Sunny Parmar
Naveen Sajeendran
Essam Abdel Hamid
Affan Beg

COMP3004 Final Project Documentation

Software-Based Prototype of AED: Use Cases

Use Case: Power On
Actor(s): User
Precondition(s):

- AED is off, Battery installed

Main Success Scenario:

1. User activates the AED by pressing the power button.
2. AED goes through a self test to check if all the systems are ready. (refer to the *Self Test* use case)
3. AED confirms readiness through visual and audio indicators (e.g., lights, beeps).
4. AED informs user to place electrodes (Adult or Child depending on patient age group respectively)
5. AED performs ECG analysis and detects a heart condition (Tachycardia or Fibrillation) and advises accordingly (Refer to *Regular heartbeat* use case or *Irregular Heartbeat* use cases

Extensions

- 1a. If the battery is low:
  I. AED displays a low battery warning.
  II. The user replaces the battery or connects to a power source.
- 2a. If internal self-test fails:
  I. AED displays an error code and specific malfunction.
  II. User refers to the manual for troubleshooting or contacts maintenance.

Use Case: Regular Heartbeat (Non Shockable Scenario)
Actor(s): User, Patient
Precondition(s):

- AED is on, adhesive electrode pads correctly attached to the patient.

Main Success Scenario:

1. AED starts a patient elapsed timer(To track patients heart condition) and begins analyzing the patient's heart rhythm.
2. AED detects a regular heartbeat and assesses it is non-shockable (Sinus Rhythm/Pulseless electrical activity, Asystole).
3. AED indicates through visual and audio signals that no shock is required.
4. AED prompts the user that the patient has a regular heartbeat.
5. AED advises the user to begin CPR while AED is monitoring heart rate (refer to *Real-Time CPR* use case for detailed CPR instructions).

Extensions

- 1a. If electrodes are poorly attached:
    I. AED requests users to adjust electrode placement for accurate reading.
- 5a. If patient's condition changes:
    I. AED re-evaluates the heart rhythm for any new instructions. (Refer to step 5 of Power On use case)

Use Case: Irregular Heartbeat (Shockable Scenario)
Actor(s): User, Patient
Precondition(s): AED is on, adhesive electrode pads correctly attached to the patient.

Main Success Scenario

1. AED starts a patient elapsed timer(To track patients heart condition) and begins analyzing the patient's heart rhythm.
2. AED detects a shockable rhythm (Ventricular fibrillation or ventricular tachycardia).
3. AED will prompt the user to clear and press the button to charge for a shock.
4. AED administers the shock to the patient if user preses button to shock.
5. AED advises the user to begin CPR while AED is monitoring heart rate (refer to *Real-Time CPR* use case for detailed CPR instructions).

Extensions

- 2a. If the electrode pads lose contact during the analysis:
    I. AED alerts the user to recheck and secure electrode placement.
- 5a. If patient's condition changes:
    I. AED re-evaluates the heart rhythm for any new instructions. (Refer to step 5 of Power On use case)

Use Case: Real-Time CPR
Actor(s): User, Patient
Precondition(s): AED is on, Heart condition has been analysed
Main Success Scenario:

1. AED advises user to preform CPR on the patient.
2. AED monitors the patient's heart rhythm during CPR.
3. AED provides real-time feedback on CPR, guiding the user on compression depth and rate:
    I. Perform 30:2 chest compressions to ventilations (until airway is secure). Fully extend your elbows, wrists and fingers. Have both hands palm downwards with fingers interlocked. Place the carpal area of the hand over the lower sternum and apply all pressure over this point. Perform compressions at a rate (2/second) and a depth of 5-6cm. Chest compressions should be continuous
    II. Switch CPR provider during the rhythm check every 2 minutes (or earlier if they tire)

Extensions:

- 1a. If CPR is ineffective (incorrect depth or rate)
    I. AED instructs the user to adjust the technique. (Refer to Real-Time CPR step 3)
- 3a. If AED detects any interruption in CPR
    I. AED prompts the user to resume CPR immediately.

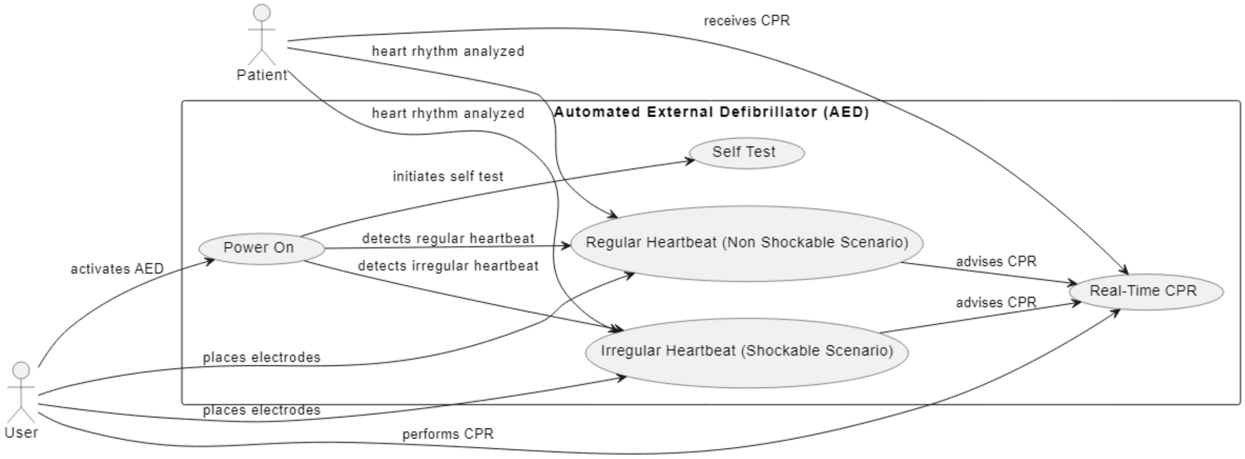
Use Case: Self Test
Actor(s): User, Patient
Precondition(s): AED is powered on (Initiating Self Test)
Main Success Scenario:

1. AED performs a series of checks, including battery level, electrode pad readiness, internal circuitry, and software integrity.
2. AED displays to the system that it is ready to use, confirming functionality or indicating any issues.
3. If all systems are operational, AED signals readiness for use with visual and auditory cues.

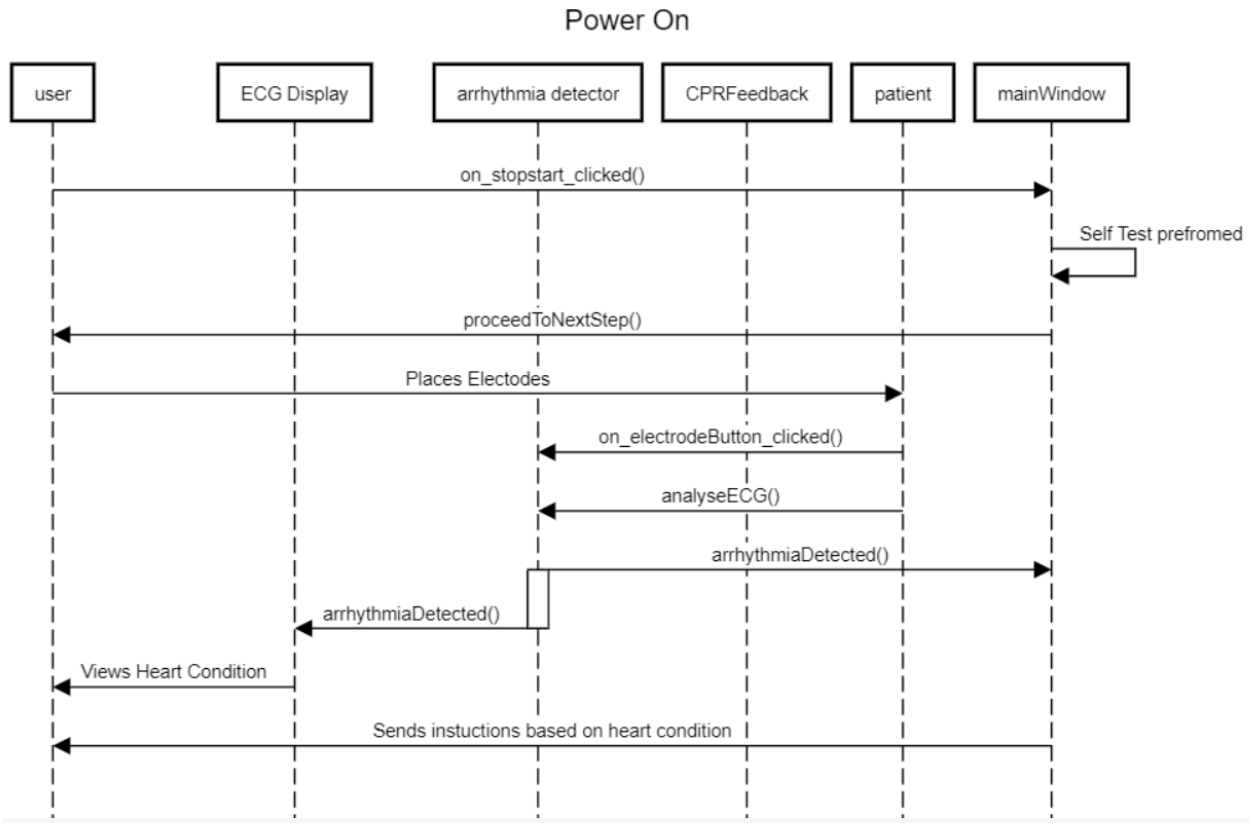
Extensions:

- 1a. If the battery level is low:
    I. AED displays a low battery warning, prompting battery replacement.
- 1b. If electrode pads are missing or not functional:
    I. AED prompts to check or replace the electrode pads.

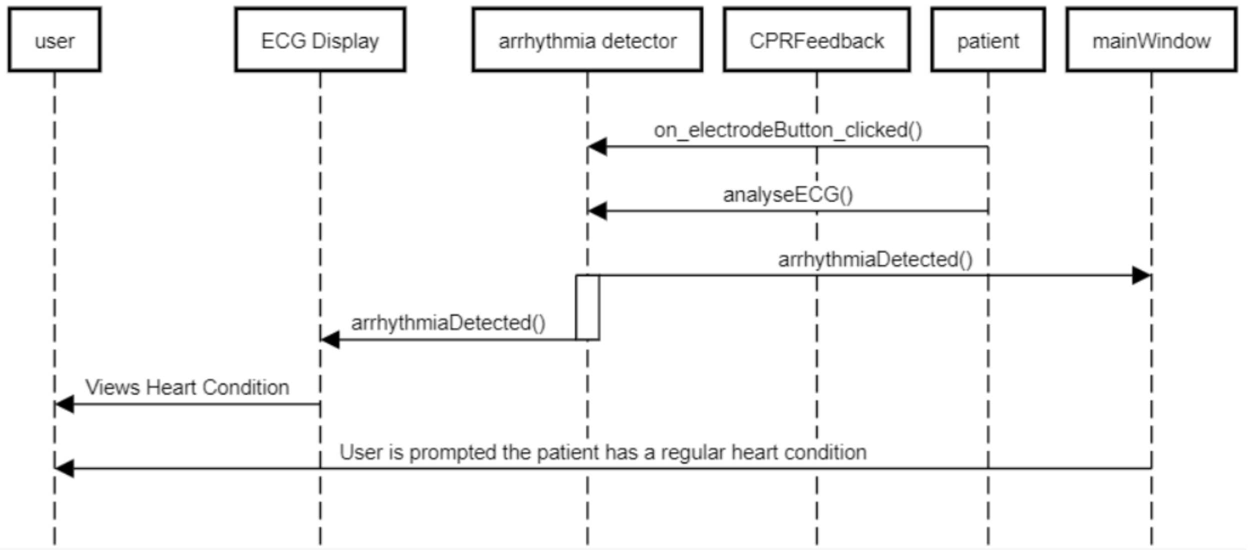
AED Use Case Diagram



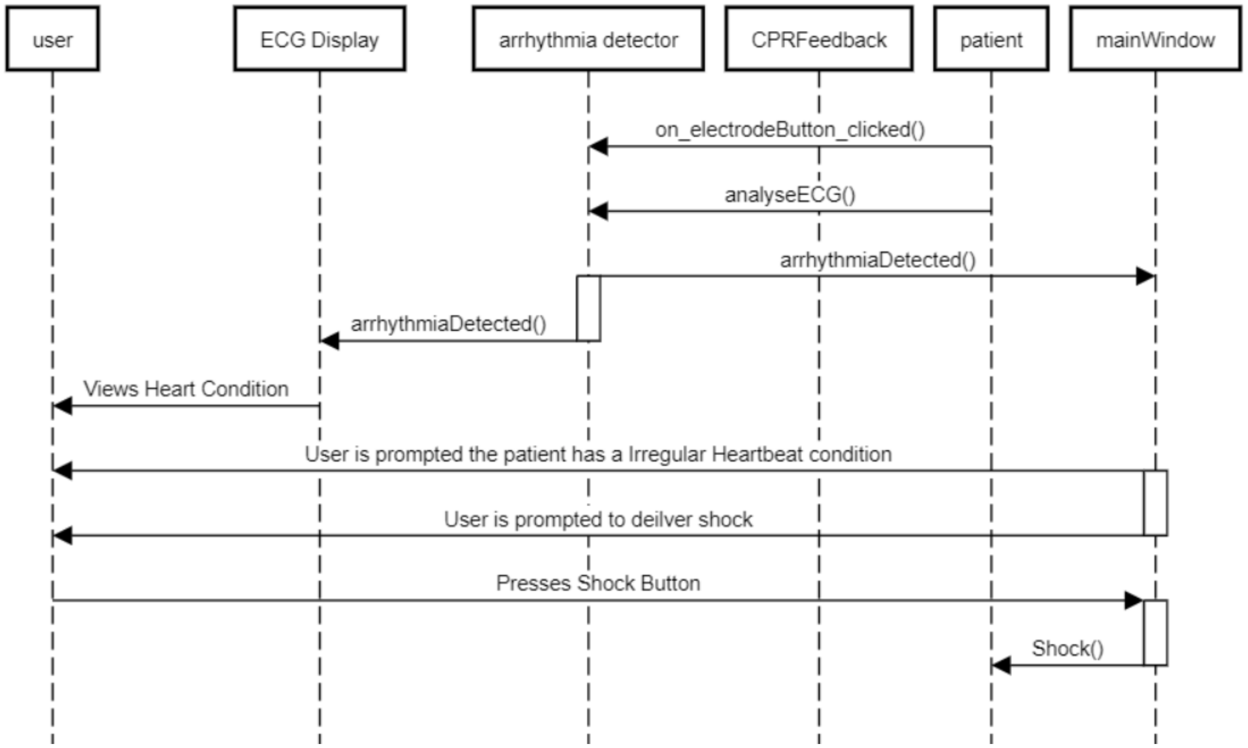
AED Sequence Diagrams



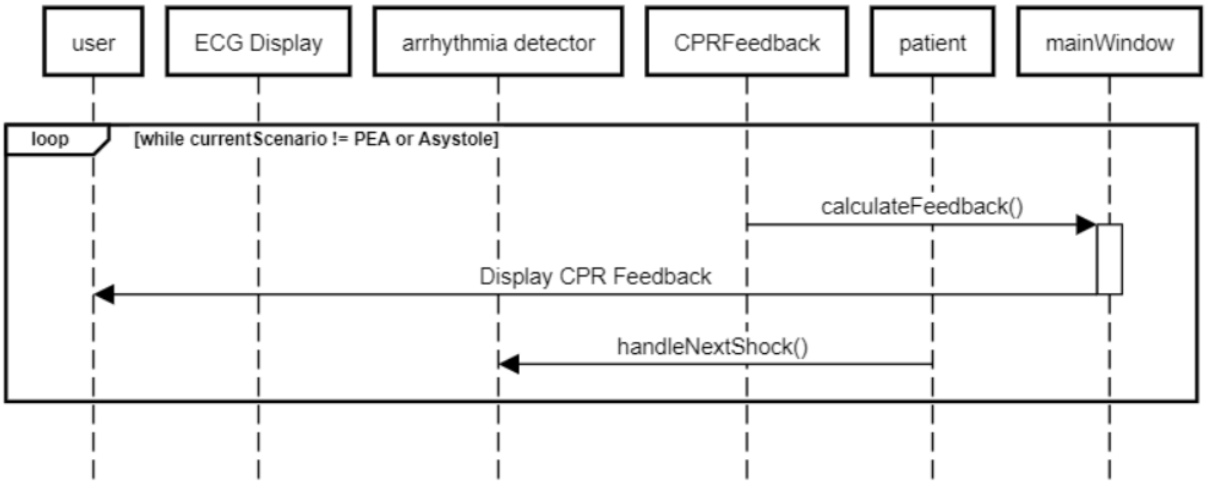
Regular Heartbeat



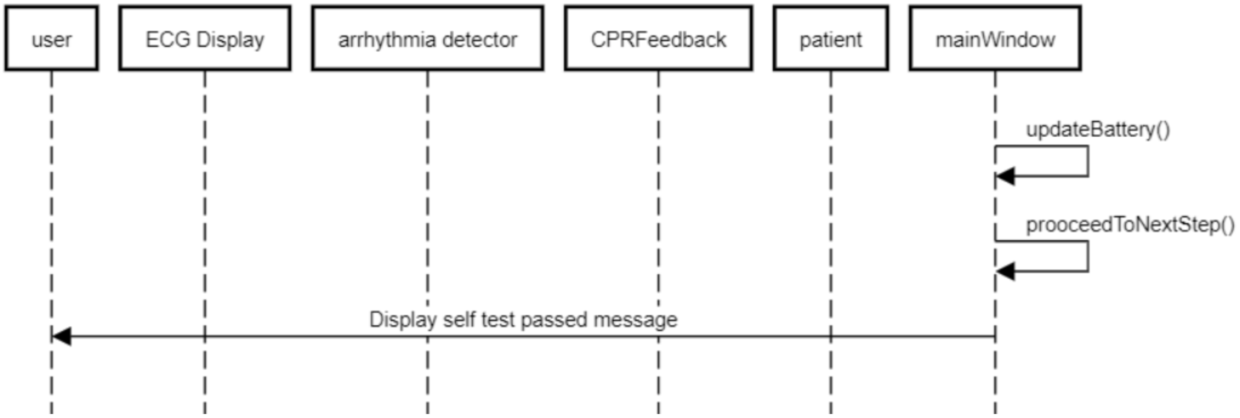
Irregular Heartbeat



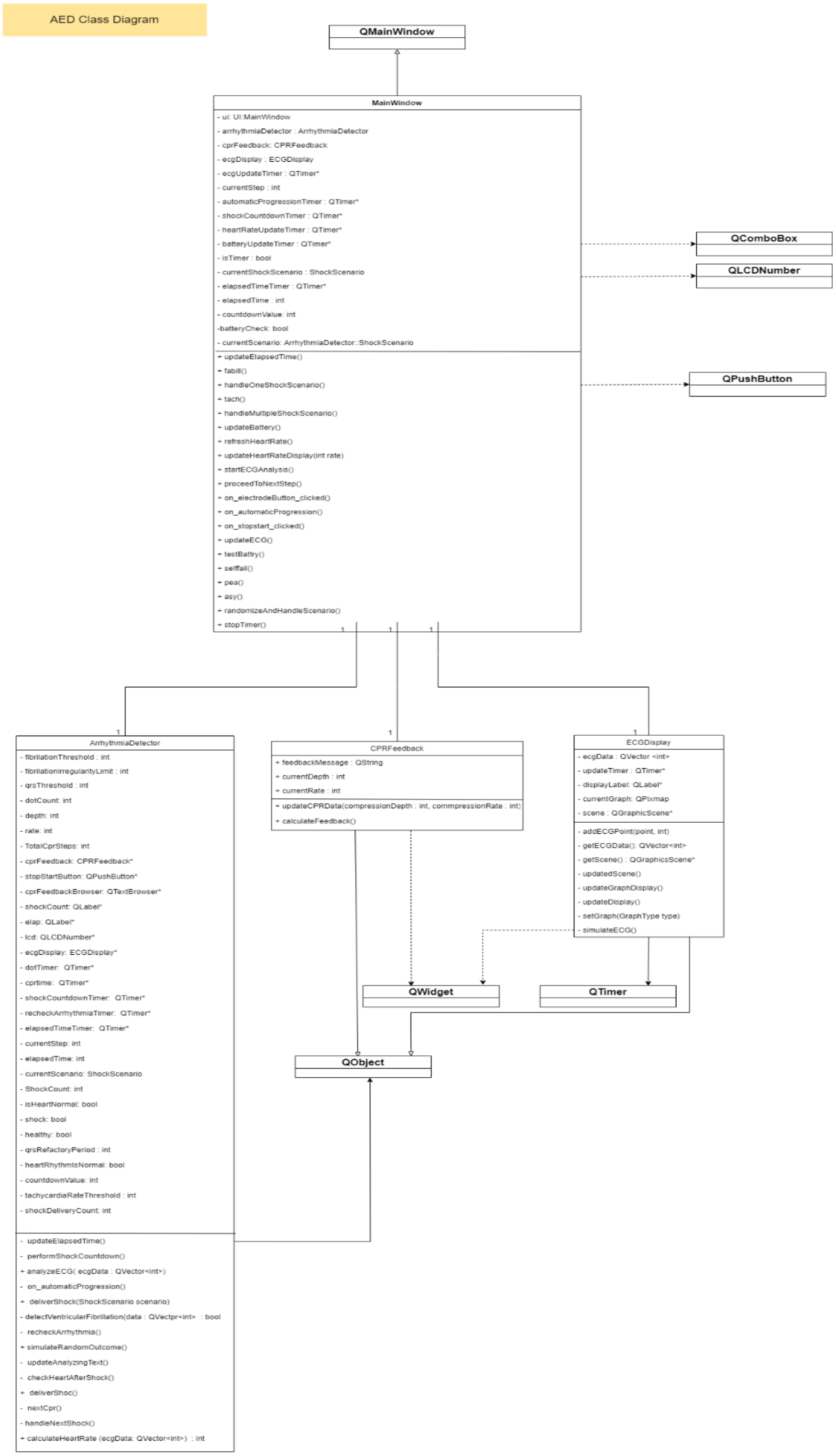
Real-Time CPR



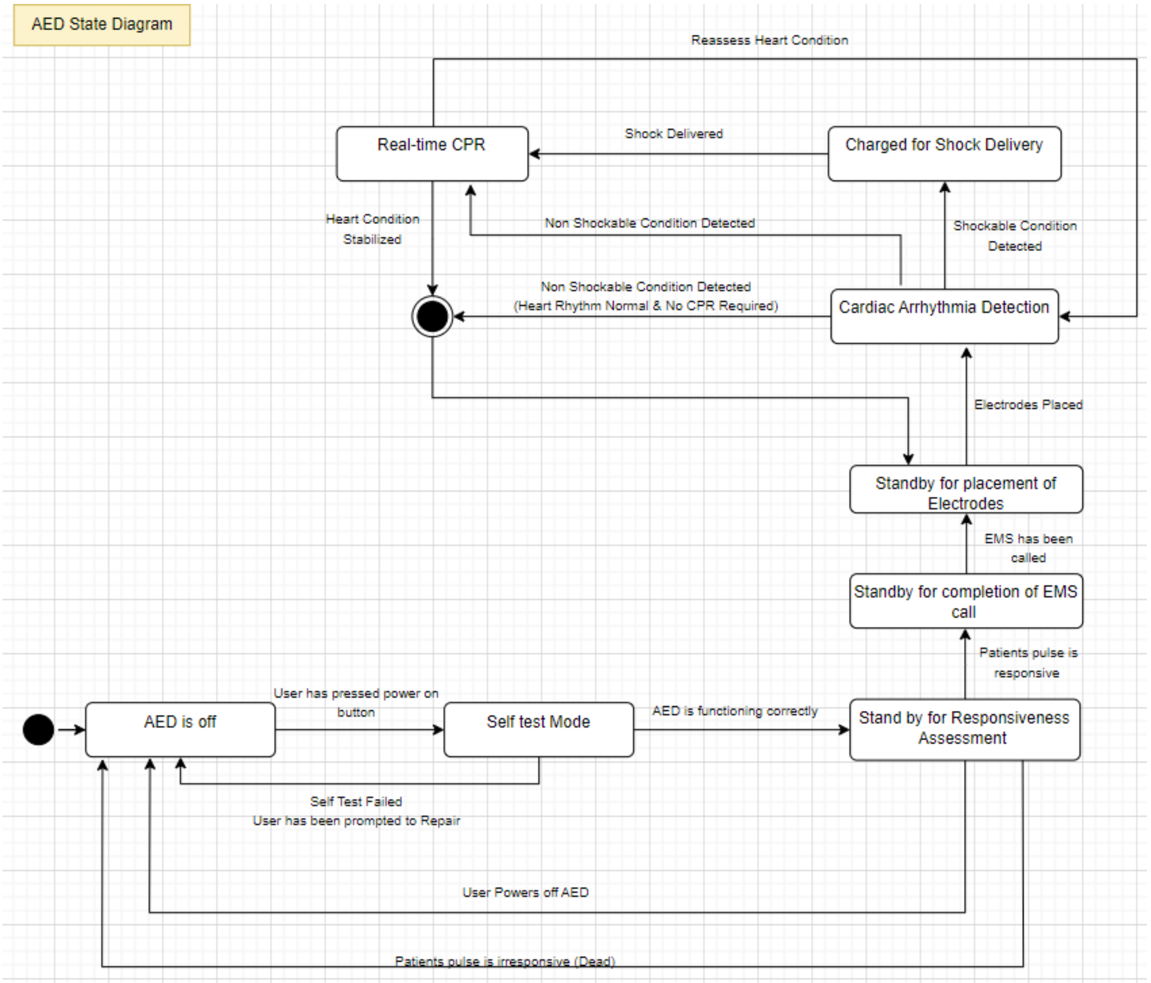
Self Test



AED Software Class Diagram


AED Class Diagram
QMainWindow
MainWindow
- ui: UI:MainWindow
- arrhythmiaDetector : ArrhythmiaDetector
- cprFeedback: CPRFeedback
- ecgDisplay : ECGDisplay
- ecgUpdateTimer : QTimer*
- currentStep : int
- automaticProgressionTimer : QTimer*
- shockCountdownTimer : QTimer*
- heartRateUpdateTimer : QTimer*
- batteryUpdateTimer : QTimer*
- isTimer : bool
- currentShockScenario : ShockScenario
- elapsedTimeTimer : QTimer*
- elapsedTime : int
- countdownValue: int
-batteryCheck: bool
- currentScenario: ArrhythmiaDetector::ShockScenario
+ updateElapsedTime()
+ fabill()
+ handleOneShockScenario()
+ tach()
+ handleMultipleShockScenario()
+ updateBattery()
+ refreshHeartRate()
+ updateHeartRateDisplay(int rate)
+ startECGAnalysis()
+ proceedToNextStep()
+ on_electrodeButton_clicked()
+ on_automaticProgression()
+ on_stopstart_clicked()
+ updateECG()
+ testBattry()
+ selffail()
+ pea()
+ asy()
+ randomizeAndHandleScenario()
+ stopTimer()
1
1
1
QComboBox
QLCDNumber
QPushButton
1
ArrhythmiaDetector
- fibrilationThreshold : int
- fibrilationirregularityLimit : int
- qrsThreshold : int
- dotCount: int
- depth: int
- rate: int
- TotalCprSteps: int
- cprFeedback: CPRFeedback*
- stopStartButton: QPushButton*
- cprFeedbackBrowser: QTextBrowser*
- shockCount: QLabel*
- elap: QLabel*
- lcd: QLCDNumber*
- ecgDisplay: ECGDisplay*
- dotTimer: QTimer*
- cprtime: QTimer*
- shockCountdownTimer: QTimer*
- recheckArrhythmiaTimer: QTimer*
- elapsedTimeTimer: QTimer*
- currentStep: int
- elapsedTime: int
- currentScenario: ShockScenario
- ShockCount: int
- isHeartNormal: bool
- shock: bool
- healthy: bool
- qrsRefactoryPeriod : int
- heartRhythmIsNormal: bool
- countdownValue: int
- tachycardiaRateThreshold : int
- shockDeliveryCount: int
- updateElapsedTime()
- performShockCountdown()
+ analyzeECG(ecgData : QVector<int>)
- on_automaticProgression()
+ deliverShock(ShockScenario scenario)
- detectVentricularFibrillation(data : QVectpr<int> : bool
- recheckArrhythmia()
+ simulateRandomOutcome()
- updateAnalyzingText()
- checkHeartAfterShock()
+ deliverShoc()
- nextCpr()
- handleNextShock()
+ calculateHeartRate (ecgData: QVector<int>) : int
1
CPRFeedback
+ feedbackMessage : QString
+ currentDepth : int
+ currentRate : int
+ updateCPRData(compressionDepth : int, commpressionRate : int)
+ calculateFeedback()
1
ECGDisplay
- ecgData : QVector <int>
- updateTimer : QTimer*
- displayLabel: QLabel*
- currentGraph: QPixmap
- scene : QGraphicScene*
- addECGPoint(point, int)
- getECGData(): QVector<int>
- getScene() : QGraphicsScene*
- updatedScene()
- updateGraphDisplay()
- updateDisplay()
- setGraph(GraphType type)
- simulateECG()
QWidget
QTimer
QObject

AED State Diagram


AED State Diagram
Reassess Heart Condition
Real-time CPR
Shock Delivered
Charged for Shock Delivery
Heart Condition Stabilized
Non Shockable Condition Detected
Shockable Condition Detected
Non Shockable Condition Detected (Heart Rhythm Normal & No CPR Required)
Cardiac Arrhythmia Detection
Electrodes Placed
Standby for placement of Electrodes
EMS has been called
Standby for completion of EMS call
Patients pulse is responsive
User has pressed power on button
AED is off
Self test Mode
AED is functioning correctly
Stand by for Responsiveness Assessment
Self Test Failed
User has been prompted to Repair
User Powers off AED
Patients pulse is irresponsive (Dead)

Explanation of Design Decisions

In developing the AED simulation software, we incorporated several architectural design patterns to ensure a robust and user-centric system:

- Observer Pattern: We've implemented Qt's signal and slot mechanism, which exemplifies the Observer pattern. This allows our ArrhythmiaDetector class to emit signals such as arrhythmiaDetected and heartRateCalculated, to which our MainWindow class responds by updating the user interface. Here, ArrhythmiaDetector acts as the subject, and MainWindow functions as the observer, promoting loose coupling and enhancing responsiveness.

- Mediator Pattern: We positioned the MainWindow class to function as a mediator. It's the central hub that orchestrates interactions between components like ArrhythmiaDetector, CPRFeedback, and ECGDisplay. This design pattern simplifies communication among the components, streamlining the control flow within the application.

- Modular Design: We structured the software into modules that closely replicate the AED Plus's essential functionalities. These modules encompass:

    - A graphical user interface designed to mirror the look and feel of the AED Plus, facilitating a seamless user experience.
    - A real-time CPR feedback system that provides users with immediate and actionable guidance during CPR procedures.
    - A cardiac arrhythmia detection module that accurately simulates the AED's diagnostic capabilities, offering clear instructions for shock delivery or CPR based on the detected rhythm.

Traceability Matrix

ID	Requirement	Related use cases	Fulfilled By	Test	Description
1	The system must be able to detect different shock scenarios	Irregular Heartbeat without CPR (Shockable Scenario)	ArrhythmiaDetector::ShockScenario, ArrhythmiaDetector::deliverShock	Run simulation and check for correct shock scenario handling.	The system can detect multiple shock scenarios, such as OneShock, MultipleShocks, Fibrillation, etc., and handle them accordingly using ArrhythmiaDetector::deliverShock.
2	The system should analyze ECG data and calculate heart rate.	Power On, Regular Heartbeat (Non Shockable Scenario), Irregular Heartbeat without CPR (Shockable Scenario)	ArrhythmiaDetector::analyzeECG, ArrhythmiaDetector::calculateHeartRate	Input ECG data and verify calculated heart rate.	The ArrhythmiaDetector class includes methods to analyze ECG data for arrhythmias and calculate the heart rate from the ECG data.
3	The system must provide CPR feedback based on compression depth and rate.	Real-Time CPR	CPRFeedback::updateCPRData, CPRFeedback::calculateFeedback	Perform CPR and observe feedback updates.	The CPRFeedback class updates CPR data and provides feedback messages

					that indicate whether compression depth and rate are within acceptable ranges.
4	The system should simulate an ECG and update the display accordingly.	Power On (as part of the self-test to check system readiness)	ECGDisplay::simulateECG, ECGDisplay::updateDisplay	Simulate ECG and check display updates.	The ECGDisplay class can simulate ECG data points and update the display with the generated ECG waveform.
5	The system must update elapsed time during shock scenarios.	Power On, Regular Heartbeat (Non Shockable Scenario), Irregular Heartbeat without CPR (Shockable Scenario)	ArrhythmiaDetector::updateElapsedTime, MainWindow::updateElapsedTime	Observe the UI for elapsed time during a shock scenario.	Elapsed time is updated in both the ArrhythmiaDetector and MainWindow classes and reflected in the UI.
6	The system should update the battery level and provide alerts when low.	Power On (Extension 1a), Self Test Use Case (Extension 1a)	MainWindow::updateBattery, MainWindow::testBattery	Deplete the battery in the simulation and check for alerts.	The MainWindow class includes functionality to update and test the battery level, with UI alerts when the

					battery is low.
7	The system must allow starting and stopping of ECG analysis.	Power On, Regular Heartbeat (Non Shockable Scenario), Irregular Heartbeat without CPR (Shockable Scenario)	MainWindow::on_stopstart_clicked	Click the start/stop button and observe the ECG analysis.	The MainWindow class handles UI interaction to start and stop ECG analysis.
8	The system should handle automatic progression for various scenarios.	Power On, Regular Heartbeat (Non Shockable Scenario), Irregular Heartbeat without CPR (Shockable Scenario), Real-Time CPR	MainWindow::on_automaticProgression, ArrhythmiaDetector::on_automaticProgression	Activate automatic progression and observe scenario handling.	Both MainWindow and ArrhythmiaDetector contain methods to handle automatic progression through different shock scenarios.
9	GUI that resembles the AED Plus's display.	Power On, Regular Heartbeat, Irregular Heartbeat without CPR, Real-Time CPR	MainWindow, ui->setupUi	Run the application and verify the GUI layout and elements.	The GUI is designed to resemble the AED Plus's display, including all necessary visual elements.
10	Real-time CPR feedback display.	Real-Time CPR	CPRFeedback, MainWindow::updateCPR	Simulate CPR and observe feedback on	Visual elements for real-time CPR

			Data	the GUI.	feedback are implemented and displayed on the GUI.
11	Cardiac arrhythmia diagnosis display.	Power On, Regular Heartbeat, Irregular Heartbeat without CPR	ArrhythmiaDetector::analyzeECG, MainWindow::updateHeartRateDisplay	Input ECG data and check for displayed diagnosis results.	The system simulates cardiac arrhythmia diagnosis and displays results on the screen.
12	Device status indicators.	Power On, Self Test Use Case	MainWindow::updateBattery, MainWindow::selffail	Observe the GUI for status updates during operation.	The GUI includes indicators for battery level and electrode placement status.
13	Simulated user interaction controls.	User Interaction	MainWindow, ui	Interact with the simulated controls and observe the system's response.	The GUI provides simulated controls for user interaction such as buttons for electrode placement.
14	Display panel for ECG waveform and device status.	Power On, Regular Heartbeat, Irregular Heartbeat without CPR	ECGDisplay, MainWindow::updateECG	Run ECG simulation and verify the waveform and status display.	A display panel is implemented to show the simulated ECG waveform and device

					status.
15	Module for diagnosing shockable and non-shockable rhythms.	Cardiac Arrhythmia Detection	ArrhythmiaDetector::deliverShock, ArrhythmiaDetector::analyzeECG	Simulate different rhythms and verify the analysis and shock advice.	A module is developed to diagnose cardiac arrhythmias and guide the user for shockable and non-shockable rhythms.
16	Visual and textual feedback for CPR.	Real-Time CPR	CPRFeedback::calculateFeedback, CPRFeedback::feedbackUpdated	Perform CPR in the simulation and observe feedback.	The system simulates the assessment of chest compressions and provides visual and textual feedback.
17	Visual prompts for simulated cardiac emergencies.	Visual Prompts	MainWindow::proceedToNextStep, MainWindow::on_automaticProgression	Run the simulation and verify that visual prompts are displayed at appropriate times.	Visual prompts are specified to guide users through a simulated rescue situation.
18	Interactive system for AED operation.	User Interaction	MainWindow, ui	Use the GUI to perform actions and verify the system's response.	An interactive system is implemented allowing users to perform actions as if operating a

					real AED.
19	Scenarios for testing software with varying conditions.	Simulated Scenarios	MainWindow::randomizeAndHandleScenario, MainWindow::Shock	Run different scenarios and observe outcomes.	Scenarios with varying patient conditions and outcomes are developed to test the software's response.