysis.	Analyze heart rhythm, verify communication message displayed
R12	Manage multiple shock deliveries	Use Case 4 (Ext 5a) `Stage4.cpp`	`Stage4::step()`	Manages and tracks the number of shocks delivered.	simulation multiple shocks, verify correct tracking and management
R13	Handle non-responsive patient	Use Case 2 `Stage2.cpp`	`Stage2::checkSafetySystems()` & `Stage3::step()`	Manages the scenario when the patient is non-responsive during electrode placement.	simulate non-responsive patient, verify correct handling
R14	CPR Instructional Feedback	Use Case 5 `Stage5.cpp`	`Stage5::checkCompression()`	Offers instructional feedback during CPR, such as compression rate and depth.	Perform CPR, verify feedback messages
R15	Emergency Communication	Use Case 1 `Stage1.cpp`	`Stage1::step()`	Prompts user to communicate with emergency services during initial operation.	verify message shows
R16	Pediatric Adjustment for Shock Energy	Use Case 2 `Stage2.cpp`	`adjustForPediatricShockEnergy()`	Adjusts shock energy levels based on the detection of paediatric pads.	simulate shock delivery with paediatric pads, verify adjustments
R17	User Guidance for AED Setup	Use Case 1 `Stage1.cpp`	`Stage1::step()`	Guides the user through initial setup procedures, including battery installation.	Turn on AED, verify guidance
R18	Shock Delivery Countdown	Use Case 4 `Stage4.cpp`	`Stage4::step()`	Performs and displays the countdown before shock delivery.	simulate shock delivery, verify countdown
R19	Post-use Device Maintenance Check	Use Case 1 `Stage1.cpp`	`Stage6::step()`	Checks the device for maintenance needs after use, including battery and pad status.	simulate post-use maintenance - verify correct checks

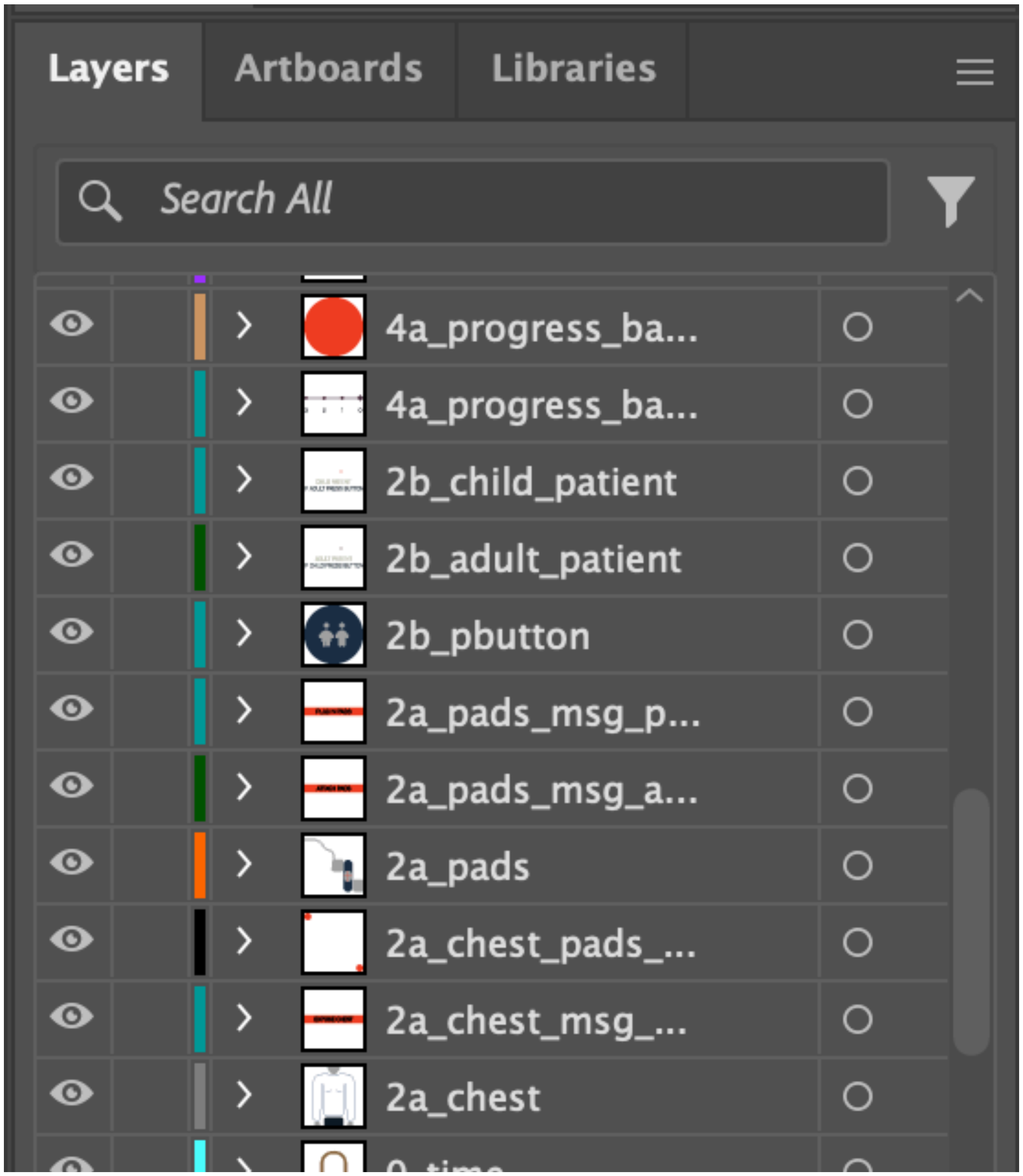
R20	Real-time Battery Status Monitoring	All Stages	`AEDController::isLowBattery()` & `Stage1::checkSafetySystems()`	Continuously monitors the battery status and alerts if levels are critical.	simulate battery status changes, verify alerts
R21	Periodic Device Self-Test	Use case 6 `Stage6.cpp`	`Stage1::checkSafetySystems()`	Regularly performs self-tests to ensure device readiness and functionality.	simulate periodic self-test, verify successful self-test
R22	Robust Electrode Adhesion Check	Use case 2 `Stage2.cpp`	`Stage2::checkSafetySystems()`	Verifies the adhesion of electrodes to the patient's skin for effective analysis and shock delivery.	simulate electrode adhesion check, verify correct response
R23	Advanced Arrhythmia Detection Algorithms	Use case 3 `Stage3.cpp`	`Stage3::step()`	Utilizes advanced algorithms for accurate arrhythmia detection.	simulate heart rhythm analysis with advanced algorithms
R24	Maintenance and Service Notifications	Use Case 6 `Stage6.cpp`	`Stage6::step()`	Notifies the user of maintenance and service needs of the AED device.	simulate maintenance needs verify notifications

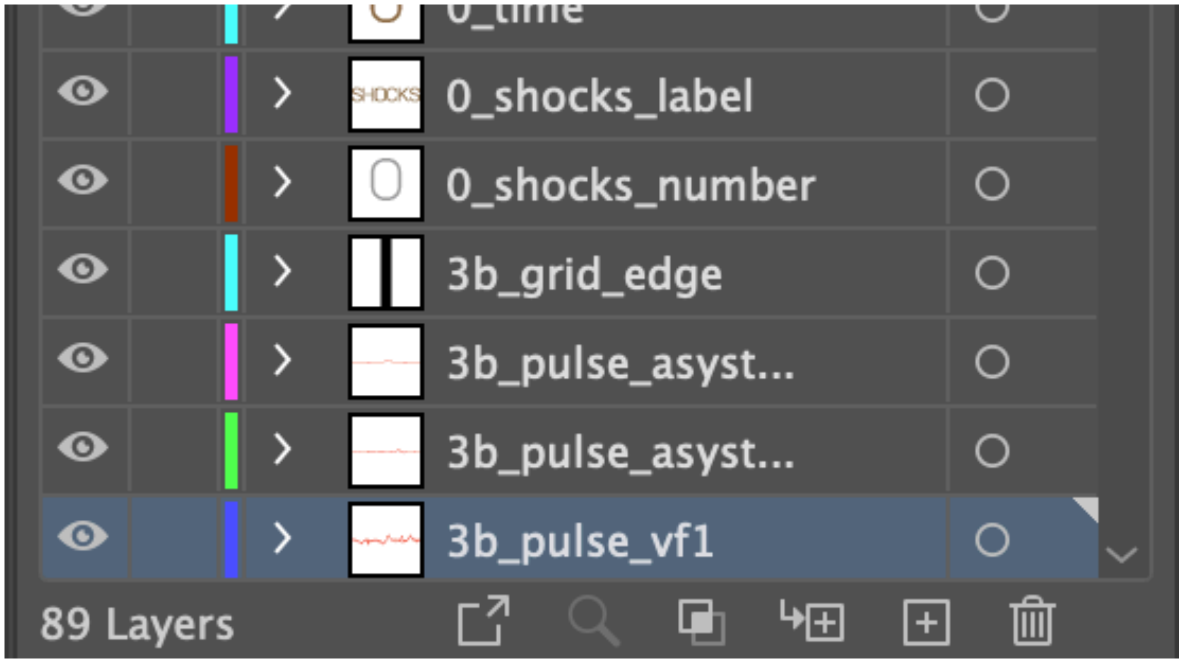
Display

The display is an SVG file which I created that consists of multiple different layers and objects, each assigned their own unique ID.




CPR
SHOCKS
DEPTH
ADULT PATIENT
IF ADULT PRESS BUTTON
BPM
3
2
1
0
STAY CALM
CHECK RESPONSIVENESS
CALL FOR HELP
OPEN AIRWAY
CHECK BREATHING


Layers
Artboards
Libraries
Search All
4a_progress_ba...
4a_progress_ba...
2b_child_patient
2b_adult_patient
2b_pbutton
2a_pads_msg_p...
2a_pads_msg_a...
2a_pads
2a_chest_pads_...
2a_chest_msg_...
2a_chest



The class `QSvgWidget` handles all direct SVG element manipulation, it has methods to show, hide and move these elements. It also has a method that can edit the text in the SVG file, this is used to update the timer displays on the screen.

The `QAEDScreen` extends `QSvgWidget` and provides methods that allow anyone to easily update and show info on the AED's screen. Such methods include `QAEDScreen::showMsg2aAttachPads()` to easily show a message on the screen or `QAEDScreen::setShockCount` to increase shock count. This class is key for the screen to function as it allows you to not have to deal with directly with manipulating the SVG file.

Team members & Contributions

```
    Moses Muwanga - 101007920
    Wendy Pang - 101196606
    Abdullah Mostafa - 101008311
    Fatemeh Mashhadi - 101204634
    Jian Zhuang - 100997755
```

1. **Use cases**: Moses, Wendy
2. **UML class diagram**: Abdullah
3. **Sequence diagram**: Moses, Wendy
4. **State diagram**: Fatemeh
5. **Use case diagram**: Fatemeh
6. **Traceability matrix**: Abdullah, Fatemeh, Jian
7. **Design Patterns textual explanations:** Jian

8. **GUI and graphics**: Abdullah

9. **Code implementation**: Abdullah
    - Stage1.cpp implementation: Moses (Power on sequence)
    - Stage2.cpp implementation: Fatemeh (Install pads sequence)
    - Stage6.cpp implementation: Wendy (Power off sequence)