IoT HomeCare System SRS

Hristian Vitrychenko
Nikki Constancon
Juan du Preez
Gregory Austin
Marthinus Richter

August 10, 2017

1   Introduction

1.1   Purpose

The purpose of this document is to put forth a description detailing the ReVA(Revolutionary Vitality Analyser) system. It will explain the main purpose of the system, as well as additional subsystems, the interface of these systems, and what they will and will not do, as well as the constraints. Providing a detailed requirement specification of the system as a whole. It is intended for the client, as well as developers who will integrate the system.

1.2   Scope

The product that this systems requirements specification is describing is known as ReVA (Revolutionary Vitality Analyser). ReVA is an IoT Homecare System meant to monitor bed-ridden patients in terms of certain vital measurements such as heart rate, blood pressure, and temperature, to name but a few. ReVA analyses and collects real-time vitality data from the patient, stores it in a remote database, and reports it (either as realtime data or historical statistics). ReVA is also capable of alerting caretakers and medical professionals if there are fluctuations from the patients norms, so as to facilitate quick medical responses if necessary.

ReVA is not meant to replace true medical care, but enhance it by simplifying the monitoring of patients and optimizing the response time in case of emergencies. As ReVA is a monitoring and reporting system, it is incapable of aiding a patient's recovery. As such, a caretaker is necessary for the ReVA system to be optimally utilised. ReVA may be used for any bedridden patients such as the elderly, injured, comatose, pregnant, or simply sick. As it is a system with small hardware and wireless, attachable sensors, ReVA is comfortable and easy to use, providing a stress free means to keep track of a patient without hindering much of their comfort with wires and constant check-ups.

ReVA is also meant to bring clarity to family members and reduce stress as they can always monitor the patient so long as the patient is attached to ReVA. ReVA is a dedicated and persistent system that will do its best to keep an eye on you and see you safely back to a healthy state.

1.3   Definitions, Acronyms, and Abbreviations

1.3.1   Definitions

- Account:
  A profile that is associated with individual that will make use of, and or provide services to the system.

- Account Holder:
  An individual that is the owner of a Account.

- Back End:
  Includes the server, the communication with the client and the database management.

- Cloud:
  A decentralized serves that is related to Back End functionality.

- End User:
  The individual that will make use of the product.

- Front End:
  This includes an Android user interface and a rudimentary interface on the Raspberry Pi.

- Invalid:
  The medical term to describe a patient that is incapacitated by illness or injury.

- Peripheral Device:
  Physical equipment that interacts with, and or monitors the environment, and individuals respectively.

- System User:
  The individual that will develop and maintain the product.

- Unsecure medium:
  A medium that is susceptible to breaches that threaten client confidentiality, data availability and the integrity of the system.

1.3.2 Abbreviations

- EUID: External Unique Identifier:
  A unique number external to the system (i.e. third party systems) that uniquely identifies an user from said external system.

- IUID: Internal Unique Identifier:
  A unique number internal to the system that uniquely identifies an user.

1.4 References

IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

This document will begin with a discussion on the different interfaces in the system and what they will entail. Then it will outline the system, including who uses the system and the constraints and dependencies that need to be considered. The main functionalities of the system will then be depicted as use-case diagrams with a traceability matrix indicating the corresponding relationships between the two. Following this will be a dissection of the requirements into sub-requirements and descriptions that state exactly what they entail. Finally, some extra requirements will be outlined that are optional goals depending on time constraints.
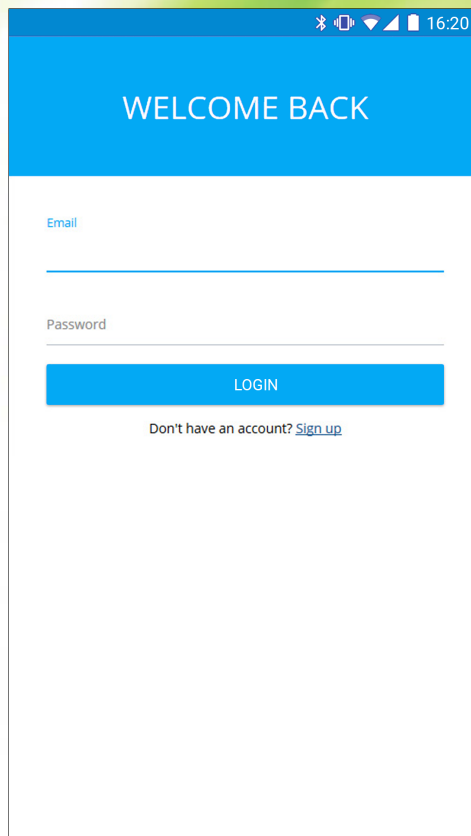
2 Overall Description

2.1 Product Perspective
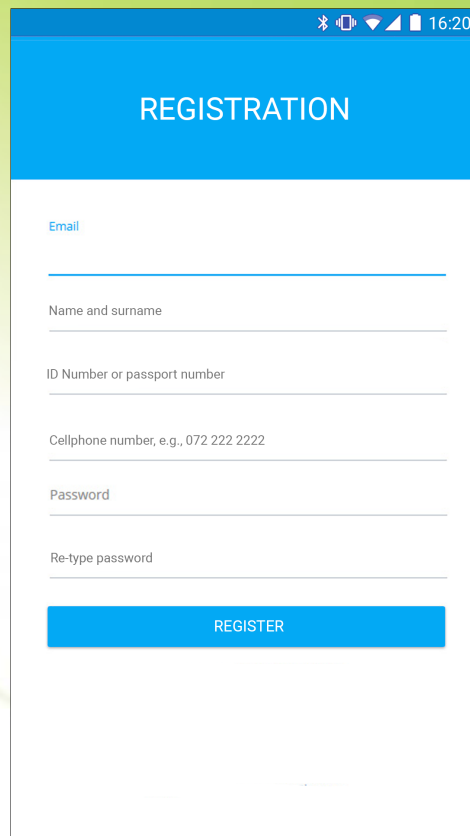
2.1.1 System Interface
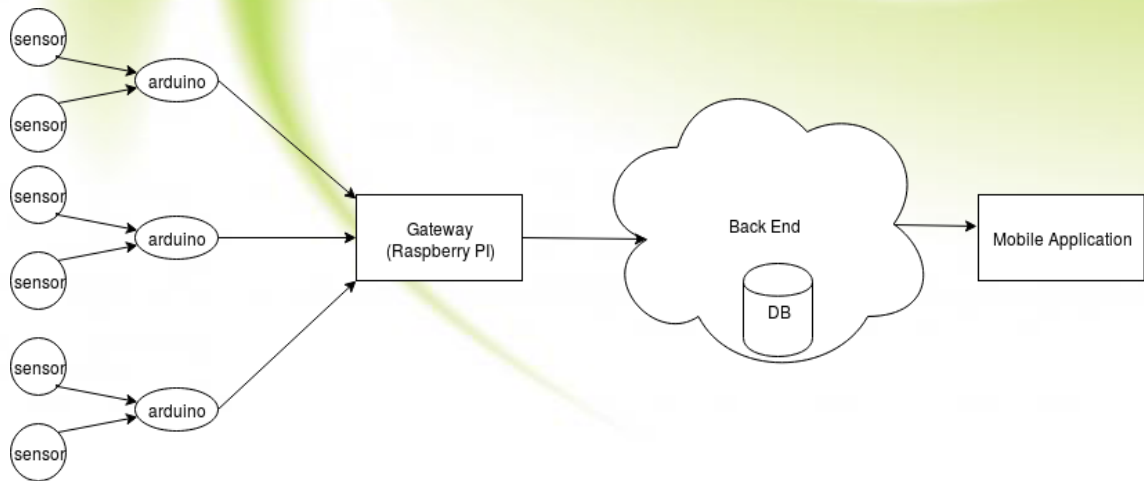
System Diagram

The diagram shows the overall system and how each component interacts with another. When reading the following details, refer to the diagram above.

Figure 1: The login screen

Figure 2: The registration screen

2.1.2   User Interface
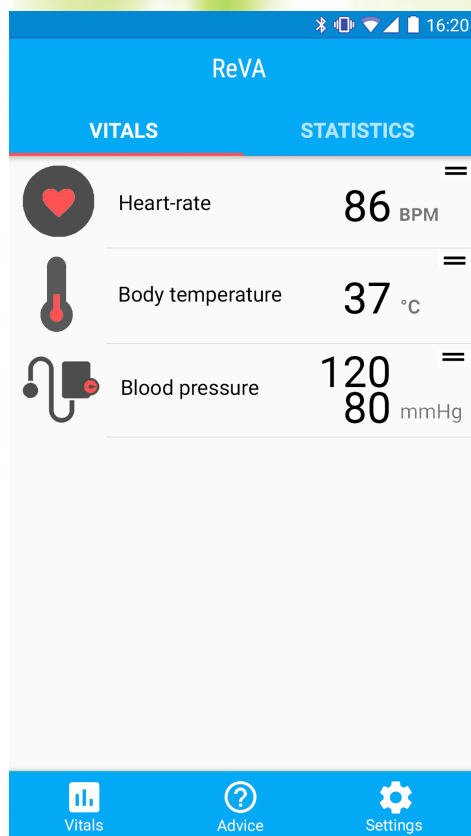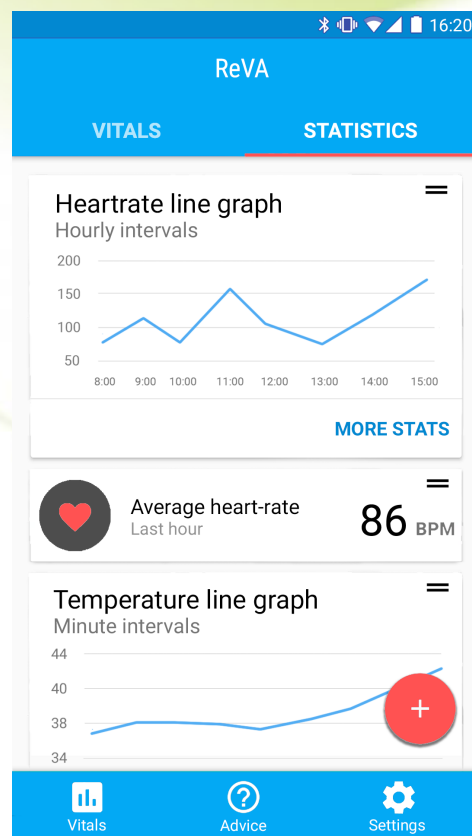
Figure 3: The vitals screen
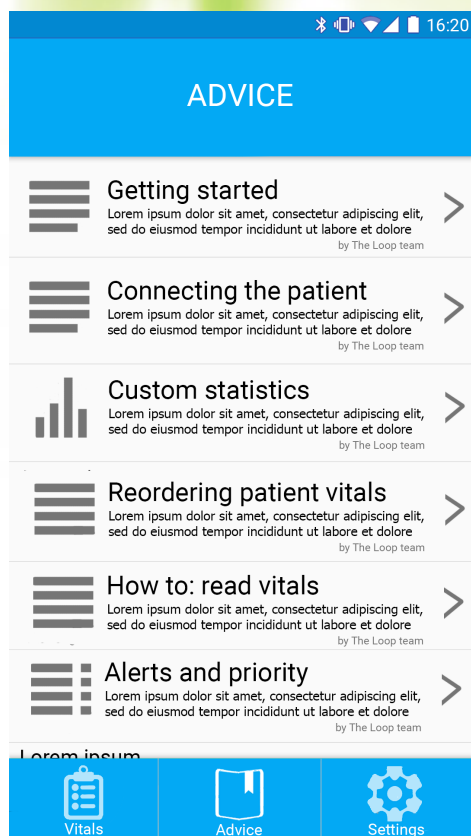

Figure 4: The statistics screen

Figure 5: The advice screen
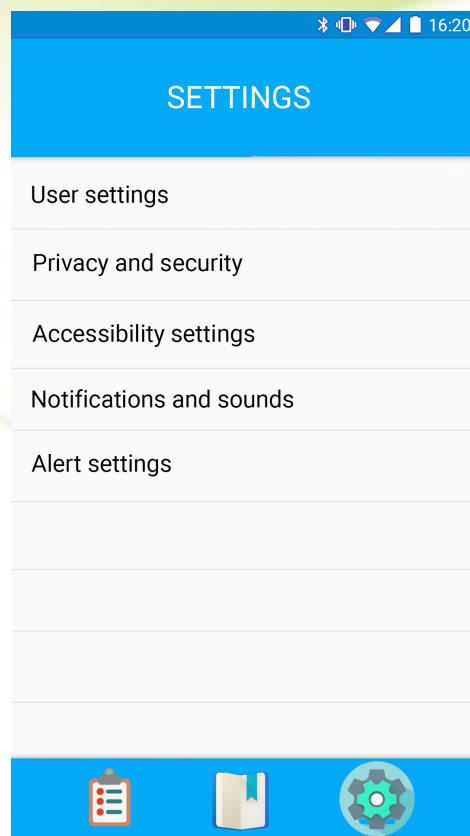

Figure 6: The settings screen

2.1.3 Hardware Interface

The hardware interface comprises of the following technologies:

- Raspberry Pi:
  The Raspberry Pi is the main medium for capturing all of the patient's vital's information. Attached Arduinos send sensory data to the Pi where it is processed, transformed and transfered to the main cloud server. The Pi keeps track of each Arduino and ensures that they all remain functional and working correctly. The Pi will be stationed in the same vicinity as the patient.

- Arduino:
  Arduinos are the devices to which the sensors are attached. Several sensors may be attached to one Arduino. Several Arduinos may be attached to one Raspberry Pi. It acts as a gateway between the sensors and the Raspberry Pi, trafficking data in a trackable manner. Helps manage sensors better.

- Sensors:
  Attached to Arduinos. The sensors comprise of several different types ranging from heart rate monitors to thermometers. They are preferably wireless but may be wired if no alternative exists. Sensors are attatched to the bed-ridden patient and report sensory data to the Arduino.

- Server:
  The server is remotely located and accessed only through the cloud. The server is where the main functionality of ReVA comes into play. It stores the majority of all subscribed patient information and analyses real-time patient info, triggering alert functionality in emergency cases. It communicates with all Raspberry Pi's, recieving patient information from all of them. It then reports the real-time and historical, statistical data to the mobile device of all subscribed members and doctors the patient or family member has approved of.

- Mobile Device:
  The android mobile application that is installed on all subscribed par-

tie's mobile devices communicates solely with the cloud server, re-
cieving patient data from it. All of which is only possible if the sub-
scribed party has access to the internet from their device. Parties
also also recieve alerts and notifications about the patient in emer-
gency situations.

2.1.4   Software Interface

The functionality of the software interface is to facilitate the communication
between the hardware infrastructure of the devices running the application
and the actual REVA application. An example of this would be the oper-
ating system of the device in use. The operating system would allow the
application to make use of the internet infrastructure to communicate the
necessary informa- tion to the application. This would enable the device
to communicate with the server and perform the necessary requests.

- **Client on Mobile**
  ReVA is compatible on mobile devices running on an Android OS.
  The mobile device must have internet capabilities to facility the up-
  dating of patient stats on the device.

- **Database on the Server**
  ReVA will communicate with a database to store real-time patient
  stats. This database will also be queried to retrieve historical stats
  for patients and to provide anonymous stats to researchers.

- **Raspberry Pi**
  ReVA will utilise a Raspberry PI to collect the data from the different
  devices. It will need to communicate the collected data with a server
  where it can be analysed and utilised as needed.

2.1.5   Communications Interface

- *Inter*-communications within the holistic system will strictly imple-
  ment the publish-subscribe pattern to facilitate the diversity in con-

nection permutations. This will allow sufficient abstraction while maintaining fine grained control, such that maintenance and overhead is minimized while scalability is improved within the context of message passing.

- – Special precautions will be made to ensure that subscribers will receive *critical* messages, or alternatively to invoke the appropriate fall back procedure in the event that said message fails to arrive at the subscriber(s).
- – Publishers will push to a single broker per logical subsystem, where messages may be handled appropriately within the context of said subsystem. The handling of messages will always include store and forward functions, and my perform sorting into priorities if it is required.
- – Both topic-based and content-based filtering will be applied within the system as to provide a grater range of potential services.

- *Intra*-communications does not have strict restrictions, though event-driven paradigms are highly favorable.

2.1.6   Memory

In terms of storage, most of the information that ReVA works with is stored on the remote cloud server in a Cassandra database after it has been formatted to adhere to standerdised medical record formats. Since the information is vast and vitally, medically important, Cassandra was chosen as it is capable of working with large amounts of data in a quick NoSQL manner with zero failure points.

In terms of local storage as data is being transfered from device to device, Arduinos temporarily store very small data from attached sensors on their local storage. As the data from sensors is not very large, it does not hinder Arduino performance. The data then moves to the Raspberry Pi, where it is also placed in local storage before it is sent through to the cloud. Once again, data is not large and does not hinder Pi performance. Information coming from the cloud server reaches the mobile device and is stored in

the mobile device's local storage. As the data that the mobile application reports is not large, local storage is sufficient and does not hinder mobile device performance.

2.1.7 Operations

Each operation will relate closely to the requirements of the system. What these operations are and what they include follows:

- An operation to gather or collect the data from the relevant devices attached to the Raspberry Pi. This operation operation will concatenate all the relevant data of a unit time of together.

- An operation to send collected data to a server.

- An operation to store the collected data to a database for analysis or retrieval in the future.

- An operation to send the collected data straight to the application in order to facilitate real-time updates

- An operation to monitor the real-time data and create relevant responses to changes in the data

- An operation to report both real-time and historical data to the application.

- An operation to to notify medical professions should an emergency situation arise

- An operation to provide advice to the caretaker as to what should be done next. This will include contact numbers for relevant medical personal and basic advice on the problem.

- An operation to handle user authentication in order to keep patient data secure.

2.2 Product Functions

The main function of the ReVA system will be to monitor a bed ridden patient's vitality and report on that patient's status.

- **Data Collection:**
  Collect sensory input from peripheral devices. This data will be formatted to the correct medical format.

- **Recording Data:**
  Put the collected data into a neatly formatted (EMR format) database for easy retrieval and easy calculation. This is to be able to provide the history of the patient's vitality as well as providing an anonymous source for researchers to analyse the data. The calculations are for the statistics that could be things like the average or a graph which tracks the data in a certain time period.

- **Reporting Real-Time Data:**
  Display real-time data from peripheral sensors on the mobile application. The application will be able to give the user as up-to-date data as possible. This means that a device data may be updated as quickly as it changes to provide accurate and real time data anyone who has authority to view the data.

- **Access:**
  Control access rights and enforcing authentication before gaining access to collected and real-time patient data. A user must be registered and logged in before being able to see any data whatsoever. An Admin person has the right to add/remove users from the system, and also grant admin rights to certain users.

- **Notifications:**
  Analyse real-time patient data and report on fluctuations from the norms and suggest possible courses of action. There will be a set of criteria that outline emergencies or things which are worthy of notification. After analysis, the severity of the emergency will be found by means of assigning it a certain code or level. Then the appropriate people, whether a professional doctor or the caretaker or someone

else, will be notified about what the stats are showing and about what the problem seems to be.

2.3 User Characteristics

The system will provide features that is specifically intended for a *patient* most likely to be an invalid. It may be assumed that *patients* will fall under the elderly demographic, whoever, system support can easily be extended to included an array of bedridden conditions.

The elderly have an tendency to be less inclined with mobile applications, and might have low technical skills regarding IoT related technologies (however this might be a gross generalization).

It is thus preferable that a family member can stand in for the Patient that has sufficient technical skills to manage said *patient's* account, or make authoritative decisions for medical procedures. This member will most likely be interested in real time monitoring facility to keep a close eye on their beloved.

The *patient* will require aid in most daily activities that may includes personal hygiene, prescription fulfillment, and minimal exercise. A *care taker* will take on the responsibility to provide this aid. Such user should have the education and technical skills to handle the mobile application without difficulties.

Elderly patients tend to develop open wounds due to bad blood circulation. As such wound nurses needs to dress the wound regularly. A nurse aught to have the expertise to work with relevant services provided by the system.

Home doctors and doctors on standby aught to have the expertise to work with relevant services provided by the system.

Researchers aught to have the expertise to work with relevant services provided by the system.

In some cases one person may assume multiple responsibilities.

To allow for loose coupling, each user will have a single account, and have a specific IUID associated with them. An user will be a logical compound of User Roles, where each role has a contextual significants.

2.3.1 User Role

Patient:
> The individual connected to medical peripherals. The peripherals will continuously monitor this individual's medical information and upload the gathered data to the Cloud. Routines will be performed on this individual by a set of Practitioners.

Invoker:
> The individual that may invoke actions of peripheral devices, given that they have surfactant authorization.

Monitor:
> The individual that has access to a specific Patient's medical information.

Practitioner:
> The individual that performs specialized routines on the Patient. The Two main specialization branches are educated and uneducated practitioners. Educated practitioners are then further specialized into Care Taker, Nurses, and Doctors.

Proxy:
> The individual that will authorize a class of procedures to be performed on the Patient, or make queries on behalf of the Patient.

Researcher:
> The individual that may view anonymous and aggregated statistics gathered by the system.

2.4   Assumptions and Dependencies

The design of this application assumes and depends on a number of things, including the following: It is assumed that the user has a android smart phone and can connect to the internet. It is assumed that the user can acquire the application and can use it from the patients household. The Raspberry Pi device must be in the patient's vicinity and connected to the internet for access to the server. The monitoring devices must be connected to the Raspberry Pi (via Wi-Fi or wires).

3 Specific Requirements

3.1 Functional Requirements

3.1.1 Actor-System Interaction Models

UC1: Reporting of Real-Time Data

Precondition: Caretaker must be registered and logged in. Mobile device must be connected to the internet	
Actor: Caretaker	**System: Mobile App\Server**
	0: TUCBW the system shows the start up page
1: The user selects the patient whose data they wish to view	2: The system retrieves the current data of that patient
3: TUCEW the user sees the patients data displayed on the screen.	
Postcondition: Device must still be connected to the internet.	

UC2: A user can view historical data detailing a period of time.

Precondition: Caretaker must be registered and logged in. Mobile device must be connected to the internet. Devices must be connected to the Raspberry Pi.	
Actor: Caretaker	**System: Mobile App\Server**
0: TUCBW the user selects the patient whose data they wish to view	1: The system subscribes to that patients data
2: The user selects the device for which they wish to view the historical data	3: The system retrieves the data for that particular device.
4: The user specifies the start time and end time	5: The system extracts the relevant data and displays it on the mobile App
6: TUCEW the user sees the patients historical data displayed on the screen.	
Postcondition: Device must still be connected to the internet.	

UC3: A user can view statistical data from over a period of time.

Precondition: Caretaker must be registered and logged in. Mobile device must be connected to the internet. Devices must be connected to the Raspberry Pi.	
Actor: Caretaker	**System: Mobile App\Server**
0: TUCBW the user selects the statistics tab at the top of the screen	1: The system retrieves the statistical data from the database. It displays the data and draws graph points.
2: TUCEW the user sees the patient's statistical data displayed on the screen. This includes min, max, average and a graph depicting fluctuations.	
Postcondition: Device must still be connected to the internet.	

3.1.2   Use Cases

- UC1: A user can view real time data gathered from a patient.

- UC2: A user can view historical data detailing a period of time.

- UC3: A user can view statistical data from over a period of time.

- UC4: A user can register to create an account and link to a particular patient.

- UC5: A user can log in to his or her account.

- UC6: A user can log out of his or her account.

- UC7: A user can manage his or her account information.

- UC8: An admin user can manage other users' access rights.

- UC9: A user, caregiver, or doctor can get notifications if something is wrong according to the data.

- UC10: A user can see all previous notifications.

- UC11: A user can view general advice on the App.

- UC12: A user can add or remove medical persons' contact details to the app.

- UC13: A user can configure devices remotely.

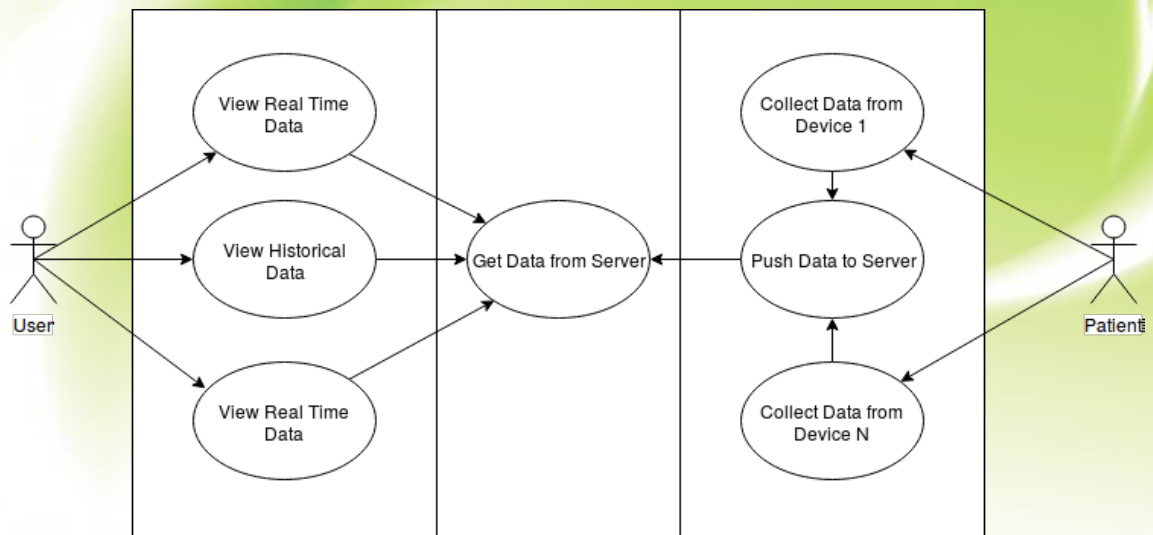- UC14: A researcher can view the data in the database anonymously.
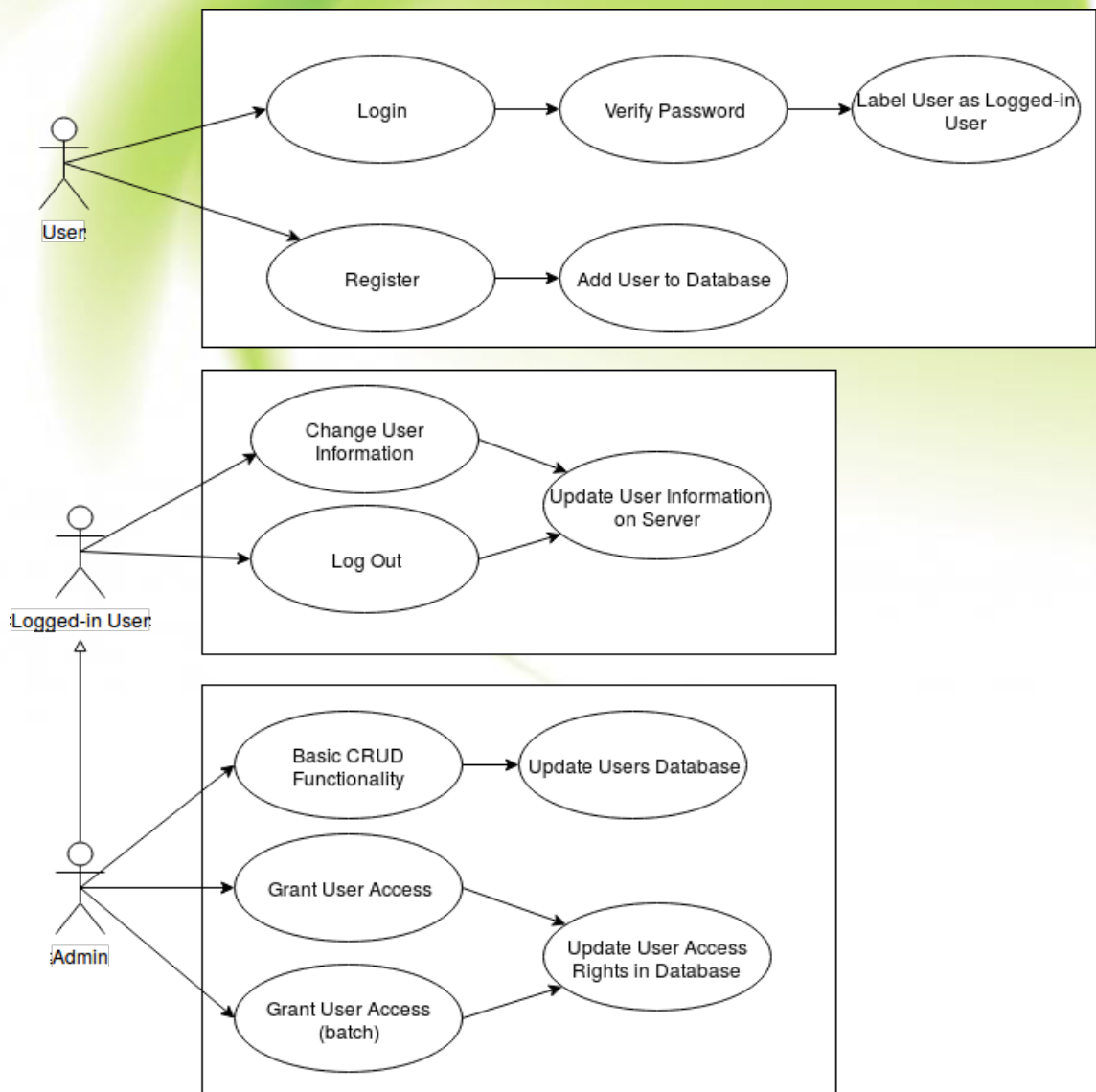
Figure 7: UC1, UC2, UC3
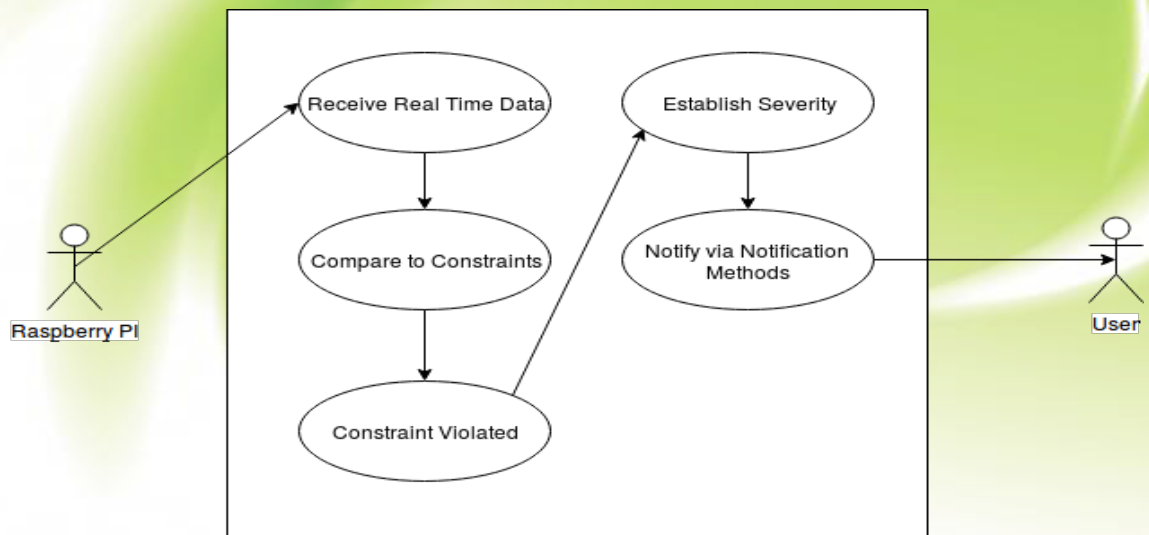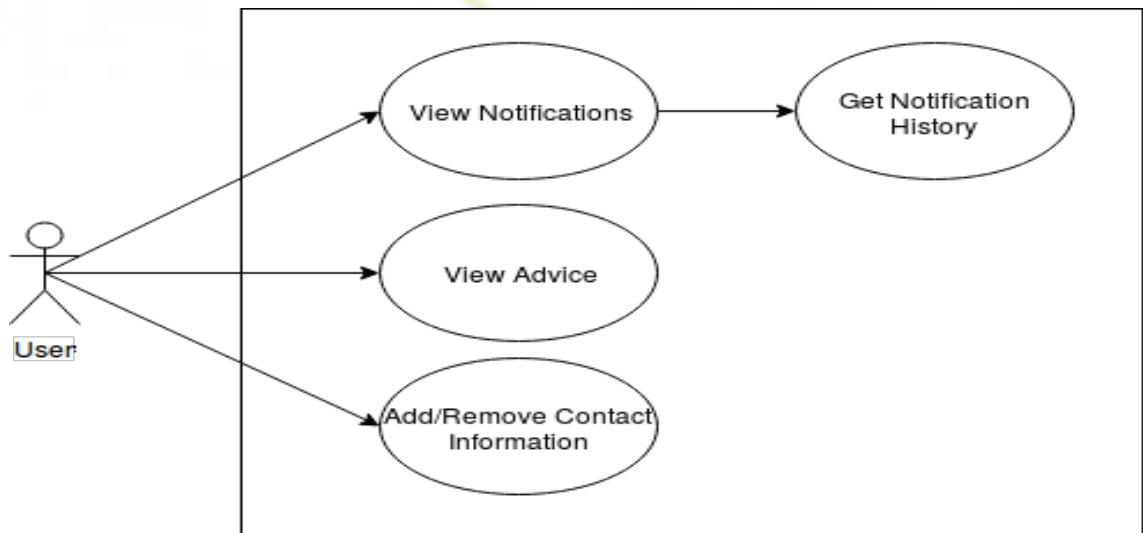
Figure 8: UC4, UC5, UC6, UC7, UC8

Figure 9: UC9
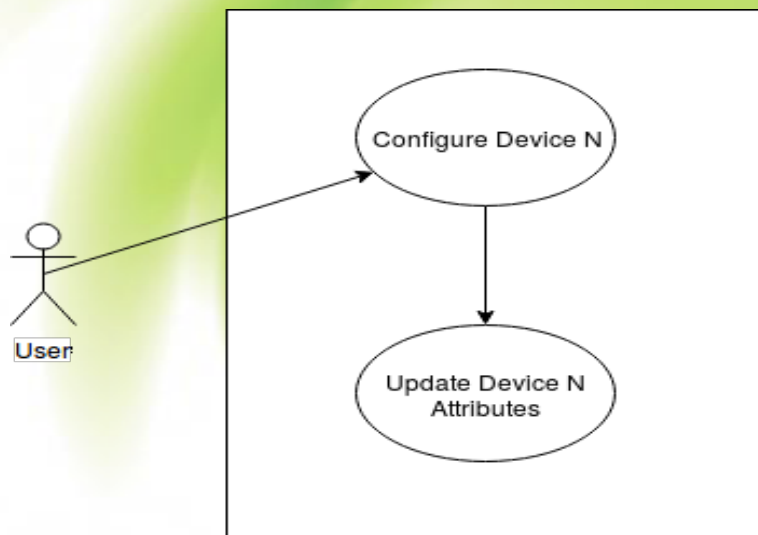

Figure 10: UC10, UC11, UC12

Figure 11: UC13

3.1.3 Functional Requirements

- RQ1: ReVA shall report data.

  - RQ1.1: ReVA shall collect patient vitality data and store it in the cloud.
  - RQ1.2: ReVA shall display real time data for each connected device.
  - RQ1.3: ReVA shall display a patient's history of data for each device.
  - RQ1.4: ReVA shall display statistical data for each device.

- RQ2: ReVA shall notify appropriate persons at appropriate times.

  - RQ2.1: ReVA shall analyse data and notify appropriate people when certain conditions are violated.
  - RQ2.2: ReVA shall keep a track of all notifications for the patient.
  - RQ2.3: ReVA shall allow a user to see the past notifications.

- RQ3: ReVA shall manage users' accounts.

  - RQ3.1: ReVA shall allow a person to register and subscribe to a single patient.
  - RQ3.2: ReVA shall allow a person to log in and out of a particular account.
  - RQ3.3: ReVA shall allow a person to manage his or her account information.
  - RQ3.4: ReVA shall allow admin users to add or remove users as well as grant admin rights to users.

- RQ4: ReVA shall have an advice/emergency section.

  - RQ4.1: ReVA shall have static advice to be consulted in an emergency.
  - RQ4.2: ReVA shall allow a user to add or remove medical professionals' contact details.

- RQ5: ReVA shall be able to configure devices remotely.

3.1.4 Traceability Matrix

Table 1: Traceability Matrix

UC \RQ	1	1.1	1.2	1.3	1.4	2	2.1	2.2	2.3	3	3.1	3.2	3.3	3.4	4	4.1	4.2	5
1	X		X															
2	X	X		X														
3	X				X													
4										X	X							
5										X		X						
6										X		X						
7										X			X					
8										X				X				
9						X	X											
10						X		X	X									
11															X	X		
12															X		X	
13															X			X
15																		

3.1.5 Access Requirements

- Product Acquisition

    – End users shall download the android application onto their phone
    via Google Play Store. At which stage the application shall as-
    sume one of two states. If the user's Google account have not
    yet been registered onto the system, then the application's shall
    enter the registration state, else the normal operational state is
    assumed.

    – Front end hardware shall be acquired by business transactions
    that is beyond the scope of this document. At the time hardware
    is deemed to be in an operational state, each components shall
    be uniquely addressable to a specific Affected user.

- User Authentication

- Users shall be uniquely identified by an IUID that shall never be transmitted over unsecured mediums.
- To allow coupling with third party authentication mechanisms, mappings shall be made between the external mechanism's account's EUID and the internal IUID.
- Only external authentication mechanisms that is sufficiently secure shall be used.
- Google Accounts shall be supported as an external authentication mechanisms.
- The internal authentication system shall be designed such that additional external identifying mechanisms can be attached with no or minimal refactoring.
- Protocols and procedures specified by relevant external authentication mechanisms shall be implemented and executed without exceptions.

- User Authorization

  - Authorization shall be managed per User Role. This will allow fine grained access control that will enhance user experience while preserving sufficient security.
  - Depending on the security level, persistent authorization shall be passively granted given that a connection is made by a certified device, or a previously authenticated user. Higher level authorization shall require additional authentication, that may vary depending on the context. Said higher authorization may include, but is not limited to, passwords, patterns, biometrics, and certificates.

- User Registration

  - Account registration shall be controlled by third party authentication mechanisms such as Google Accounts. At registration a IUID shall be allocated to the registering user and linked to the EUID.
  - Account Holders shall further register to a set of User Roles, where applicable parties shall provide authorization before registration is committed.

– Predefined collections of User Roles shall be available to simply the registration process. These collections shall include, but is not limited to; Invalid, Family Member, Care Worker, Wound Nurse, House Doctor.

– Back End Batch User Registration

  ∗ This class of registration shall be implemented to reduce tedium when large amounts of high level authorized users must be fed into the system.

  ∗ System Users shall introduce a file obtained from medical facility into the system for parsing. Said file shall contain aggregated medical staff's details.

3.1.6 Real-Time Data requirements(most of these are actually non-functional)

- Keep the data moving
  process messages 'in-stream', without any requirement to store them to perform any operation or sequence of operations.

- Query using QL on streams
  relying on low-level programming schemes results in long development cycles and high maintenance costs. Process moving real-time data using a high-level language such as SQL.

- Handle stream imperfections
  built-in mechanisms to provide resiliency against stream 'imperfections' including missing and out-of-order data.

- Generate predictable outcomes
  Time series data must be processed in a predictable manner to ensure the results of processing are deterministic and repeatable.

- Integrate stored and streaming data
  have the capability to efficiently store, modify, and access state information, and combine it with live streaming data. For seamless integration, the system should use a uniform language when dealing with either type of data.

- Guarantee data safety and availability
  ensure that the applications are up and available, and the integrity of the data maintained at all times, despite failures.

- Partition and scale applications automatically
  it should be possible to split an application over multiple machines for scalability (as the volume of input streams or the complexity of processing increases) without the developer having to write low-level code.

- Process and respond instantaneously
  a highly-optimized, minimal overhead execution engine to deliver real-time response for high-volume applications.

3.1.7 Notifications Requirements

- Real-time Monitoring of Patient Data
  Patient data received is compared to stats that have been deemed normal before the data is stored in the Cassandra database. Deviations from the norm trigger notification functionality, but data is stored as normal anyway.

- Judging severity
  Notification functionality will then assess the severity of the patient data that triggered the response and determine its severity.

- Reacting Accordingly Based On Severity
  Once severity is established, the Notification functionality will decide on the appropriate response based on the severity. For example, if the patients blood sugar level is slightly bellow average, a low level response will be issued.

- Types of Responses
  The responses that will be chosen from will range from a small notification on the application itself, to an urgent alert with sound, to an emergency SMS on the caretakers mobile devices.

- NF:Format of Notifications
  Notifications will be precise, short and as to the point as possible.

Especially for emergency alerts. They may also provide minimal advice for the specific situation but should definitely not be considered replacements for medically professional approaches. They are to be read quickly and with ease so as to minimize the time to react appropriately.

- Notify Nearest Medical Professional
  If severity is judged to be too great for a caretaker to handle, the application may notify the nearest medical facility with all needed details such as address and condition of patient.

3.2 Design Constraints

1. One and only one Raspberry Pi device per bed-ridden patient

2. The system must consistently be connected to the internet for real-time reporting to function (provided that the caretaker's mobile device is also connected to the internet)

3. Arduinos (and by extension sensors) must constantly be connected to the Raspberry Pi for the system to function (either through Wi-Fi or cables)

4. The mobile application of each subscribed member's mobile device must have access to the internet

5. Caretakers must be registered and logged in to view the patient data to which they have access privileges

6. Attached sensors must have existing functionality on the Raspberry Pi in order for their data to be processed and presented correctly

3.3 Software System Attributes

3.3.1 Accessibility

The end user application will provide a customizable color palette that enables high contrast overlays to aide slightly visually impaired users.

3.3.2 Availability

Persistent availability to the system and its functions will be guaranteed throughout its operational lifetime. Special precaution might be necessary when system updates are rolled out. Load balancing techniques can also improve tolerance to single points of failure.

3.3.3 Compatibility

The end user application will be fully compatible with Android devices that have a screen resolution of at least 800 x 480 pixels. The system will be compatible with a small subset of available medical devices on the market, that implements a common protocol.

3.3.4 Failure Transparency

The system will exhibit high failure transparency both in connection, and data persistency. The system will attempt to handle errors such that end users will only be informed if critical faults occurs. Errors will however be reported to system users.

3.3.5   Interchangeability

The system will allow front end peripheral devices to be interchanged, given that they provide equivalent functionality, and implement a common protocol.

3.3.6   Interoperability

The system will employ standardized, consistent and well documented protocols to maximize interface opaqueness, thus in turn will allow maximum interoperability with present and future systems.

3.3.7   Maintainability

The system will implement suitable design patterns to cope with a changing environment, correct defects, meet new requirements, and to make future maintenance easier. A robust testing environment will be setup to detect development stage errors. Error logging will be integrated into code to detect production stage errors.

3.3.8   Security

The system will guarantee security of the users' personal information by means of encrypted communications and storage. Administrator accounts will only have access to anonymized data such that personal information exposure is rendered impossible.

3.4 Other Requirements

3.4.1 Peripheral Control Requirements

- Peripheral control access
  Peripheral control will be accessed through the mobile application where the caretaker or whoever has access to it will have options for manipulating attached devices through a user interface.

- Processing user requests
  Once the user chooses how to manipulate a peripheral from the application, the request is sent to the cloud and relayed to the raspberry pi. There, the pi processes the request and invokes the correct function.

- Processing Requests from Pi
  Requests coming from the cloud will be decoded and the correct device will be picked out for control. Once the device is identified, the pi will pick out the appropriate requested functionality for said device and run it.

- Device Monitoring
  The status of each device will be monitored by the pi and any user requests that go out of range of set parameters for that specific device will trigger error messages.

- Error responses
  If the pi realizes that the requests are invalid, it will send the appropriate error message back through the cloud and to the users mobile application where it will be presented as an alert.

- Successful responses
  If the request fits the parameters of the device, the device will respond accordingly and a success message will be sent through the cloud and to the users mobile application, notifying them that the action was completed successfully.

3.5   Non-functional requirements

3.5.1   UI Accessibility Requirements and Constraints

- Material Design usability requirements

  - Clear
    * Clearly visible elements
    * Sufficient contrast and size
    * A clear hierarchy of importance
    * Key information discernable at a glance

  - Robust
    * Navigate: Give users confidence in knowing where they are in the app and what is important.
    * Understand important tasks: Reinforce important information through multiple visual and textual cues. Use color, shape, text, and motion to communicate what is happening.
    * Access the app: Include appropriate content labelling to accommodate users who experience a text-only version of your app.

  - Specific Assistive technology helps increase, maintain, or improve the functional capabilities of individuals with disabilities, through devices like screen readers, magnification devices. This is especially the case with IoT Homecare.

- Colour and Contrast

  - Accessible colour palette
    Choose primary, secondary, and accent colors for the app that support usability. Ensure sufficient color contrast between elements so that users with low vision can see and use the app.

  - Contrast ratios
    Contrast ratios represent how different a color is from another color, commonly written as 1:1 or 21:1
    * Small text should have a contrast ratio of at least 4.5:1 against its background.

* Large text (at 14 pt bold/18 pt regular and up) should have a contrast ratio of at least 3:1 against its background.
* Icons or other critical elements should also use the above recommended contrast ratios.

– For users who are colorblind, or cannot see differences in color, include design elements in addition to color that ensure they receive the same amount of information. Use multiple visual cues to communicate important states. Use elements such as strokes, indicators, patterns, texture, or text to describe actions and content.

- Sound and Motion

    – Sound
    Give visual alternatives to sound, and vice versa. Provide closed captions, a transcript, or another visual alternatives to critical audio elements and sound alerts.

    – Motion
    - Enable content that moves, scrolls, or blinks automatically to be paused, stopped, or hidden if it lasts more than than five seconds.
    - Limit flashing content to three times in a one-second period to meet flash and red flash thresholds.
    - Avoid flashing large central regions of the screen.

- Style

    – Touch Targets
    To help users who aren't able to see the screen properly or who have motor-dexterity problems, to tap elements in the app. Touch targets are the parts of the screen that respond to user input. They extend beyond the visual bounds of an element.
    - Touch targets should be at least 48 x 48 dp.
    - They should be be separated by 8dp of space or more to ensure balanced information density and usability.

    – Grouping Items
    Keeping related items in proximity to one another is helpful for

those who have low vision or may have trouble focusing on the screen.

- Fonts
  To improve readability users can increase font size

- Hierarchy and focus

  - Hierarchy
    Place items on the screen according to their relative level of importance.

    * Important actions: Place important actions at the top or bottom of the screen (reachable with shortcuts).
    * Related items: Place related items of a similar hierarchy next to each other.

  - Focus order
    Input focus should follow the order of the visual layout, from the top to the bottom of the screen. It should traverse from the most important to the least important item.

  - Grouping
    Group similar items under headings that communicate what the groupings are. These groups organize content spatially.

  - Transitions
    - Focus traversal between screens and tasks should be as continuous as possible.
    - If a task is interrupted and then resumed, place focus on the element that was previously focused.

- Implementation

  - Use standard platform controls that are well known and standards used in most android applications.

  - Use scalable text and a spacious layout to accommodate users who may have large text, color correction, magnification, or other assistive settings turned on.

  - Any features with special accessibility considerations should be included in help documentation. Make help documentation relevant, accessible, and discoverable.

- Writing

    - Be succinct, keep content and accessibility text short and to the point. Avoid including control type or state in text
    - Indicate what an element does, use action verbs to indicate what an element or link does, not what an element looks like, so a visually impaired person can understand.
    - Don't mention the exact gesture or interaction
    - Confirm actions, snackbars (Android) to confirm or acknowledge user actions that are destructive (like "Delete" or "Remove") or difficult to undo.

3.5.2 Performance Requirements

- Process and Respond Instantaneously
  A highly-optimized, minimal overhead execution engine to deliver real-time responce for high-volume applications.

- Generate Predictable Outcomes
  Time series data must be processed in a predictable manner to ensure the results of processing are deterministic and repeatable.

- Handle Stream Imperfections
  Built-in mechanisms to provide resiliency against stream 'imperfections', including missing and out-of-order data.

- Guarantee Data Safety and Availability
  Ensure that the application is up and available, and the integrity of the data is maintained at all times despite failures.