COMP 3004B Group Project

Due 16th April

Arman Ahmed 101202927

Ishraq Tanzim 101179374

Zahin Khondoker 101151004

Use Case Model: HRV Project

Actors:

- Patient (User)
- Clinician (User)
- HRV Monitoring System
- User Interface
- Battery Monitor
- Breath Pacer
- Heart Rate Monitor
- Session History Manager
- Exception Handler
- System Administrator (User)
- Notification System
- Data Storage

Use Cases:

1. Power On/Off System
   - Primary Actor: Patient
   - Supporting Actor: HRV Monitoring System
   - Precondition: HRV Monitoring System is either in the off or on state
   - Postcondition: HRV Monitoring System is turned on or off
   - Flow of Events:
     - Patient presses the power button
     - HRV Monitoring System turns on or off
2. Start/Stop Session
   - Primary Actor: Patient
   - Supporting Actor: HRV Monitoring System
   - Precondition: HRV Monitoring System is turned on and ready for a session
   - Postcondition: A session is started or stopped
   - Flow of Events:
     - Patient presses the start/stop session button
     - HRV Monitoring System starts or stops the HRV session
3. Monitor Battery Level
   - Primary Actor: Battery Monitor
   - Supporting Actor: User Interface
   - Precondition: HRV Monitoring System is turned on
   - Postcondition: Battery level is displayed and updated on User Interface
   - Flow of Events:
     - Battery Monitor continuously monitors battery level
     - Battery Monitor sends battery level data to User Interface
     - User Interface displays battery level and updates it during treatment
4. Display Breath Pacer

- ○ Primary Actor: Breath Pacer
- ○ Supporting Actor: User Interface
- ○ Precondition: HRV session is in progress
- ○ Postcondition: Breath pacer is displayed on the User Interface
- ○ Flow of Events:
  - ■ Breath Pacer calculates breath pacing during the HRV session
  - ■ Breath Pacer sends breath pacing data to User Interface
  - ■ User Interface displays breath pacer

5. Show HR Contact
   - ○ Primary Actor: Heart Rate Monitor
   - ○ Supporting Actor: User Interface
   - ○ Precondition: HRV session is in progress
   - ○ Postcondition: HR contact is displayed on the User Interface
   - ○ Flow of Events:
     - ■ Heart Rate Monitor continuously monitors HR contact during the HRV session
     - ■ Heart Rate Monitor sends HR contact data to User Interface
     - ■ User Interface displays HR contact

6. Manage Session History
   - ○ Primary Actor: Clinician, User
   - ○ Supporting Actor: Session History Manager
   - ○ Precondition: HRV Monitoring System is turned on
   - ○ Postcondition: Session history is saved, retrieved, or deleted
   - ○ Flow of Events:
     - ■ Clinician or User requests to view, delete, or save session history
     - ■ Session History Manager performs the requested action on session history

7. Handle Exceptions
   - ○ Primary Actor: Exception Handler
   - ○ Precondition: HRV Monitoring System is turned on and an exception occurs
   - ○ Postcondition: HRV Monitoring System responds to the exception accordingly
   - ○ Flow of Events:
     - ■ Exception Handler detects an exception (e.g. loss of connection during treatment or critically low battery)
     - ■ Exception Handler responds to the exception (e.g. pausing session, alerting user, or shutting down)

8. Update HRV Monitoring System Firmware
   - ○ Primary Actor: System Administrator
   - ○ Supporting Actor: HRV Monitoring System
   - ○ Precondition: HRV Monitoring System is turned on and a firmware update is available
   - ○ Postcondition: HRV Monitoring System firmware is updated
   - ○ Flow of Events:

- ■ System Administrator initiates firmware update
- ■ HRV Monitoring System downloads and installs the update

9. Receive Notifications and Alerts
   - ○ Primary Actor: Notification System
   - ○ Supporting Actor: User Interface
   - ○ Precondition: HRV Monitoring System is turned on, and an event triggers a notification or alert
   - ○ Postcondition: Notification or alert is displayed on the User Interface
   - ○ Flow of Events:
     - ■ Notification System detects an event that requires notification or alert (e.g., low battery warning, lost connection, etc.)
     - ■ Notification System sends the notification or alert to the User Interface
     - ■ User Interface displays the notification or alert

10. Secure Data Storage and Retrieval
- ● Primary Actor: Data Storage
- ● Supporting Actors: Session History Manager, User Interface
- ● Precondition: HRV Monitoring System is turned on and requires data storage or retrieval
- ● Postcondition: Data is securely stored or retrieved
- ● Flow of Events:
  - ○ Data Storage receives a request to store or retrieve data from the Session History Manager or User Interface
  - ○ Data Storage securely stores or retrieves the requested data

Use case diagram:

HEART WAVE

Made with Visual Paradigm For non-commercial use

Actors/ Supporting Actors

Use Cases

Patient/ User — Presses on/off → Power on/off system ⟶ HRV Monitoring system turns on or off

Presses on/off / Presses start/stop

HRV Monitoring System — Presses start/stop → Start/Stop Session ⟶ A session is started or stopped

System Administrator — Initiates → Update HRV Monitoring System Firmware ⟶ HRV Monitoring System is updated

Battery Monitor — Monitors → Monitor battery level ⟶ Battery level is displayed and updated

Sends battery level data

Breath Pacer — Calculates → Display breath pacer ⟶ Breath pacer is displayed

Sends data

User Interface — Sends data → Show HR contact ⟶ HR contact is displayed

Monitors

HR Monitor — Sends notification → Receive Notifications and Alerts ⟶ Notification or Alert is displayed

Detects

Notification System — Receives request → Secure Data Storage and Retrieval ⟶ Data is securely Stored or Retrieved

Data Storage — Receives request

Receives request

Session History Manager — Performs requests → Manage Session History ⟶ Session history is saved, retrieved or deleted

Clinician /User — Requests

Exception Handler — Detects → Handle Exceptions ⟶ HRV monitoring System responds to the action accordingly

CLABar
- -*m_layout:QHBoxLayout
- -*m_coherenceLabel:QLabel
- -*m_lengthLabel:QLabel
- -*m_achievementLabel:QLabel
- -m_coherence:float
- -m_length:int
- -m_achievement:float

- +CLABar(QWidget *):explicit
- +setCoherence(float):void
- +setLength(int):void
- +setAchievement(float):void
- +getLength():int
- +getAchievement():float
- +coherenceChanged(float):void
- +lengthChanged(int):void
- +achievementChanged(float):void
- +onUpdateCoherence(double,bool):void

NewSessionView
- +*navigationBar:NavigationBar
- +*navigationPad:NavigationPad
- +*contextManager:ScreenContextManager
- -*hrrgraph:HeartRateGraph
- -*coheranceCalculator:CoherenceCalculator
- -*claBar:CLABar
- -*screen:Screen
- -*hrIndicator:HeartRateIndicator

- +NewSessionView(NavigationBar *, NavigationPad *, Screen *, QWidget *):explicit
- +~NewSessionView()
- +onPlottingFinished():void
- +changesScreenContext(ScreenContext):void
- -endSession():void
- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void
- -onBackButtonClicked():void
- -onMenuButtonClicked():void
- -onHeartRateSensorStateChanged(bool):void
- #keyPressEvent(QKeyEvent *event)

BreathPacer
- -x:double
- -bmThread:BreathMeter

- +BreathPacer(QWidget);
- +~BreathPacer()
- +begin():void
- +end():void
- +BreathPacer(const BreathPacer &)
- &operator BreathPacer
- -onUpdateBreathPacer(double):void
- #paintEvent(QPaintEvent):virtual void

BreathMeter
- -m_stop:bool

- +BreathMeter(QObject)
- +run():void
- +stop():void
- +updateBreathPacer(double):void

CoherenceCalculator
- +heartRateValues:QVector<double>
- +*currentCoherenceResult:HRV_Coherence_Result
- +coherenceValues:QVector<double>
- -*claBar:CLABar
- -*coherenceBar:CoherenceBar
- -calculateHRV_Coherence(QVector<double>):void
- -_inc:int
- -_coherenceTempSum:double

- *CoherenceCalculator(CLABar *, CoherenceBar *, QObject *)

- +run():void
- +updateCoherenceValue(double,bool):void
- +onUpdateData(double,double):void

HeartRateIndicator
- -m_state:HeartRateSensor

- +HeartRateIndicator(QWidget *):explicit
- +sizeHint():QSize
- +setState(HeartRateSensor):void
- +isActive():bool
- +heartRateSensorStateChanged(bool):void
- #paintEvent(QPaintEvent *):void

HeartRateGraph
- +*p:Plotter
- +x:QVector<double>
- +y:QVector<double>
- -*mPlot:QCustomPlot

- +HeartRateGraph(QWidget *)
- +~HeartRateGraph()
- +begin():void
- +end():void
- -plottingFinished():void
- -onStopSession():void
- -onStartSession():void
- -onUpdateData(double,double):void
- -onPlottingFinished():void

ScreenContextManager
- +ScreenContextManager(QObject *):explicit
- +~ScreenContextManager()
- +onScreenContextChange(ScreenContext context):void

MainWindow
- +*settingsConfig:SettingsConfig
- -*deviceConfig:DeviceConfig
- -*screen:Screen
- -*hrIndicator:HeartRateIndicator
- -*ui:MainWindow

- +MainWindow(QWidget *parent=nullptr)
- +~MainWindow()

CoherenceBar
- -m_coherenceState:CoherenceState

- +CoherenceBar(QWidget):explicit
- +setCoherenceState(CoherenceState):void
- +onUpdateCoherence(double,bool):void
- #paintEvent(QPaintEvent):void

«enumeration»
ScreenContext
- ChallengeLevel
- BatteryLevel
- HeartRateSensor
- CoherenceState
- SessionsSimulation

SettingsConfig
- +breathDuration:int
- +challengeLevel:ChallengeLevel
- -settingsFilePath:QString

- +SettingsConfig(QObject *):explicit
- +writeToFile():void
- -onBreathDurationChanged(int):void
- -onChallengeLevelChanged(ChallengeLevel):void

Screen
- -*navBar:NavigationBar
- -*navPad:NavigationPad
- -*contextManager:ScreenContextManager
- -*screenLayout:QVBoxLayout
- -*hrIndicator:HeartRateIndicator
- -loadedLogfileName:QString
- -fromLogView:bool
- -selectedLogIndex:int
- -selectedMenuIndex:int
- -initializeLayout(QLayout *screenLayout):void
- -clearLayout(QLayout *layout,bool deleteWidget=true):void

- +Screen(NavigationBar *, HeartRateIndicator *, QWidget *):explicit
- +~Screen()
- +loadLogFile(QString _logfileName):void
- +setFromLogView(bool _value):void
- +setSelectedLogIndex(int _index):void
- +getSelectedLogIndex():int
- +setSelectedMenuIndex(int _index):void
- +getSelectedMenuIndex():int
- -initializeLayout(QLayout *screenLayout):void
- -clearLayout(QLayout *layout,bool deleteWidgets=true):void
- -onChangeScreen(ScreenContext context):void

DeviceConfig
- +batteryLevel:BatteryLevel
- -hrSensorState:HeartRateSensor
- -deviceConfigFilePath:QString

- +DeviceConfig(QObject *):explicit
- +writeToFile():void
- -onBatteryLevelChanged(BatteryLevel):void
- -onHeartRateSensorStateChanged(HeartRateSensor):void

BatteryIndicator
- -m_batteryLevel:BatteryLevel

- +batteryIndicator(QWidget *parent=nullptr):explicit
- +setBatteryLevel(BatteryLevel level):void
- #paintEvent(QPaintEvent *event):void

Plotter
- -mode:SessionSimulation
- -m_stop:bool

- +Plotter(SessionSimulation _mode =SessionSimulation::GoodCoherence, QObject *parent = nullptr):explicit
- +run():void
- +stop():void
- +cycleCoherenceMode():void

MainMenuView
- -*listview:QListView
- -*model:QListStringModel
- -*navigationBar:NavigationBar
- -*navigationPad:NavigationPad
- -*screen:Screen

- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void
- -onBackButtonClicked():void
- -onMenuButtonClicked():void

LogView
- -*listView:QListView
- -*fileList:QFileInfoList
- -*fileNames:QStringList
- -*model:QStringListModel
- -*navigationBar:NavigationBar
- -*navigationPad:NavigationPad
- -*screen:Screen
- -isEmpty:bool

- +LogView(NavigationBar *, NavigationPad, Screen *, QWidget *)
- -deleteLogs():void
- +changesScreenContext(ScreenContext)
- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void
- -onBackButtonClicked():void
- -onMenuButtonClicked():void

SettingsView
- -*resetToDefaultLabel:QLabel
- -*breathDurationLabel:QLabel
- -*challengeLevelLabel:QLabel
- -*mainLayout:QVBoxLayout
- -*listView:QListView
- -*settingsConfig:SettingsConfig
- -breathDuration:int
- -challengeLevels:QStringList
- -*model:QStringListModel
- -selectedRow:int
- -selectedChallengeLevelIndex:int
- -*navigationBar:NavigationBar
- -*navigationPad:NavigationPad

- +SettingsView(NavigationBar *, NavigationPad *, QWidget *):explicit
- +goBack():void
- +changeScreenContext(ScreenContext):void
- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void
- -onBackButtonClicked():void
- -updateLabels():void

NavigationBar
- -*m_layout:QHBoxLayout
- -*m_backButton:QPushButton
- -*m_menuButton:QPushButton

- +NavigationBar(QWidget *):explicit
- ~NavigationBar()
- +onBackButtonClicked():void
- +onMenuButtonClicked():void

NavigationPad
- -*gridLayout:QGridLayout
- -*upButton:QPushButton
- -*downButton:QPushButton
- -*leftButton:QPushButton
- -*rightButton:QPushButton
- -*centerButton:QPushButton

- +NavigationPad(QWidget *):explicit
- +upButtonClicked():void
- +downButtonClicked():void
- +leftButtonClicked():void
- +rightButtonClicked():void
- +centerButtonClicked():void
- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void

SummaryView
- -*sessionLabel:QLabel
- -*challengeLabel:QLabel
- -*lowCoherenceLabel:QLabel
- -*mediumCoherenceLabel:QLabel
- -*highCoherenceLabel:QLabel
- -*averageCoherenceLabel:QLabel
- -*sessionLengthLabel:QLabel
- -*achievementsScoreLabel:QLabel
- -*mPlot:QCustomPlot
- -logfileName:QString
- -fromLogView:bool
- -summaryData:SummaryData

- +SummaryView(QString, NavigationBar *, QWidget *):explicit
- +SummaryView(SummaryData *, NavigationBar *, QWidget *):explicit
- +SummaryView(NavigationBar *, QWidget *):explicit
- +writeToFile():void
- +getNextLogNumber():int
- +changesScreenContext(ScreenContext):void
- -initializeSummaryView():void
- -setupUi():void
- -updateLabels():void
- -populateFromLoadedFile():void
- -populateFromLatestFile():void
- -onBackButtonClicked():void
- -onMenuButtonClicked():void

InitSessionView
- -*centralLayout:QHBoxLayout
- -*m_label:QLabel
- -*navigationBar:NavigationBar
- -*heartRateIndicator:HeartRateIndicator
- -*navigationPad:NavigationPad

- +InitSessionView(NavigationBar *, NavigationPad *, QWidget *):explicit
- +changesScreenContext(ScreenContext):void
- -onUpButtonClicked():void
- -onDownButtonClicked():void
- -onLeftButtonClicked():void
- -onRightButtonClicked():void
- -onCenterButtonClicked():void
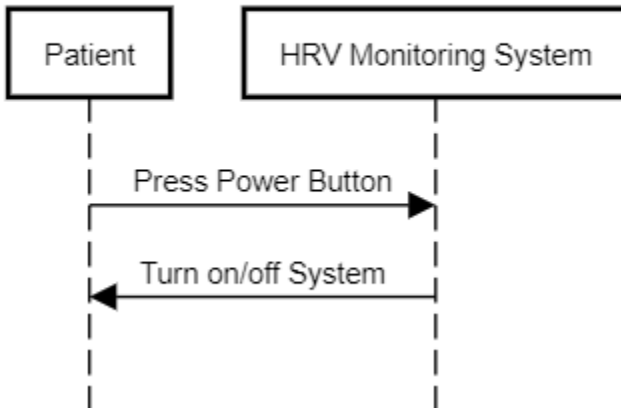- -onBackButtonClicked():void
- -onMenuButtonClicked():void
- #keyPressEvent(QKeyEvent *):void

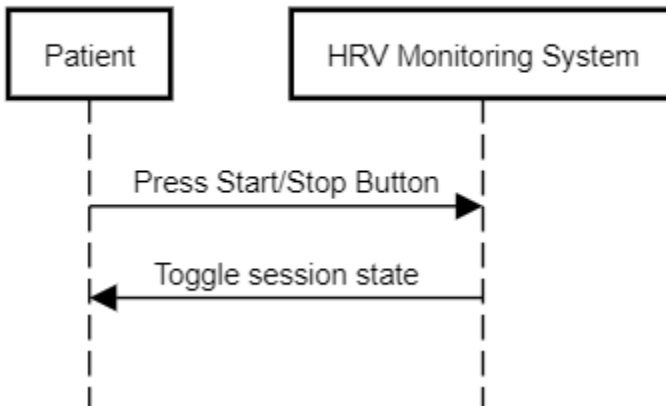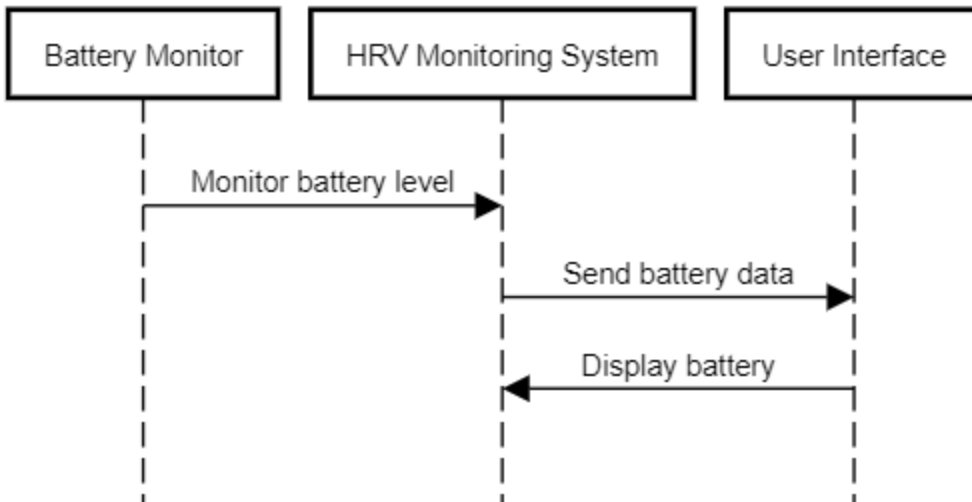## Power On/Off System

1. The patient presses the power button, and the HRV Monitoring System responds by toggling its power state between on and off.
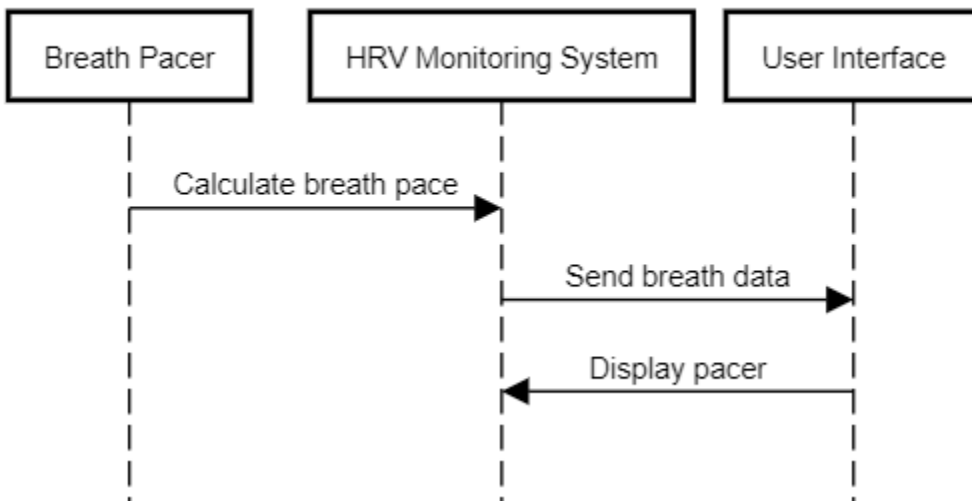
Start/Stop Session

2. The patient presses the start/stop button, and the HRV Monitoring System starts or stops an HRV session accordingly.

Monitor Battery Level

3. The Battery Monitor continuously monitors the battery level and sends the data to the User Interface, which then displays and updates the battery level information.

Display Breath Pacer

4. The Breath Pacer calculates the breath pacing during an HRV session and sends the data to the User Interface, which then displays the breath pacer for the patient.

Show HR Contact

Heart Rate Monitor	HRV Monitoring System	User Interface

Monitor HR contact →

Send HR contact →
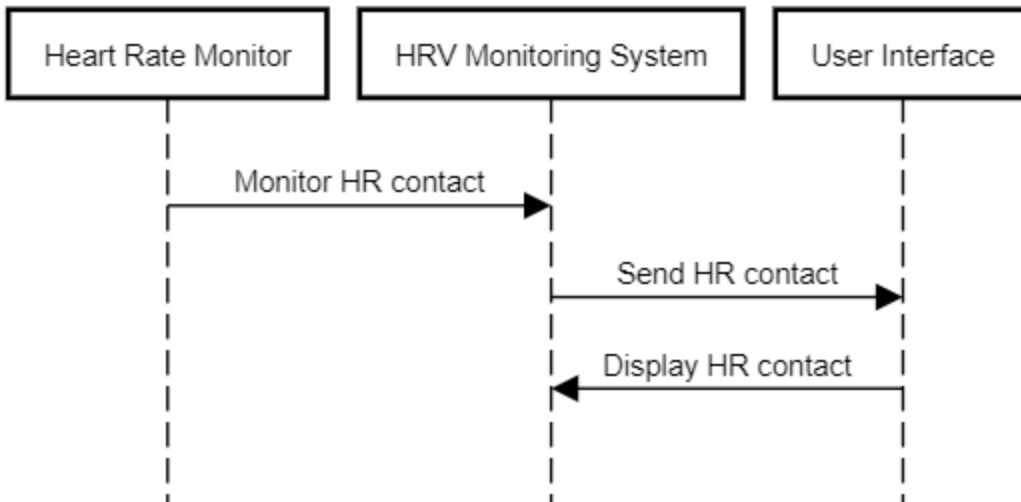
← Display HR contact

5. The Heart Rate Monitor continuously monitors HR contact during the HRV session and sends the data to the User Interface, which then displays the HR contact information for the patient.

Manage Session History

Clinician/User	Session History Manager

Request session action →
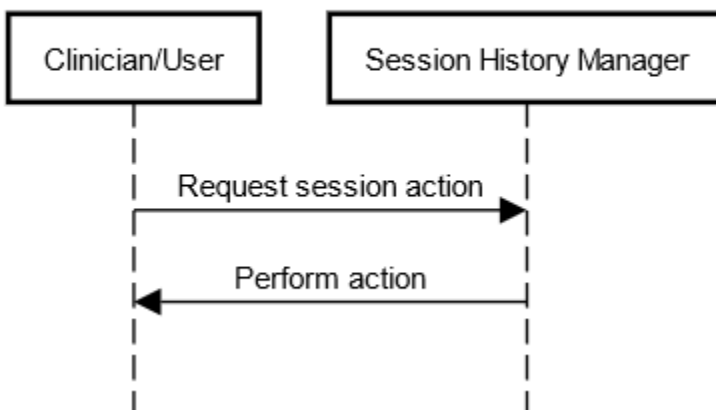
← Perform action

6. The clinician or user requests the Session History Manager to view, save, or delete session history. The Session History Manager then performs the requested action accordingly.

Handle Exceptions
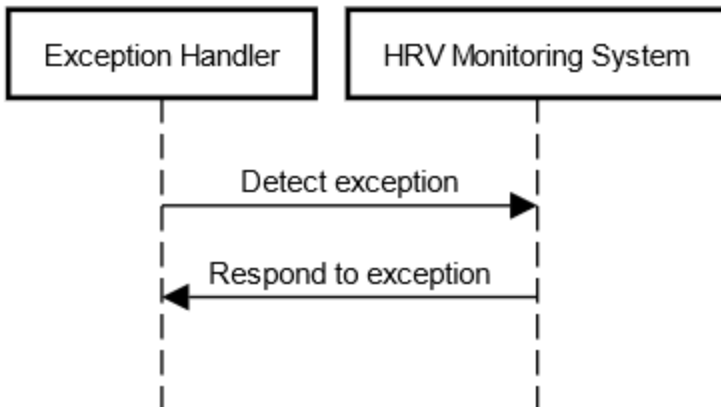
7. The Exception Handler detects any exceptions that occur during an HRV session and responds accordingly, such as pausing the session, alerting the user, or shutting down the system.

Update HRV Monitoring System Firmware
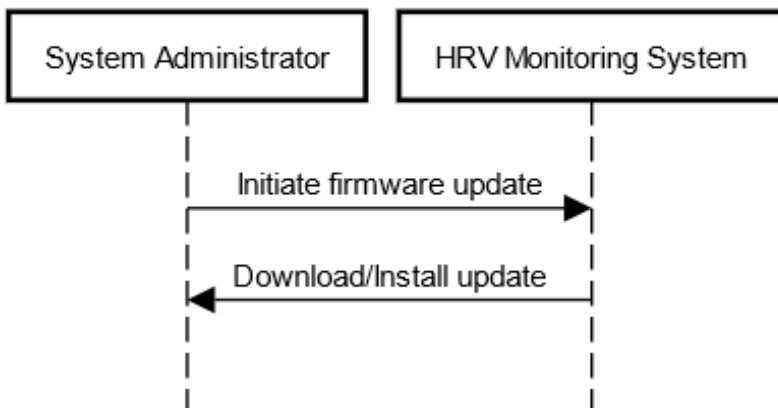
8. The System Administrator initiates a firmware update, and the HRV Monitoring System downloads and installs the update accordingly.

Receive Notifications and Alerts
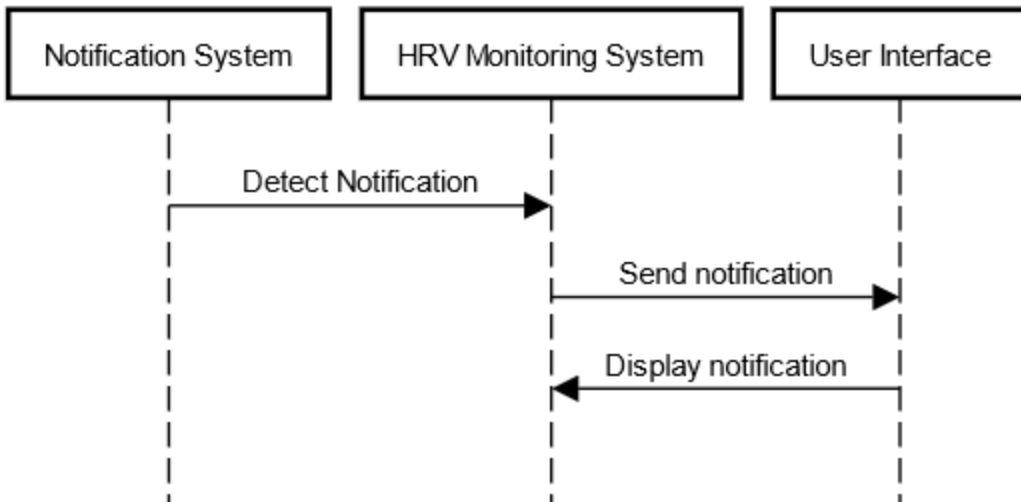
9. The Notification System detects events that require notifications or alerts (e.g., low battery warning, lost connection) and sends the information to the User Interface, which then displays the notification or alert to the user.

Secure Data Storage and Retrieval
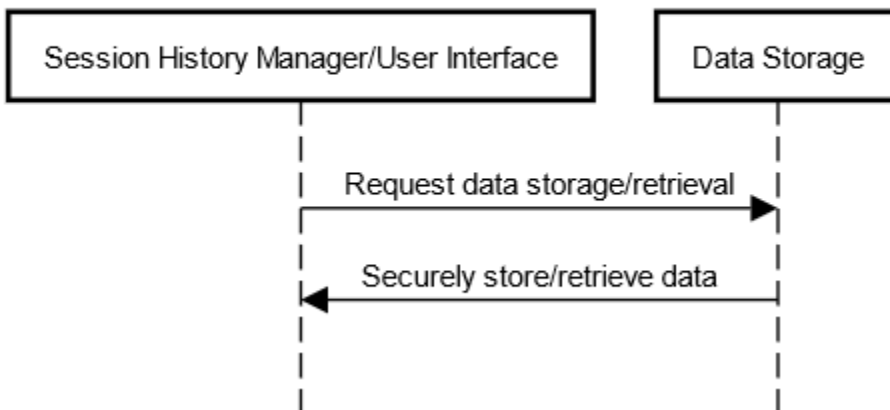
10. The Session History Manager or User Interface sends a request to the Data Storage component to either store or retrieve data. The Data Storage component then securely stores or retrieves the requested data accordingly.
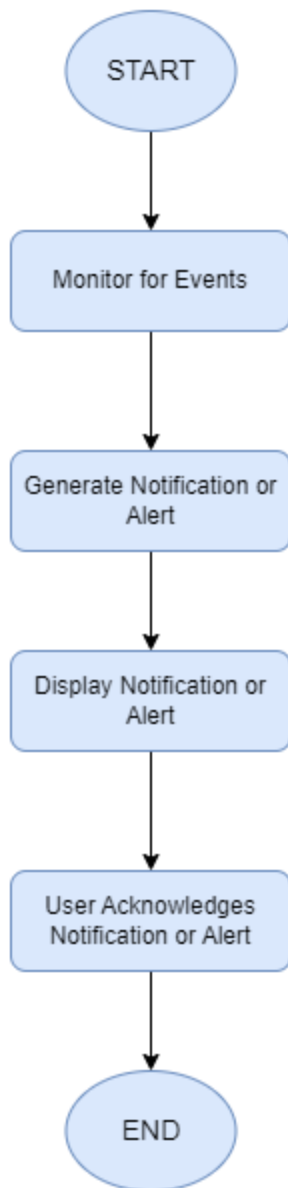
Manage Session History

Start

Request Session
History Action

Perform Action on
Session History

Save

View

Delete

Save Session
History

Display Session
History

Delete Session
History

End

Update HRV Monitoring System Firmware

START

Check for Firmware Update

Download Firmware Update

Install Firmware Update

Restart HRV Monitoring System

END
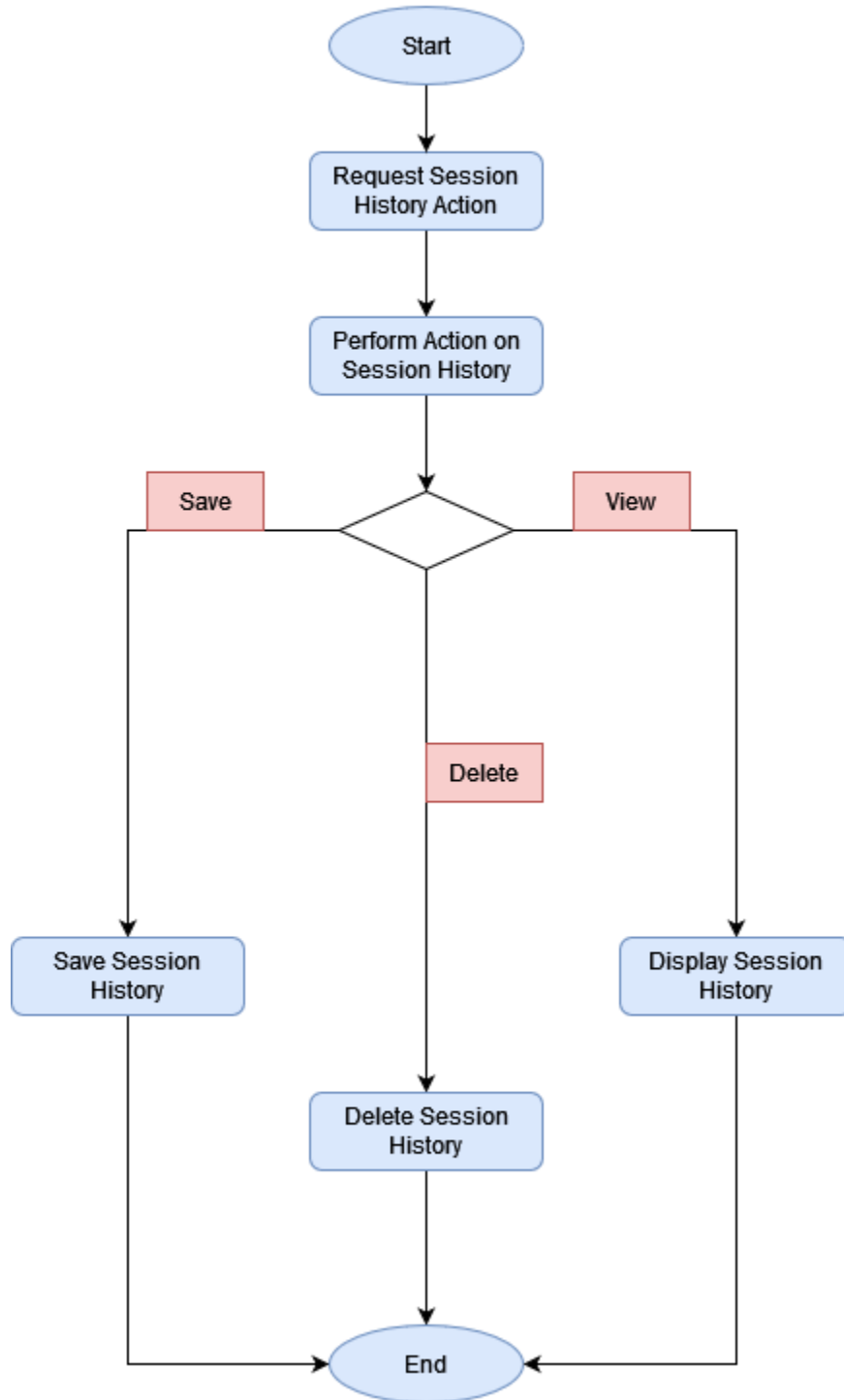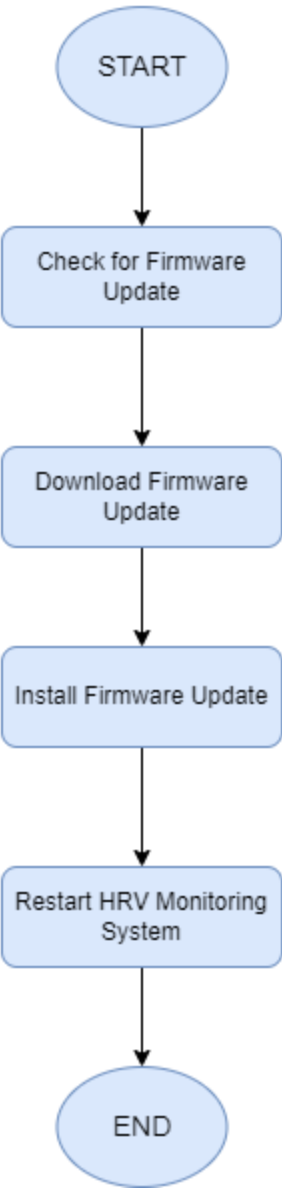
Receive Notification and Alerts

```
        ┌─────────┐
        │  START  │
        └─────────┘
             │
             ▼
    ┌───────────────────┐
    │ Monitor for Events│
    └───────────────────┘
             │
             ▼
    ┌───────────────────┐
    │ Generate Notification or │
    │       Alert       │
    └───────────────────┘
             │
             ▼
    ┌───────────────────┐
    │ Display Notification or │
    │       Alert       │
    └───────────────────┘
             │
             ▼
    ┌───────────────────┐
    │ User Acknowledges │
    │ Notification or Alert │
    └───────────────────┘
             │
             ▼
        ┌─────────┐
        │   END   │
        └─────────┘
```

Traceability Matrix:

ID	Related Use Case	Implemented By	Fulfilled By	Test	Tested By	Description
1	1. Power on HRV Monitoring System	UI, Power Manager	User	Power on test	User	Verify the system powers on correctly
2	2. Power off HRV Monitoring System	UI, Power Manager	User	Power off test	User	Verify the system powers off correctly
3	3. Start HRV session	UI, Session Manager	User	Start session test	User	Verify the HRV session starts and runs successfully
4	4. Stop HRV session	UI, Session Manager	User	Stop session test	User	Verify the HRV session stops successfully
5	5. Monitor Battery Level	UI, Battery Monitor	User	Battery level test	User	Verify battery level is displayed and updates correctly
6	6. Display Breath Pacer	UI, Breath Pacer	User	Breath pacer test	User	Verify breath pacer is displayed and functions correctly
7	7. Display HR Contact	UI, HR Contact	User	HR contact test	User	Verify HR contact is displayed and updates correctly

8	8. Update HRV Monitoring System Firmware	UI, System Administrator, HRV Monitoring System	User	Firmware update test	User	Verify system firmware updates successfully
9	9. Receive Notifications and Alerts	UI, Notification System	User	Notification test	User	Verify notifications and alerts display correctly
10	10. Secure Data Storage and Retrieval	Data Storage, Session History Manager, UI	User	Data storage test	User	Verify data is securely stored and retrieved