# Purpose

This document serves as the design document for RMSS application development. The document captures the design details through UML concepts - data model, flowcharts, dependencies etc.

# Scope

RMSS is a web based application that runs in desktop/laptop browser (typically Chrome). It allows users of Admin and Physician roles to perform clinical investigation/trials. It allows Admin to perform data entry for case report forms. It allows Physician to view the vital parameters synced from the implant device, to view symptoms and messages from patient, to control the data sync trigger.

# Reference Documents

|  |  |
| --- | --- |
| Document ID | Document Name |
|  | Software Development Life Cycle Procedure |

# Acronyms & Terminology

Refer to Glossary XXXX

|  |  |
| --- | --- |
| **Term** | **Definition** |
| DTAL | Device Twin Abstraction Layer |

# Design

## Overview

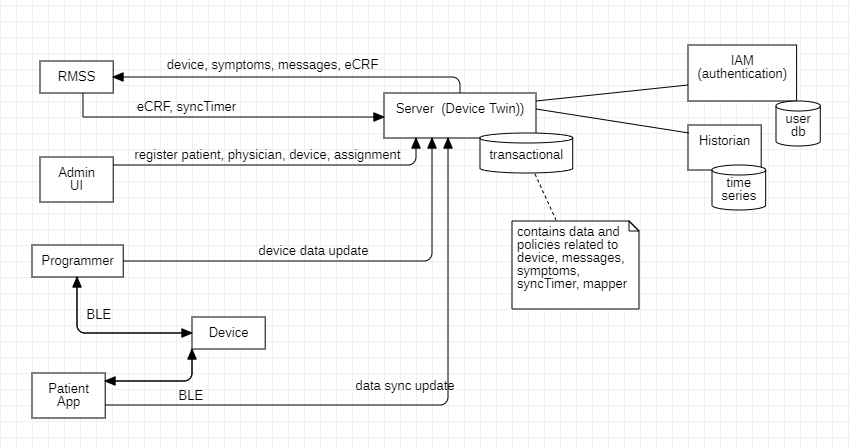


Figure 1 System overview

The above figure shows RMSS in relation with other sub-systems.

Server – This is a cloud based application that provides data agnostic device twin capabilities with authentication, authorization and historian time series.

Admin UI – This is a web application that enables admin user to register patient, physician and device (digital twin). It enables to assign a device to patient and physician. Its manages the policies (e.g. only the assigned patient and physician will have read and write access to the device twin

Programmer – This is an Android application that enables physician to view & program the device during implant and follow-up

Patient - This is an Android, iOS app that enables patient to sync the data to device twin, to report symptoms and to chat with physician

## Software Design

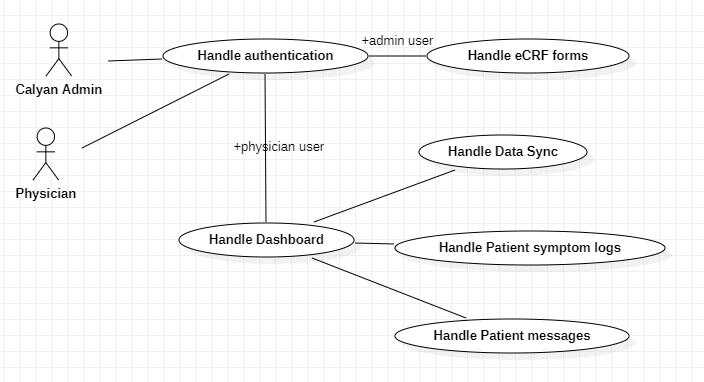


Figure 2 RMSS usecases

The above figure shows the use cases handled by RMSS. It handles two user profile specific functions

Admin – to handle data entry operations for Case Report Forms

Physician – to view data sync, to view symptom logs, to view and respond to messages from patient, to control the data sync trigger and periodicity

These use cases are realized by RMSS through the different components as shown in the following figure.

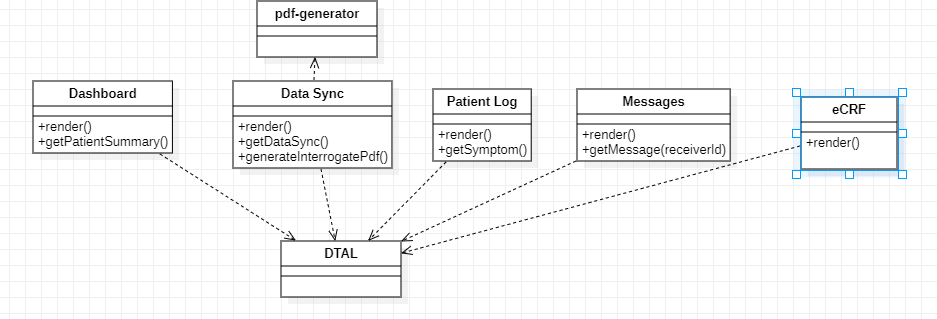


Figure 3 RMSS components

RMSS is essentially composed of UI and DTAL components.

DTAL – This is the service component that encapsulates the communication with the server to manage all the transaction data (device, symptoms, synctimer, message)

Dashboard – This is the UI component that renders the dashboard page and accessible only to physician. This is the default page with summary of data syncs from patients assigned to the given physician

Data sync – This is the UI component that renders the selected patient data sync details. This page is accessible only to physician

Patient Log – This is the UI component that renders the selected patient symptom details. This page is accessible only to physician

Message – This is the UI component that renders the message between the selected patient and the logged in physician. This page is accessible only to physician

eCRF – This is the UI component that renders tabs for different stages of Case Report form data collection. This page is accessible only to the admin user

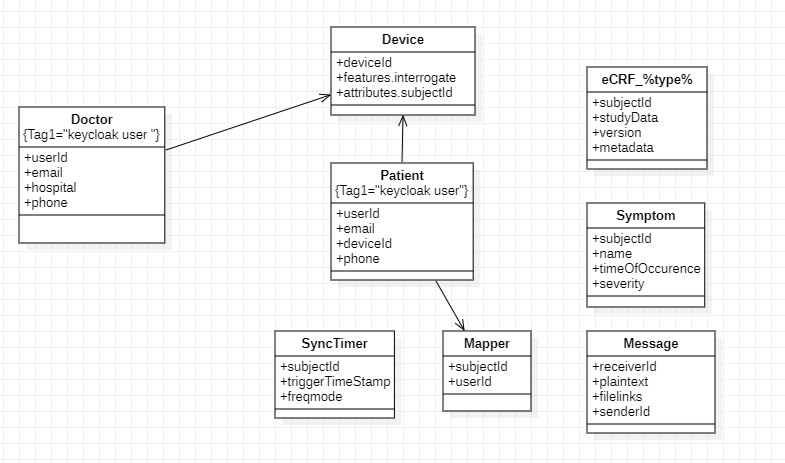
[Where appropriate, to reduce the RISK to an acceptable level, the architecture specification shall make use of:

1. COMPONENTS WITH HIGH-INTEGRITY CHARACTERISTICS
2. Fail-safe functions
3. Redundance;
4. Diversity;
5. Partitioning of functionality
6. Defensive design, e.g limits on potentially hazardous effects by restricting the available
7. The architecture specification shall take into consideration:
8. Allocation of RISK CONTROL measures to subsystems and components
9. Failure modes of components and their effect;
10. Common cause failures;
11. Systematic failures;
12. Test interval duration and diagnostic coverage;
13. Maintainability
14. Protection from reasonably foreseeable misuse;

## Conventions and Standards followed:

The UML notation and concepts are leveraged to capture the design concepts.

## Data Model Diagram:



### Device

This class represents the digital twin of pacemaker. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | string | Hospital assigned identifier to the patient to whom this device is implanted |
| serialnumber | string | Unique number assigned to the device |
| macid | string | BLE identifier of the device |
| blekey\_mfg | string | BLE key/password for Calyan admin |
| blekey\_patient | string | BLE key/password for Patient |
| blekey\_clinician | string | BLE key/password for Physician |
| deviceparameters | structure | Holds the device parameters (pacingmode, pacingrate, pacingamplitude, pacingpulsewidth, refractoryperiod, ratehysteresis, blankpostvp, blankpostvs, polarity, sensitivity) |
| Egmstream | structure | Holds the egm data (vp, vs, heartRateValues,raw) |
| Leadimpedancestream | structure | Holds the egm data (leadImpedanceValues,raw) |
| devicestatus | structure | Holds batteryvoltage, batterylife, errors |
| Devicestoreddata | structure | Holds rwave weekly amplitude (max, min), lead weekly impedance (max,min) |

### Patient

This class represents user of type patient. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| type | string | Identifies patient to whom this device is implanted |
| email | string | Login username for patient |
| phone | string | Phone number to send authorization code |
| deviceId | string | Associates patient to the implanted device |
| userId | string | IAM unique identifier assigned to the user |

### Mapper

This class represents user of type patient. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | string | Hospital assigned identifier to the patient to whom this device is implanted |
| userId | string | IAM unique identifier assigned to the user |

### Physician/Doctor

This class represents user of type clinician. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Type | string | Identifies physician or doctor |
| Email | string | Login username for physician or doctor |
| Phone | string | Phone number to send authorization code |
| hospital | string | Hospital name |
| userId | string | IAM unique identifier assigned to the user |

### eCRF\_%type%

This class represents electronic Case Report Form to capture clinical investigation data. %type% represents different stages of data collection

* enrollment
* preimplant
* implant
* followup1week
* followup2week
* followup1month
* followup3month
* followup6month
* followup9month
* followup12month
* followup18month
* adverseevent
* protocoldeviation
* devicemalfunction

It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | string | Hospital assigned identifier to the patient to whom this device is implanted |
|  |  | <refer to eCRF Requirements.xlsx for above stage specific properties> |
| Revision | number | Version number |
| metadata | string | Holds rationale for the data update |

### SyncTimer

This class represents timer to trigger data sync notification. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | string | Hospital assigned identifier to the patient to whom this device is implanted |
| triggerTimeStamp | number | Timestamp (in seconds) to indicate the next data sync time – default 1st day of next month |
| freqmode | number | Data sync periodicity (in days) – default 30 |

### Symptom

This class represents symptom details. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | String | Hospital assigned identifier to the patient to whom this device is implanted |
| name | String | Symptom name (chest pain, headache, dizziness, shortness of breath) |
| severity | Number | scale (1-10 (highest)) |
| timeOfOccurence | Number | Timestamp (in seconds) at which patient experienced this symptom |

### Message

This class represents chat message. It consists of following properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| subjectId | string | Hospital assigned identifier to the patient to whom this device is implanted |
| plaintext | string | Text message |
| filelinks | array | File links to image attachments (currently image file types are considered to restrict the type of viewer) |

* This sub-section contains detailed information about entities and the relationship among entities.
* Detailed information about the entities
* attributes
* data type and size
* primary and foreign keys
* expected or estimated data volume on an annual basis
* period for which data is to be retained.
* Detailed information about the relationships between entities
* details of the parameter

## Process Design

### RMSS process

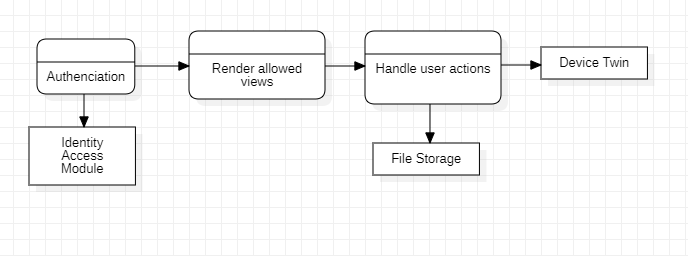


Figure 4 RMSS DFD

RMSS is a web application that comprises of following main processes

Authentication – This process authenticates the user and validates the external Identity Access to ensure right user gets access to the desired functionality

Render allowed views – This process authorizes access to the specified RMSS UI views based on the logged in user type

Handle user actions – This process performs view specific actions. It depends on external File Storage for uploading files. It also depends on Device Twin to manage the transaction data

### RMSS DTAL service

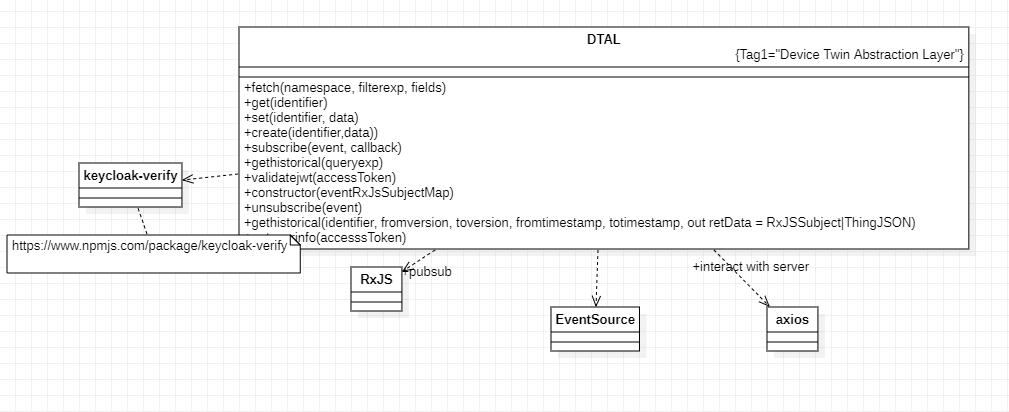


Figure 5 DTAL service class and its dependencies

DTAL service class provides abstraction layer to UI components to access the device twin functions. This class provides following methods

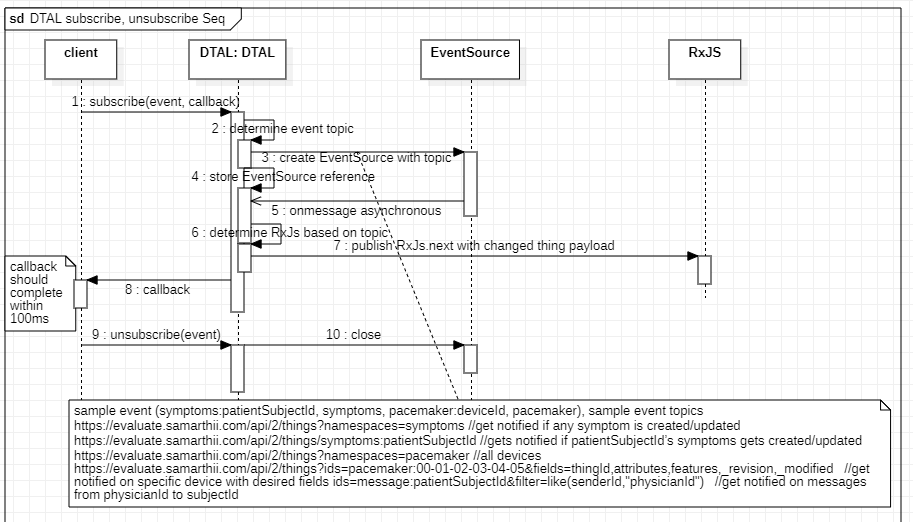
#### Global data

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| eventRxJsSubjectMap | map | maps standard events (e.g. syncTimer, symptom, messages, pacemaker) to RxJS ReplaySubject.  DTAL subscribe() relays the notification event through ReplaySubject. |
| eventTopicMap | map | maps standard events (e.g. syncTimer, symptom, messages, pacemaker) to topic names  DTAL subscribe() uses the topic names in the EventSource to receives stream of events data  eventTopicMap.set("pacemaker", "<https://evaluate.samarthii.com/api/2/things?ids=pacemaker>"); eventTopicMap.set("symptoms", "<https://evaluate.samarthii.com/api/2/things?ids=symptoms>"); |
| eventSourceMap | map | maps standard events (e.g. syncTimer, symptom, messages, pacemaker) to EventSource instances |

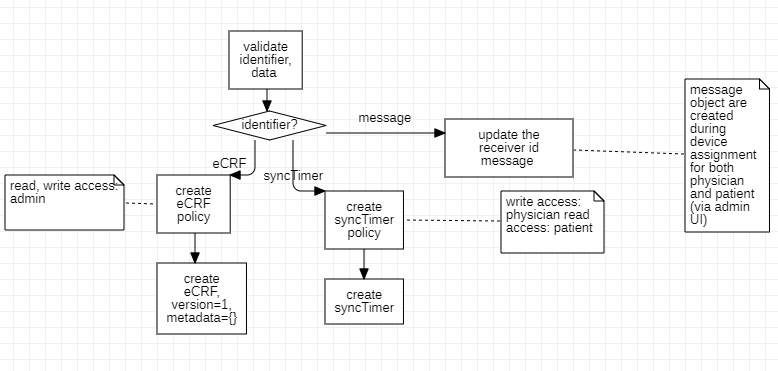
#### constructor()

|  |  |
| --- | --- |
| Prototype | constructor (in: eventSubjectMap) |
| Parameters | Inputs  eventSubjectMap – map of event vs [ReplaySubject](https://rxjs.dev/api/index/class/ReplaySubject) |
| Normal flow | Initialize global instance of eventRxJsSubjectMap |
| Error handling |  |
| Note | The UI component creating this DTAL instance should own the eventSubjectMap and should subscribe to receive events stream. |

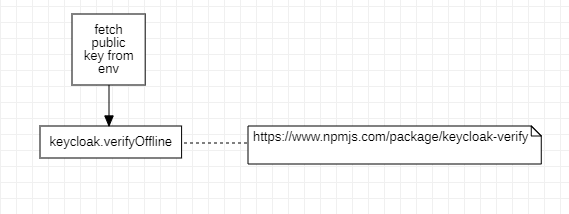
#### subscribe()



#### create()

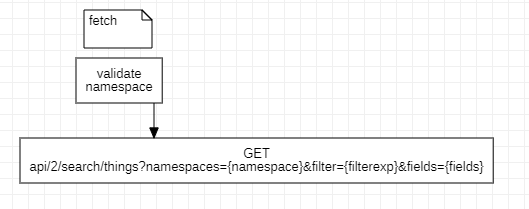


#### validatejwt()

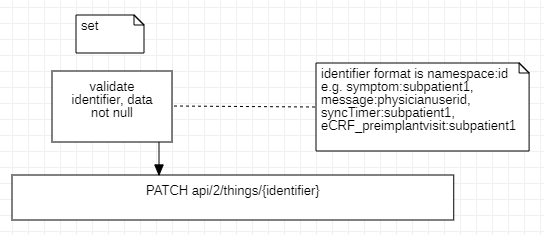


This function provides an offline and out of band method to validate the access token is indeed fetched from the trusted server instance. This token acts as root trust for the subsequent transactions with the server instance.

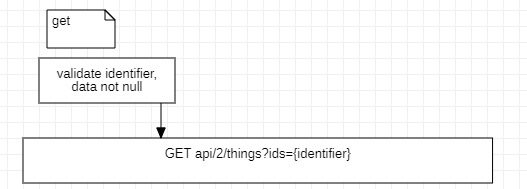
#### fetch()



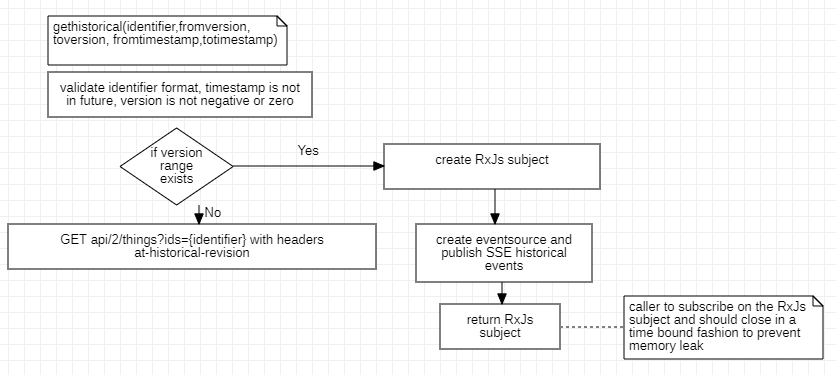
#### set()



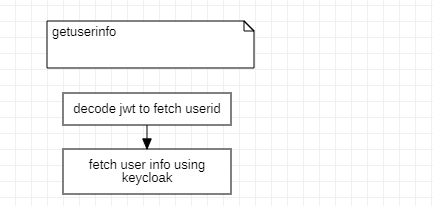
#### get()



#### gethistorical()



#### getuserinfo()



### RMSS app startup

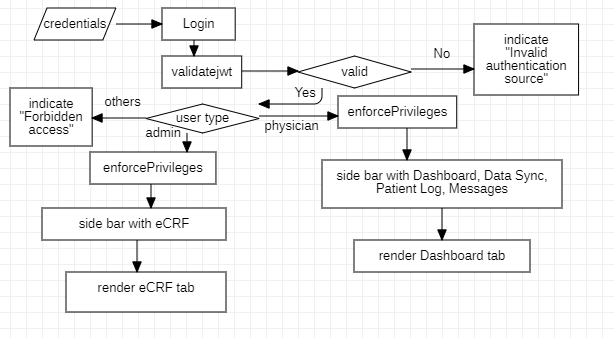


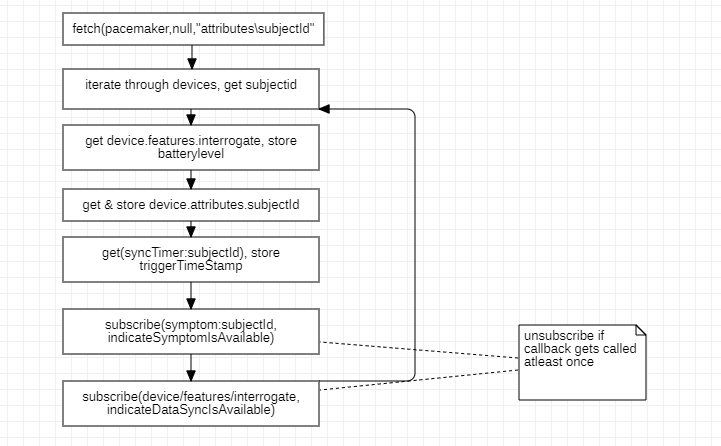
Figure 6 RMSS app flowchart

The above figure depicts the RMSS application startup flow.

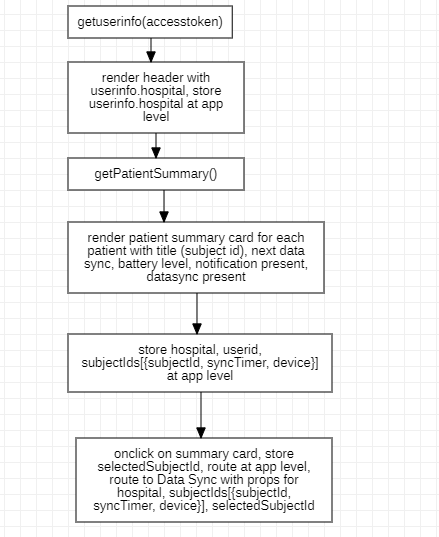
|  |  |
| --- | --- |
| Prototype | enforcePrivileges(in: type) out: componentIds[] |
| Parameters | Inputs  type – indicate user type (admin, clinician)  Outputs  componentIds – array with component identifiers |
| Normal flow | If type is admin, return [eCRF]  If type is clinician, return [Dashboard, DataSync, PatientLog, Message] |
| Error handling | If type is not valid return [] |

### RMSS Dashboard

#### getPatientSummary ()



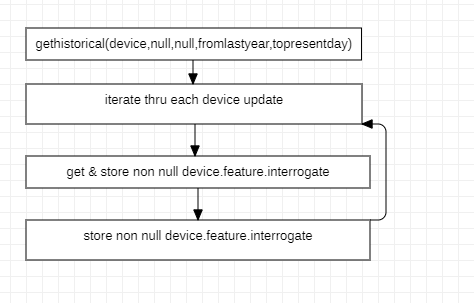
#### render ()



This sub-section provides the detailed DFDs upto the base level

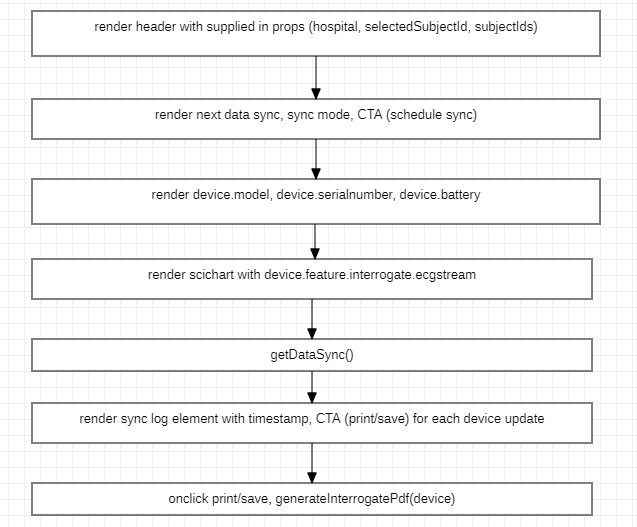
### RMSS Data Sync

#### getDataSync ()



#### generateInterrogatePdf ()

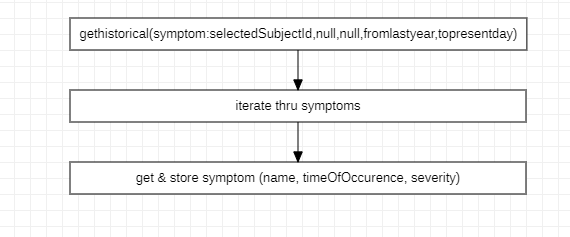
#### render ()



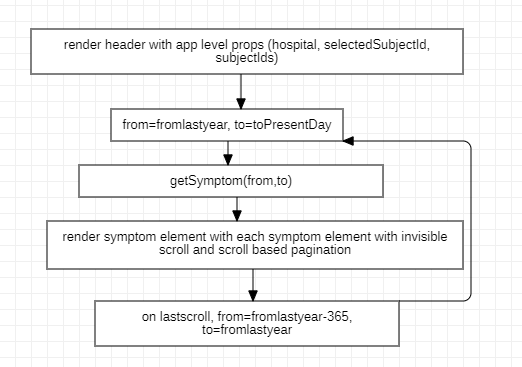
This sub-section provides the detailed DFDs upto the base level

### RMSS Patient Log

#### getSymptom ()



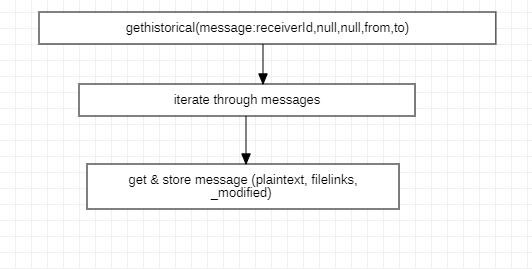
#### render ()



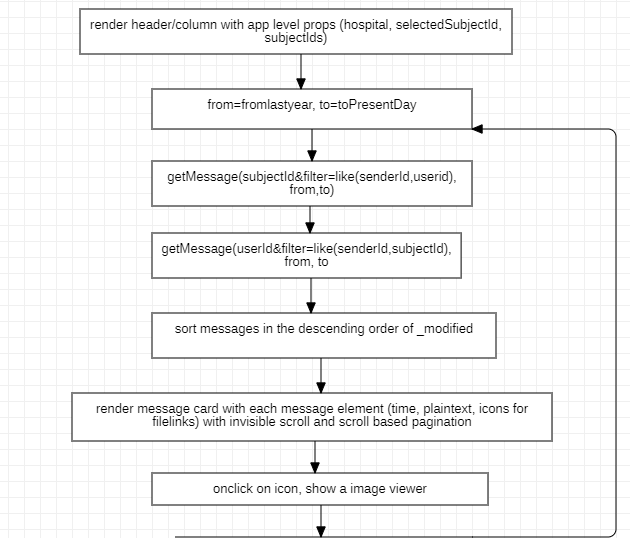
This sub-section provides the detailed DFDs upto the base level

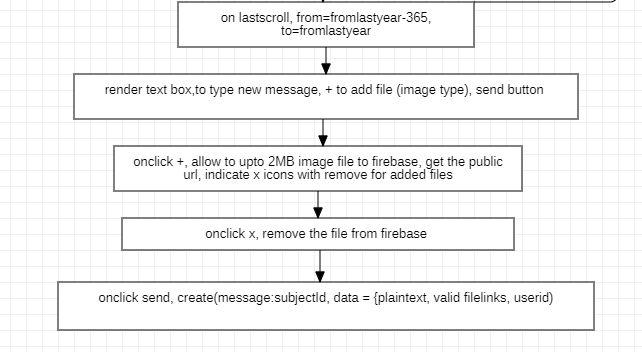
### RMSS Message

#### getMessage ()



#### render ()





This sub-section provides the detailed DFDs upto the base level

### RMSS eCRF

#### render ()

#### render ()

This sub-section provides the detailed DFDs upto the base level

## Interface Design

### Screen Hierarchy and Navigation

This sub-section provides information about hierarchical organization of screens etc.

This sub-section should provide the following information:

* hierarchical organization of the screens (i.e. screens called by screen X and fgscreens calling X)
* a detailed description of the functions performed for each action or option present in the screen (calling a process, screen etc.)

## Interface with External Systems

### Input received from External Systems

For all inputs received from systems interfacing with the target system, this sub-section provides details such as media, format, frequency, communication etc.

### Output given to External Systems

This sub-section identifies tasks or modules interfacing with external systems (defined to be those which are outside the scope of the proposed system).

## Physical Architecture Overview

This sub-section identifies Physical Architecture

## Dynamic Behavior Of Architecture

This section identifies how architecture behave dynamically like

* Turning on the device
* Pairing and streaming
* Handling errors
* Automatic disconnect and reconnect
* Alarms
* Manual disconnect and reconnect

## Justification Of Architecture

This section justify the architecture based on

* System architecture capabilities
* Network architecture capabilities
* Risk Analysis outputs
* Human factors engineering outputs
* Software Environment

## Requirements Traceability

This sub-section provides traceability between requirements and Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Requirement Description** | **Component** | **Comment** |
|  |  |  |  |

## Memory Constraints

Capture all memory Constraints requirements like Start up time, Response time etc.

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  |  |
|  |  |  |

## Error Handling and Fault Management

Describe software driven alarms, warnings and operator messages etc.

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  | RMSS display message “Invalid authentication source” if the server is not trusted |
|  |  | RMSS display message “Forbidden access” if the logged-in user is not admin or physician |
|  |  | RMSS redirects to the login page if validity of access token is expired |
|  |  | RMSS display user friendly contextual message in case of no data found, transaction errors (e.g. lost connectivity) |

## Safety / Hazard Requirement

Risk control measures for potential software defects as appropriate to the medical device and identification of potential hazards that can result from hardware failure and software failure in the system as well as any safely requirements to be implemented in software

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Function** | **Potential Failure Modes & Effects** | | | | **Initial Rating** | | | **Mitigations** | | |
| **Failure Mode** | **Causes of Failure** | **Effects of Failure** | **Safety impact** | **Sev** | **Occ** | **Criticality** | **Mitigation** | **Impact due to mitigations** | **Evidence of Implementation /effectiveness** |
| Push notification | missing notification | token expired | new data updates will be missed | Minor | 2 | 2 | Zone 1 | avoid generic topics, limit the number of active event stream sessions to 8 per laptop | currently none but need to revisit for future requirements demanding more real time and more data | Refer code |
| Buferring of notified events | Buffer full | Receiver UI is not active | intermediate data updates since the buffer full will be missed | Minor | 2 | 2 | Zone 1 |  |  |  |
| File storage | Quota exceeded | Frequent uploads | New files could not be uploaded (eCRF) | Minor | 2 | 3 | Zone 1 | limit uploads to 1 per user every 5 minute, opt for commerical plan e.g. Pay as you go | None | Refer code, firebase configuration |
| File storage | Slow response | Firebase is loaded | File viewer takes time to load and be functional | Minor | 2 | 3 | Zone 1 | opt for dedicated usage, intermediate buffering/cache | cost -> increase in additional monthly charges |  |
| Server | Service not available | Infrastructure outrages | RMSS is not functional | Critical | 4 | 2 | Zone 2 | built infrastructure redundancy | cost |  |

## Security Requirement

Those related to compromise of sensitive information, authentication, authorization, audit trial communication integrity

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  | RMSS functions are accessible only to an user authenticated from a trusted server instance |
|  |  | The authentication token is time bound |
|  |  | The transaction data (device, symptom, messages, synctimer) are guarded by policies controlled by admin user |
|  |  | The communication between RMSS and server is over secured SSL layer (https) and hence the data is encrypted during transit |
|  |  | Server IAM has brute force attack detection capability to disable user account in the event of login failures exceeding a configured threshold (5). |
|  |  | Server IAM has admin console to revoke compromised credentials |
|  |  | Server webserver extends the response payload with a signature to enable the receiver for offline sanity check to detect any tamper in response during transit |

## Non-Functional/Specific Requirement

Those related to Display, Color scheme, User Interface

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  | No default values for eCRF to rule out any software driven errors during data entry |
|  |  |  |

## Boundary Conditions Requirement

Those related to Latency time, pairing time

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  |  |

## Target Environment

Refer to SDP

## Developing Environment

Refer to SDP

## Installation Requirement

Those related to basic Installation requirements such as memory, compatibility, Application availability.

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Item** | **Description** |
|  |  | Version numbers for dependent packages will be fixed in package.json |
|  |  | Ionic CLI |
|  |  |  |