



**COMSATS University Islamabad,  
Park Road, Chak Shahzad, Islamabad Pakistan**

# **SOFTWARE DESIGN DESCRIPTION**

**(SDD DOCUMENT)**

**for**

**Health Monitoring with Theta Middleware**  
Version 1.0

***By***

**Hasnain Khawaja      CIIT/SP17-BSE-004/ISB**

**Noman Nasir Minhas      CIIT/SP17-BSE-010/ISB**

**M. Wahaj Mubeen      CIIT/SP17-BSE-029/ISB**

***Supervisor***

**Dr. Adeel Anjum**

***Bachelor of Science in Computer Science (2017-2021)***

## Table of Contents

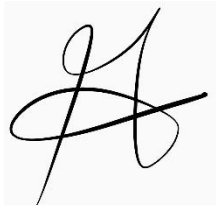
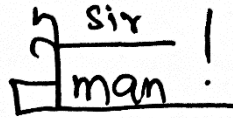
<b>1. Introduction.....</b>	<b>1</b>
<b>2. Design methodology and software process model.....</b>	<b>1</b>
<b>3. System overview.....</b>	<b>1</b>
3.1 Architectural design.....	2
3.2 Process flow/Representation.....	3
<b>4. Design models.....</b>	<b>8</b>
4.1 Data Flow Diagram .....	8
4.1.1 DFD Level 0:.....	8
4.1.2 DFD Level 1:.....	9
4.1.3 DFD Level 2:.....	10
4.2 Individual DFDs.....	11
4.2.1 Register Doctor .....	11
4.2.2 Receive real-time & batched data: .....	12
4.2.3 Add Patient/ Remove Patient: .....	13
4.2.4 View Profiles & Update Login Credentials: .....	14
4.2.5 Visualize Data: .....	15
<b>5. Data design .....</b>	<b>16</b>
5.1 Data Dictionary .....	16
<b>6. Algorithm &amp; Implementation .....</b>	<b>17</b>
6.1 Register Doctor:.....	17
6.2 Receive real-time data: .....	18
6.3 Receive batched data: .....	18
6.4 Add Patient: .....	18
6.5 Discharge Patient:.....	18
6.6 View Profile: .....	19
6.7 Data Visualization: .....	19
<b>7. Software requirements traceability matrix .....</b>	<b>19</b>
<b>8. Human interface design.....</b>	<b>20</b>
8.1.1 Web Interface: .....	20
8.1.2 Mobile Interface: .....	21
8.2 Screen images.....	21
8.2.1 Web Application .....	21
8.2.2 Mobile Application .....	25
8.3 Screen objects and actions.....	26
8.3.1 Doctor:.....	26
8.3.2 Patient:.....	26
<b>9. Appendix I .....</b>	<b>27</b>
<b>10. Demonstration Video Link.....</b>	<b>27</b>
<b>11. Github Links.....</b>	<b>27</b>
11.1 Project Code Link.....	27
11.2 Hasnain Khawaja's Github Profile Link .....	27
11.3 Noman Nasir Minhas's Github Profile Link .....	28
11.4 Muhammad Wahaj Mubeen Github Profile Link.....	28

## **Revision History**

<b>Name</b>	<b>Date</b>	<b>Reason for changes</b>	<b>Version</b>

### Application Evaluation History

Comments (by committee) *include the ones given at scope time both in doc and presentation	Action Taken
<ul style="list-style-type: none"> <li>• Include more scenarios &amp; Use cases which are related to attacks and how your middleware protect/handles them</li> <li>• In Visualizations add graphs or results related to DoS attack/attempts from unauthentic users</li> <li>• In summary, clearly present how the said middleware is effective</li> </ul>	<ul style="list-style-type: none"> <li>• Our middleware is user centric and user does not need to know about details of attacks and how they are being handled.</li> <li>• Our visualization module is for user. For DoS attack attempt we will alert user that request limit has been reached by error message.</li> <li>• Scope Document Updated</li> </ul>


**Supervised by**  
**Dr. Adeel Anjum**

Signature\_\_\_\_\_

# 1. Introduction

The following document contains the design specifications of the sub-systems of our application. Our application consists of two parts. Primary part consists of a middleware, which will allow and assist application developers to connect their application with IoTA Distributed Ledger. Secondary part of our project will consist of a demonstration system of our developed middleware, which in our case will be Health Monitoring System. Our system will use the IOTA ledger in the backend to store the patients' records. IoTA is a secure, lightweight ledger for resource constraint devices. The middleware will collect and transfer the data of the patient's vital signs in real time. We will develop a web interface for the doctor from where they can check the vital signs of their patients, both recorded and in real time. On the other hand, patients can use their mobile interface to view their data as well. Up until now, we have developed the front end of our web and mobile interface. The screens show the options available to both the doctor and the patient. This will guide the users to understand the system and its given functionalities. JavaScript implementation of our middleware is in its final phases while its implementation in Rust has entered its initial development as well. We have setup our MongoDB cluster as well which will serve as our asset manager.

# 2. Design methodology and software process model

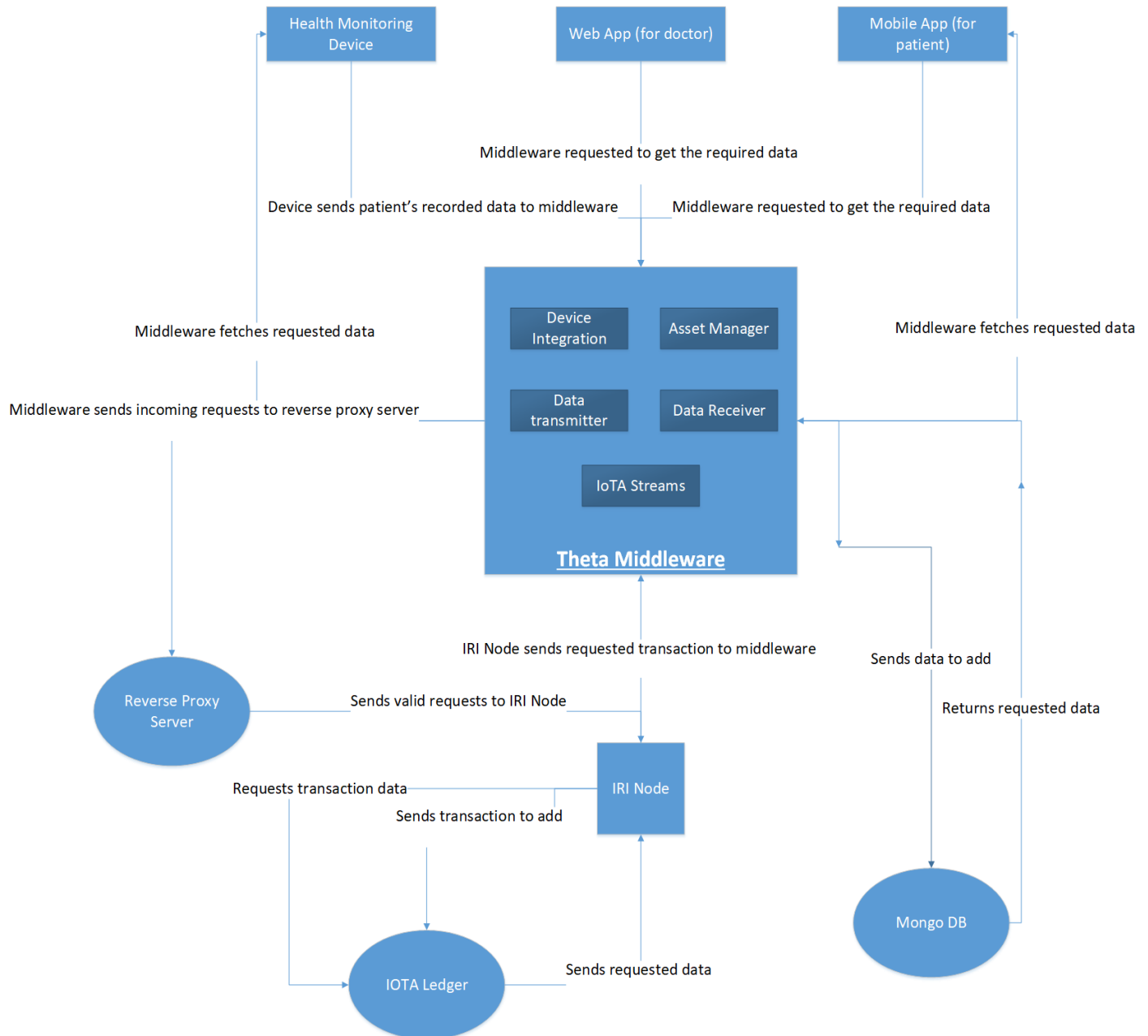
We intend to use the Object-Oriented Approach for development of our proposed system. Motivation behind using this methodology is that our middleware will mainly use IoTA Client Libraries. Middleware will interact with these libraries by using Objects. Moreover, objects will also carry out interactions between user applications and middleware.

We will use the adaptive approach for our process model. The reason behind opting for this approach is that IoTA is being updated constantly and we will direct our development approach according to IoTA updates and upgrades. We will be implementing our middleware within latest updates of IoTA implementation boundaries. In case of any change or upgradation of any existing library, we will have to add changes if required.

# 3. System overview

Our proposed system takes a layered architecture approach. In **first layer** of our system, there will be our Health Monitoring device. There will be web and mobile application to act as an interface for Doctor and Patient respectively. There will be a Data Visualization module, which will provide graphical insights to user. **Second layer** will consist of our middleware, which provides functionalities for interaction with IoTA ledger. Data Transmitter module will collect and send data to IoTA Ledger. Data Receiver module will be responsible for getting requested data from IoTA Ledger and converting it from Trytes format to user-understandable data formats. We will provide integration of devices by using our Integration Module. There will be Asset Manager module, which will keep track of our IoTA Seeds, Addresses and Transactions in a MongoDB cluster. **Third Layer** is concerned with making the system distributed using IRI Node. IRI Node will communicate with IoTA Ledger and store the data into it. IoTA Ledger will be used as main Data Storage Ledger. There will be a Reverse Proxy Server which will be used to avoid DoS attacks.

### 3.1 Architectural design



### 3.2 Process flow/Representation

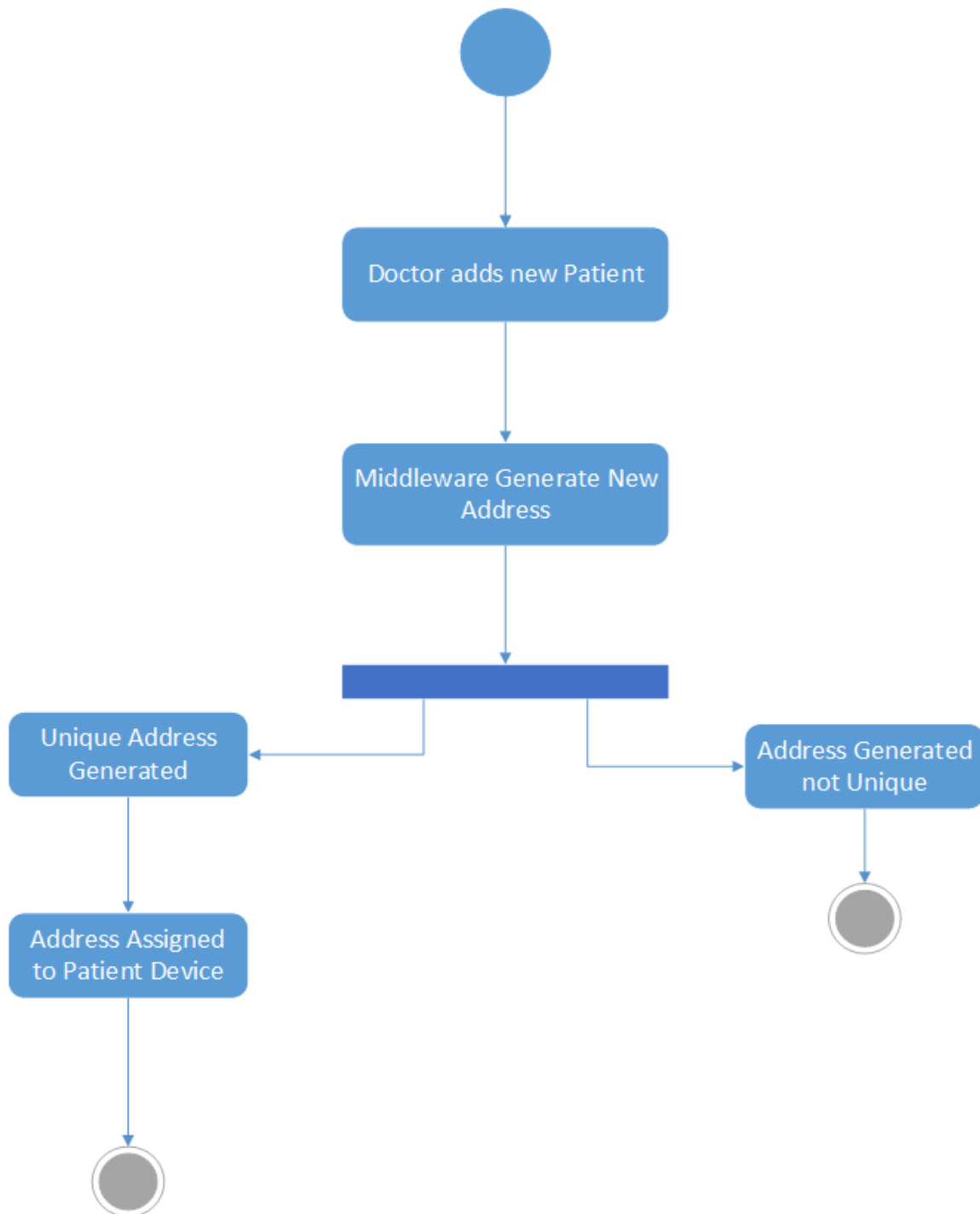
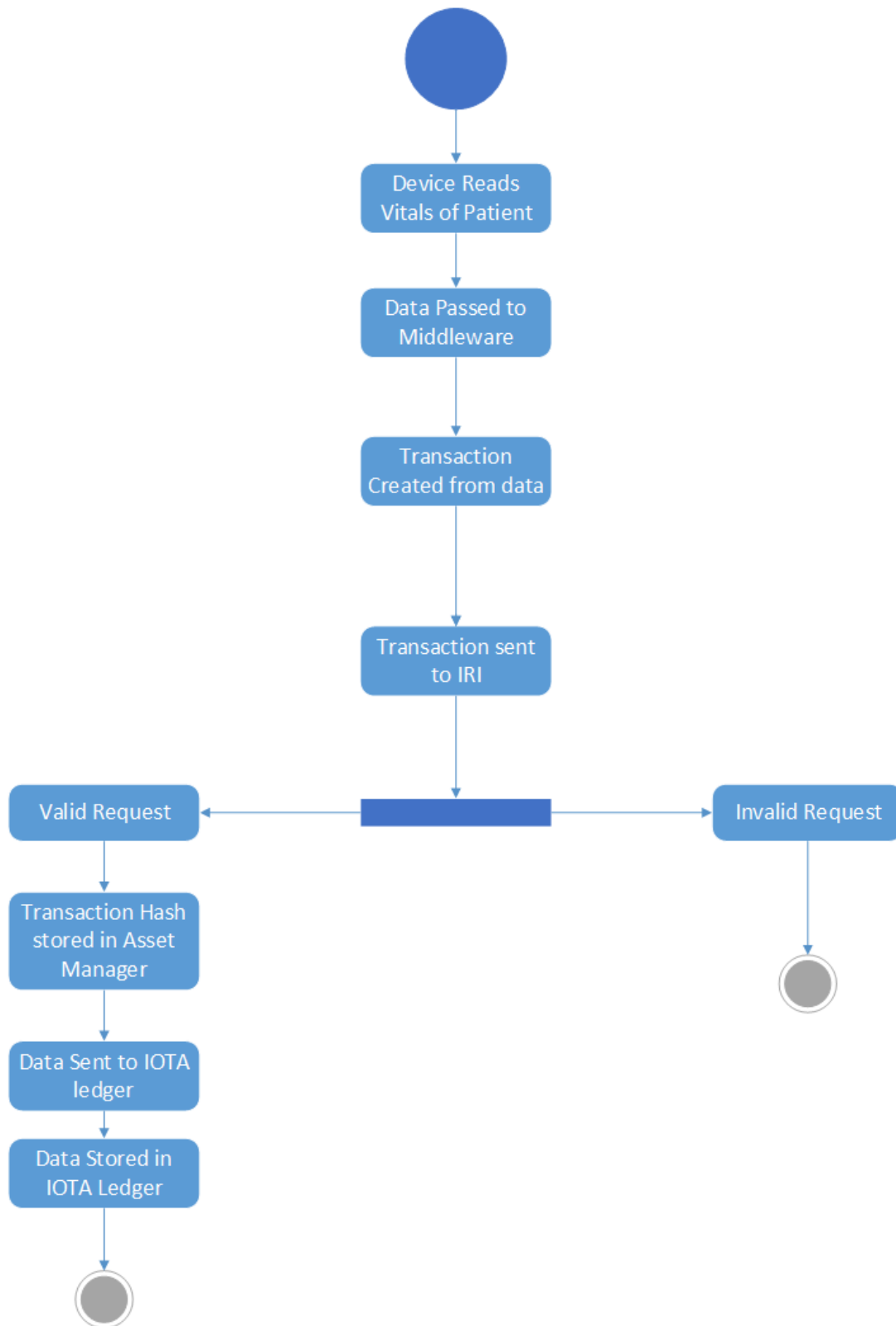
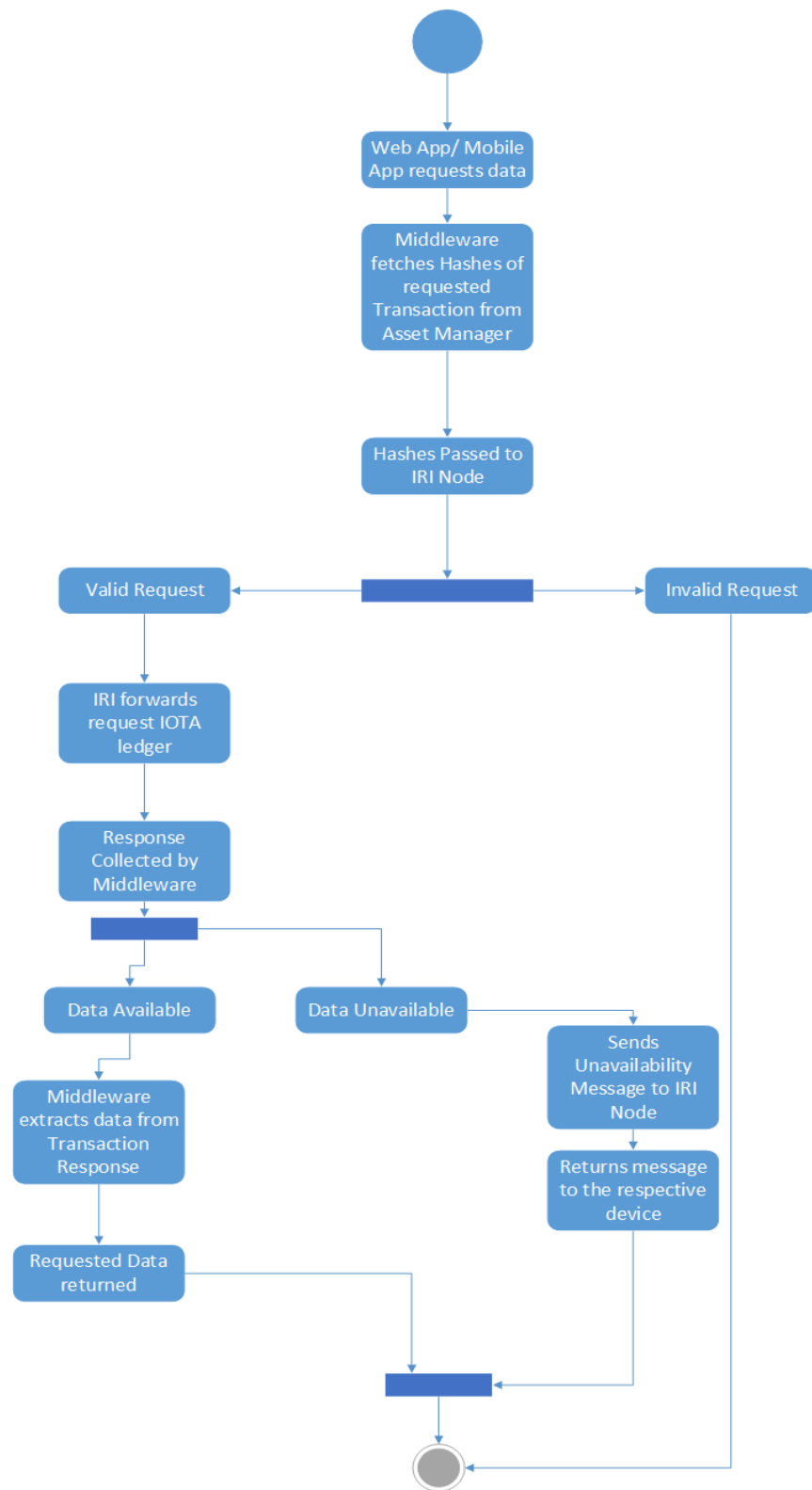


Figure 1- Device Integration

**Figure 2 - Data Transmission**



**Figure 3-Data Receiver**

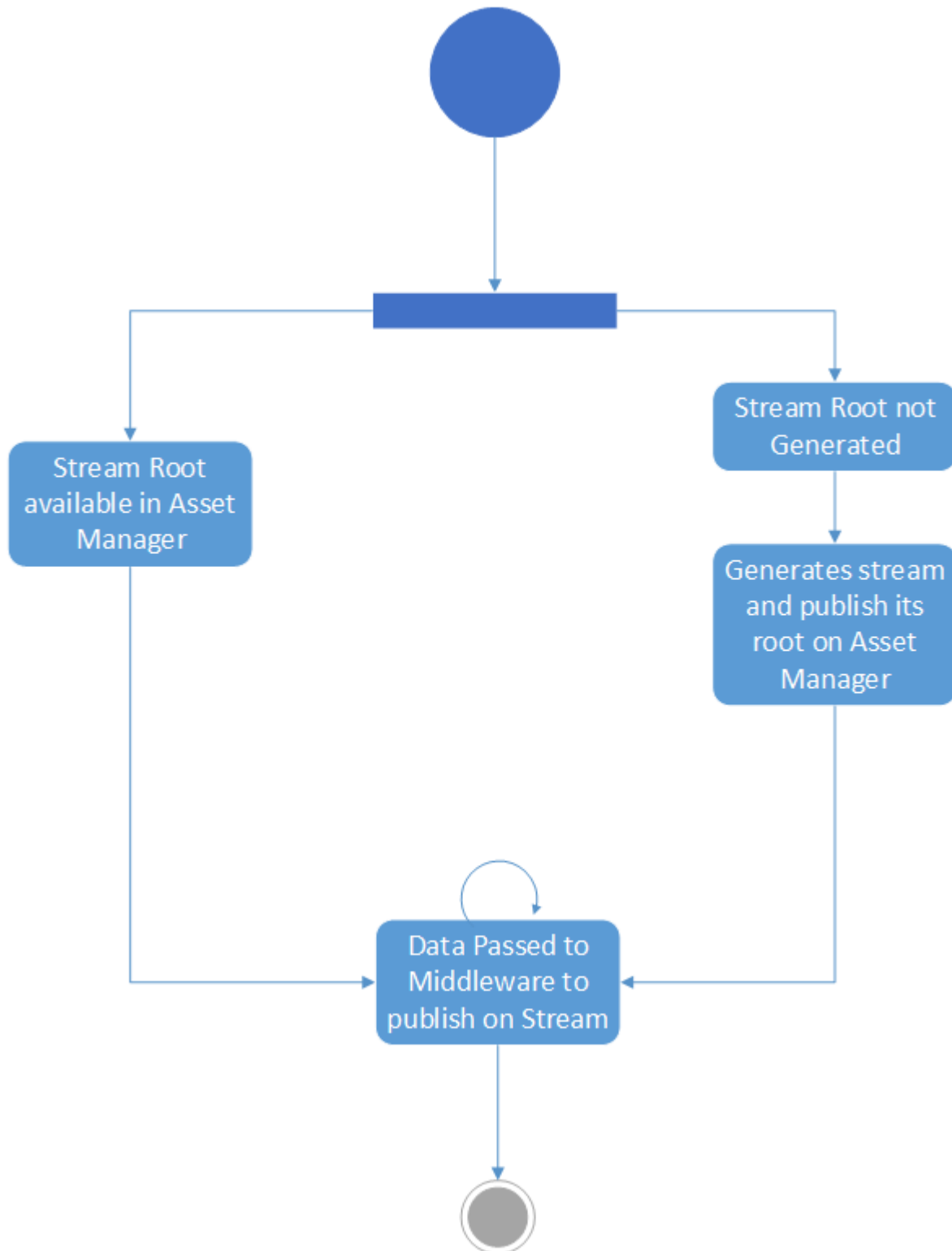


Figure 4 Publish Stream

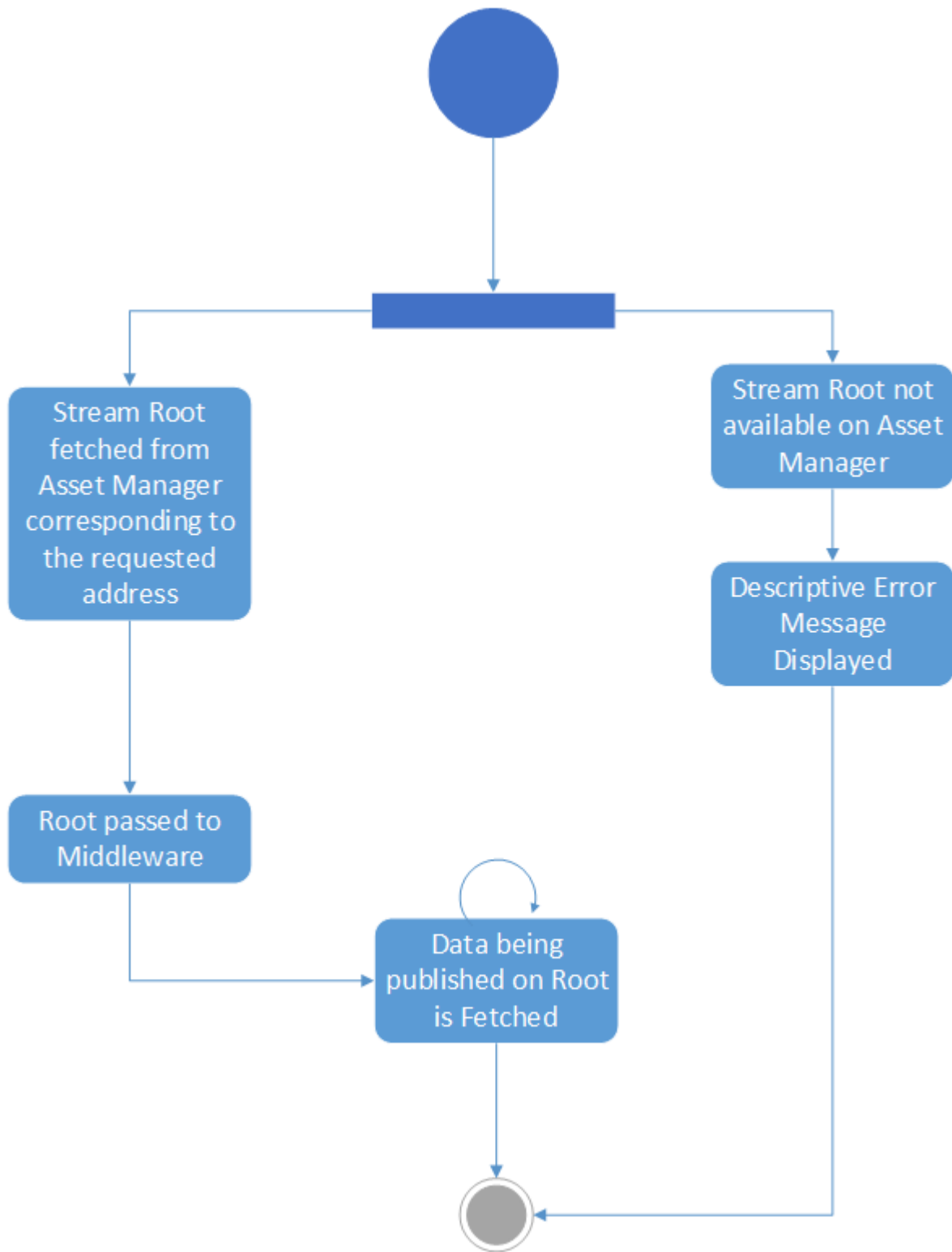
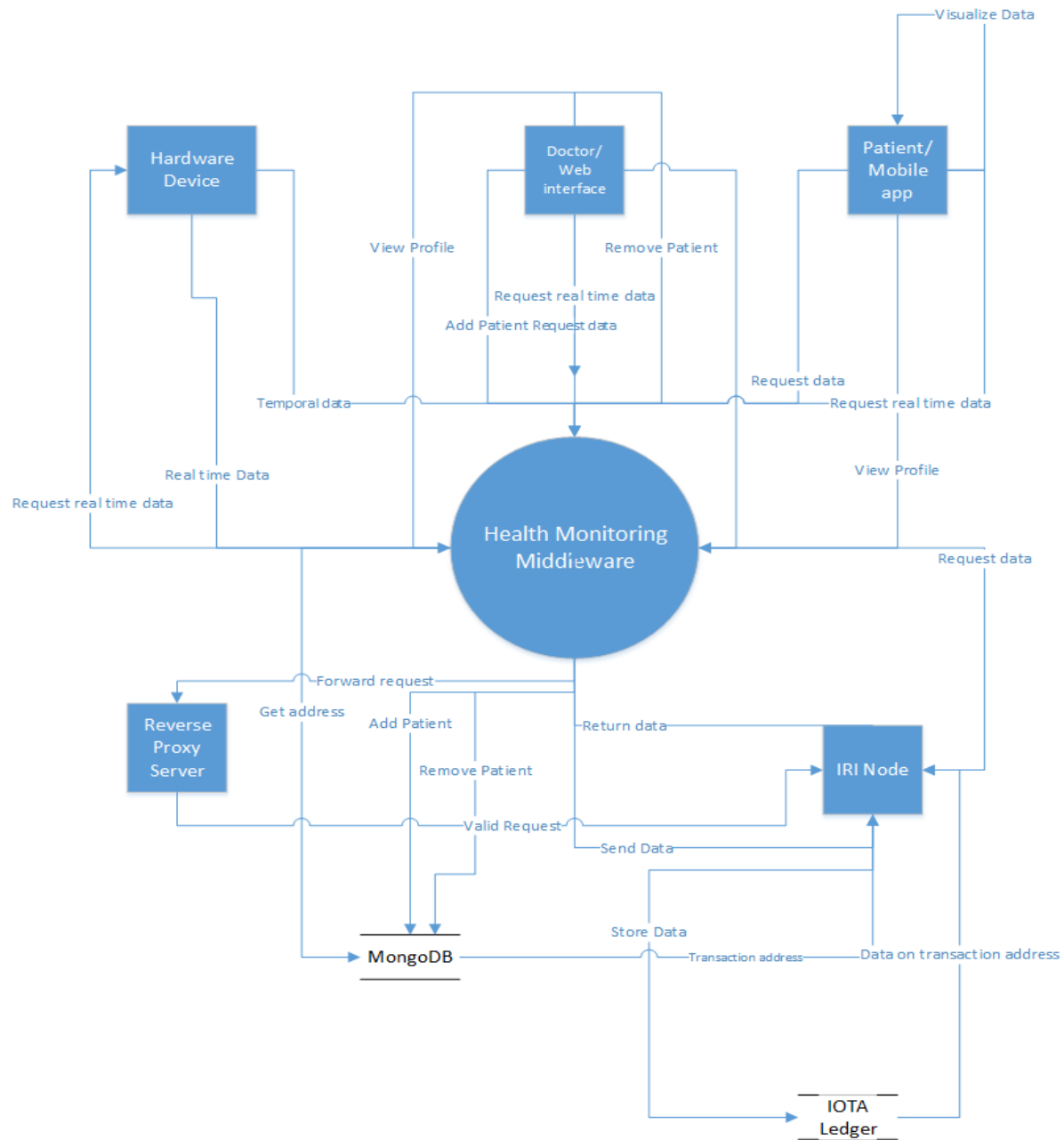


Figure 5-MAM Stream Subscriber

## 4. Design models

### 4.1 Data Flow Diagram

#### 4.1.1 DFD Level 0:

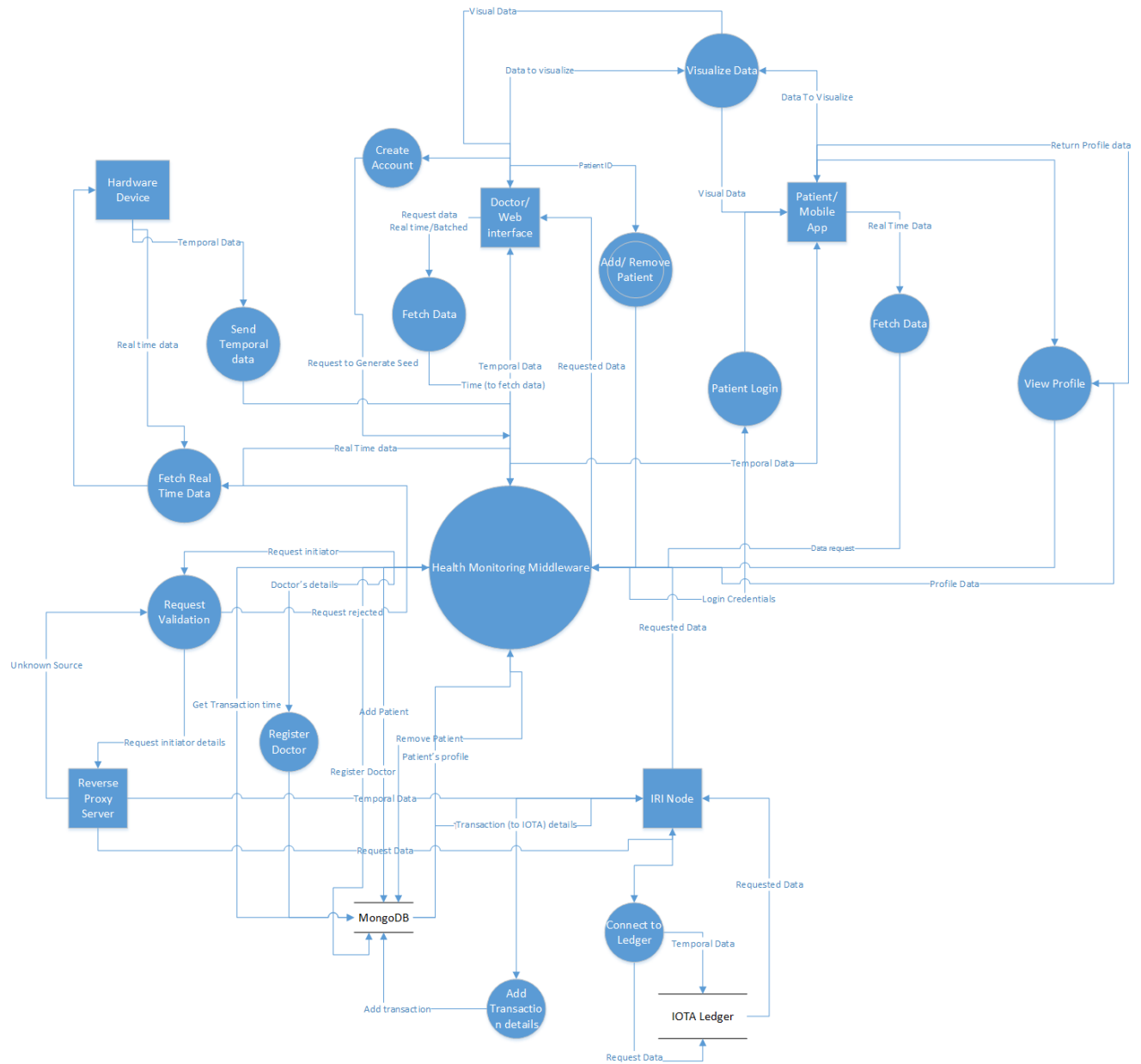


```

graph TD
    HD[Hardware Device] -- "Visual Data" --> VD((Visualize Data))
    HD -- "Temporal Data" --> STD((Send Temporal data))
    HD -- "Real time data" --> FRTD((Fetch Real Time Data))
    DWI[Doctor/ Web interface] -- "Request data Real time/Batched" --> FD((Fetch Data))
    DWI -- "Patient ID" --> VD
    DWI -- "Patient ID" --> ARP((Add/ Remove Patient))
    VD -- "Visual data" --> PMA[Patient/ Mobile App]
    PMA -- "Real Time Data" --> FD2((Fetch Data))
    HDMM((Health Monitoring Middleware)) -- "Request data Real time/Batched" --> STD
    HDMM -- "Temporal Data" --> DWI
    HDMM -- "Requested Data" --> ARP
    HDMM -- "Real Time data" --> FRTD
    HDMM -- "Request initiator" --> RV((Request Validation))
    RV -- "Get Transaction time" --> RPS[Reverse Proxy Server]
    HDMM -- "Register Doctor" --> RPS
    HDMM -- "Add Patient" --> RPS
    HDMM -- "Remove Patient" --> RPS
    RPS -- "Temporal Data" --> IN[IRI Node]
    RPS -- "Transaction address" --> IN
    RPS -- "Request Data" --> IN
    IN -- "Requested Data" --> RV
    IN -- "Temporal Data" --> IL[IOTA Ledger]
    IL -- "Request Data" --> IN
    IL -- "Requested Data" --> IN

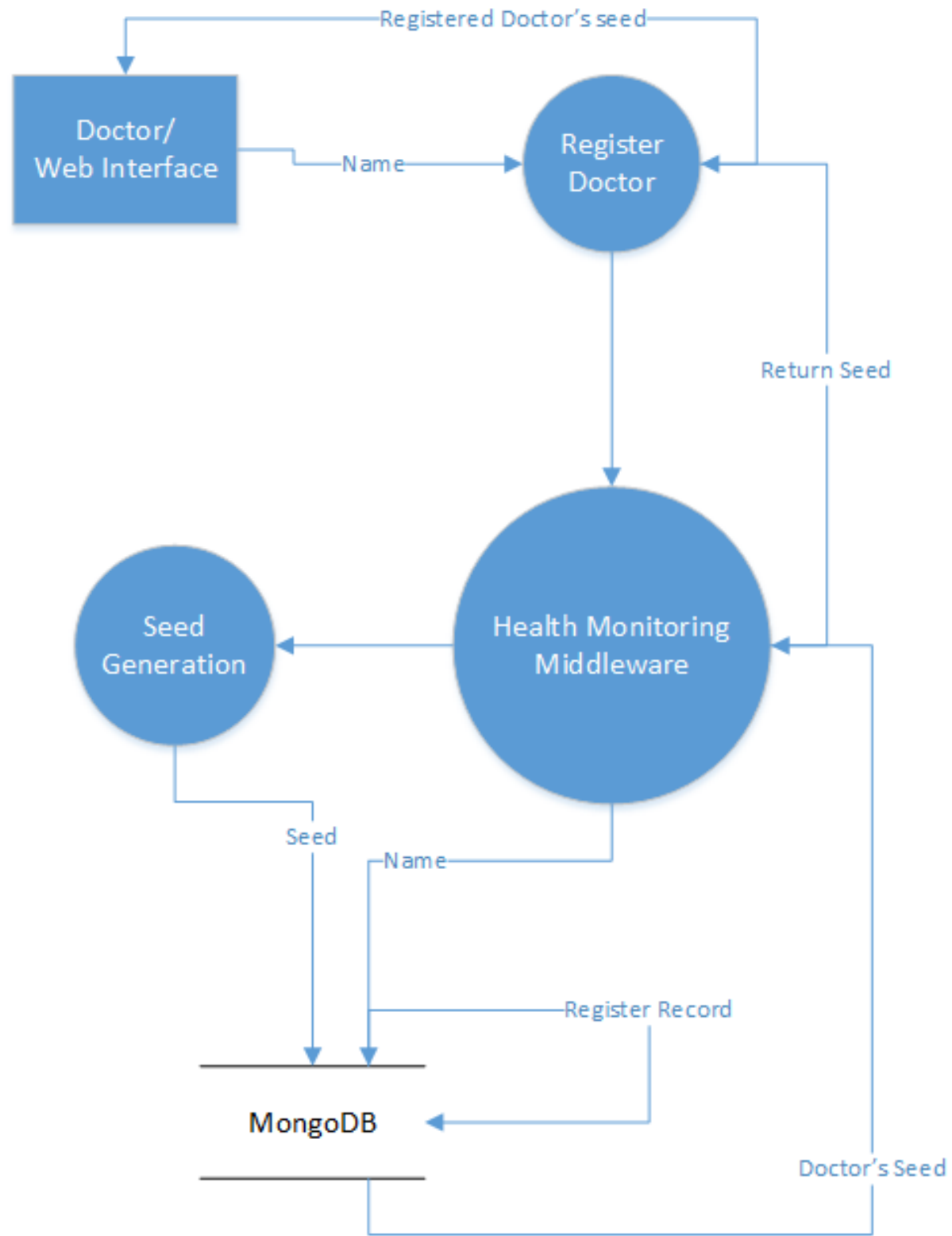
```

## 4.1.3 DFD Level 2:

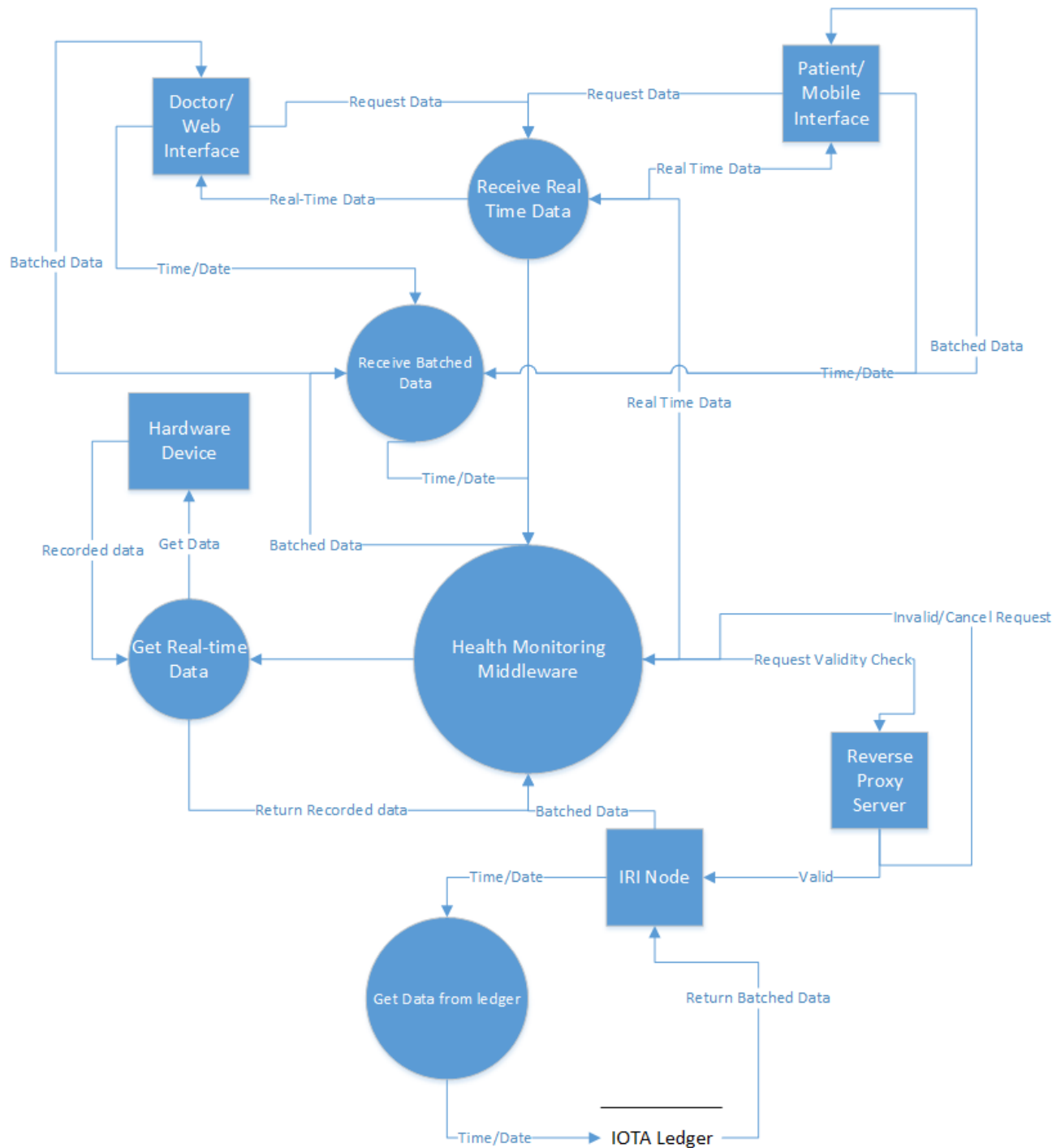


## 4.2 Individual DFDs

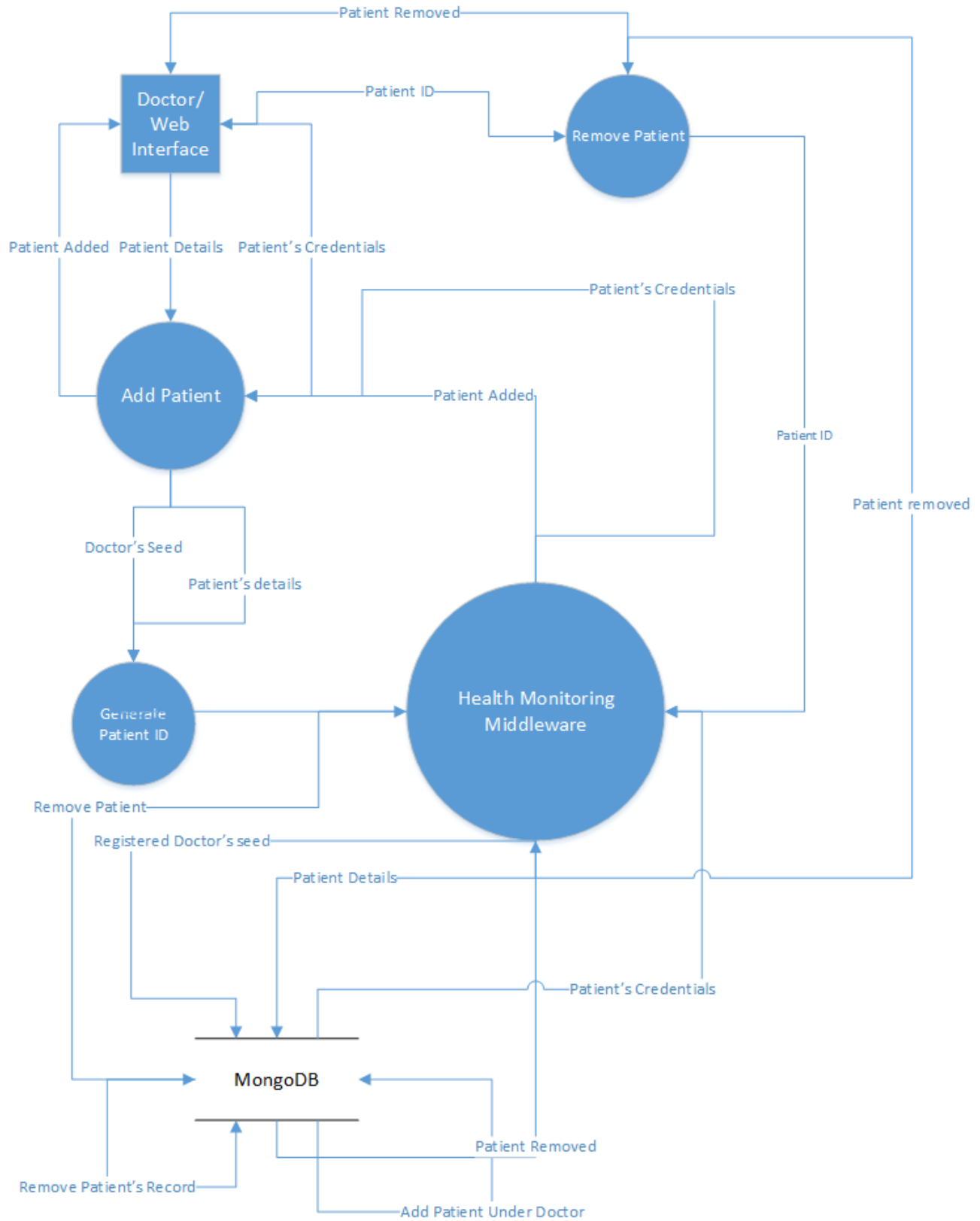
### 4.2.1 Register Doctor



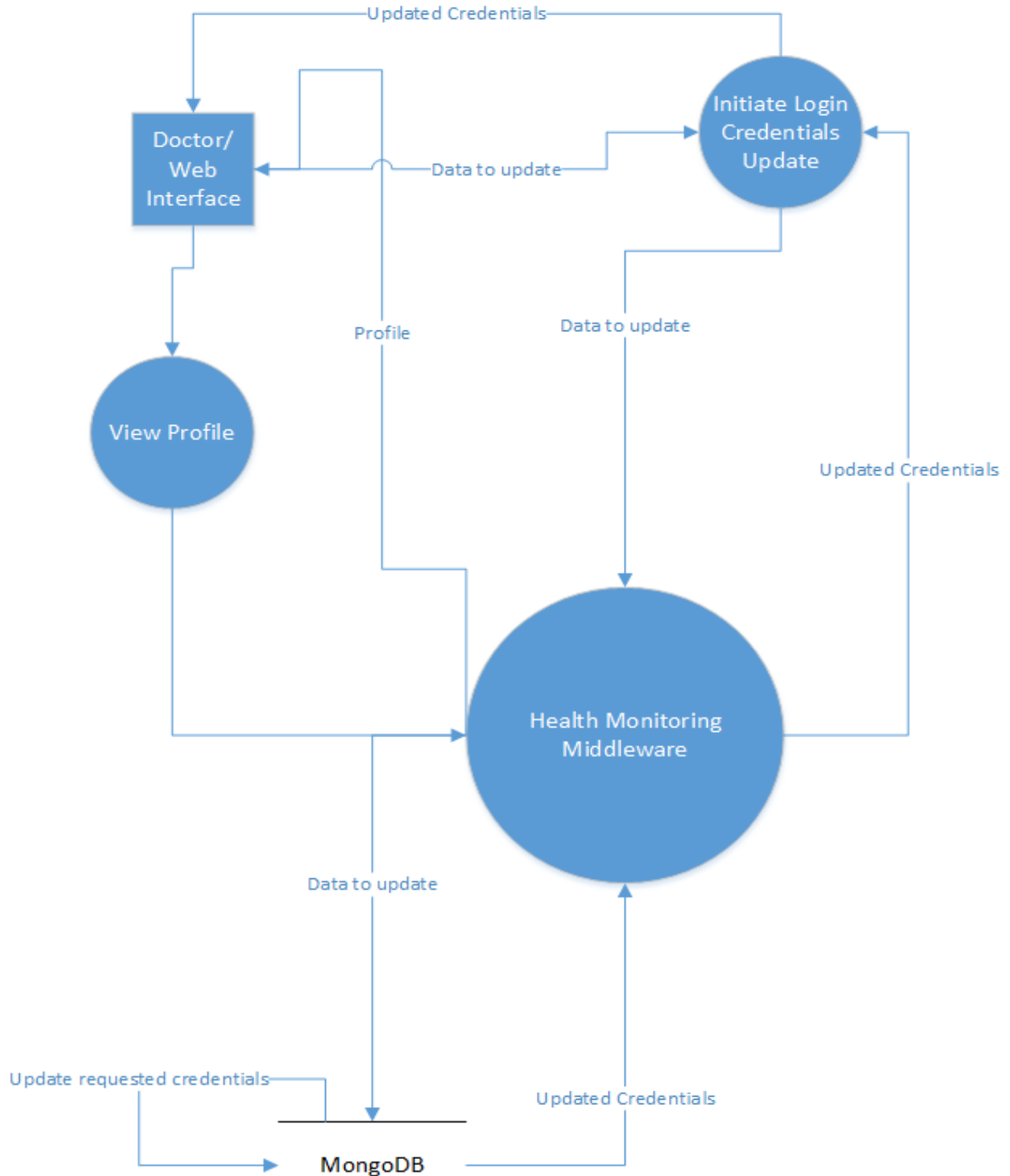
## 4.2.2 Receive real-time &amp; batched data:



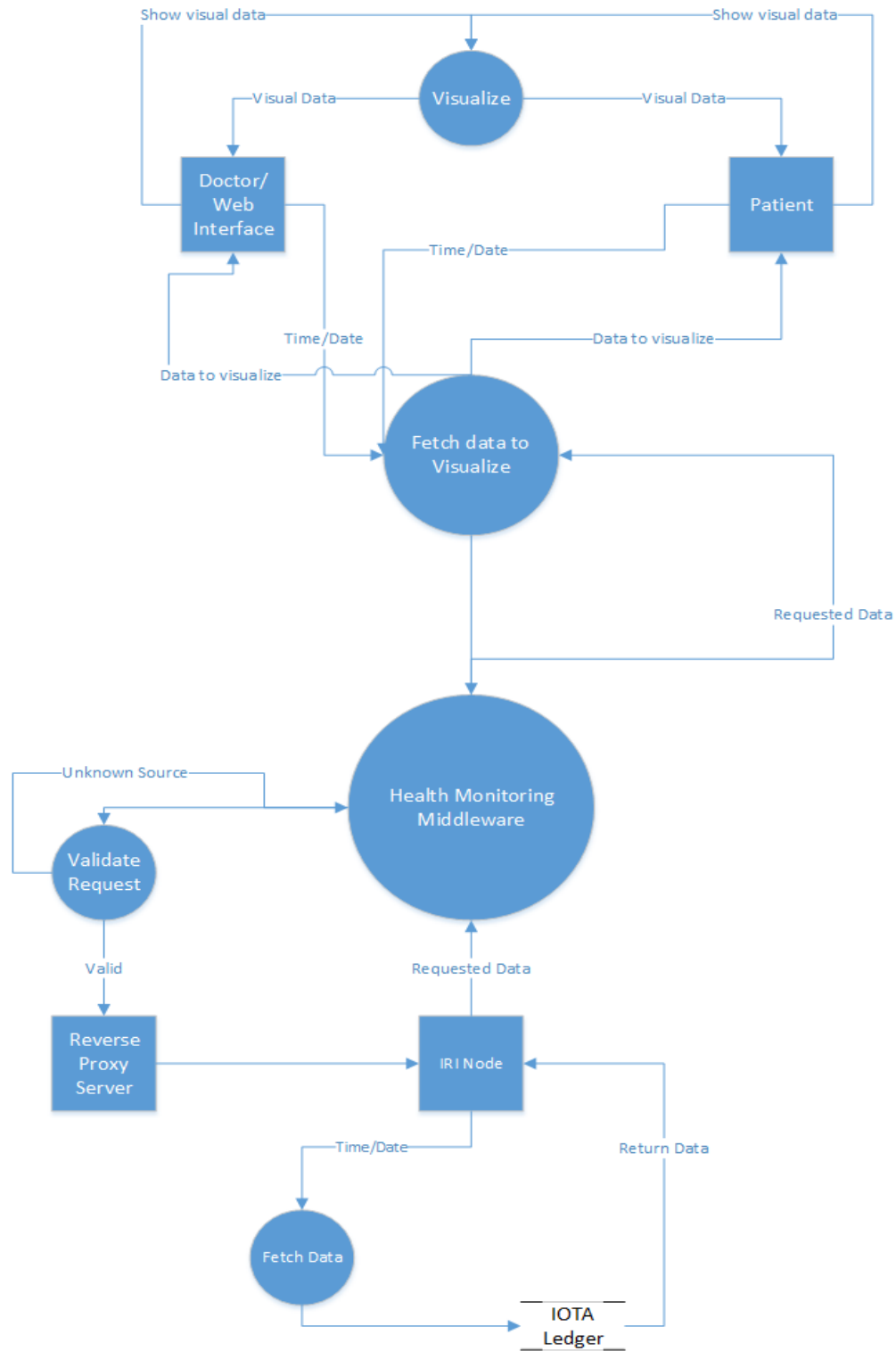


**4.2.3 Add Patient/ Remove Patient:**

#### 4.2.4 View Profiles & Update Login Credentials:



#### 4.2.5 Visualize Data:



## 5. Data design

In our system, we will use IoTA Ledger and MongoDB as our storage entities. Almost all of our data will be generated and stored on run-time, but a generic design scheme of our data will be as follows.

```
var newSeed = {
  _id: id,
  ID: id,
  PASSWORD: password,
  Profile: info*,
  SEED: seed,
  streamRoot: null
};

var newAddress = {
  _id: id,
  ID: id,
  PASSWORD: pass,
  SEED: seed,
  Profile: info*,
  ADDRESS: finalAddress,
};

var newTransaction = {
  _id: hash,
  date: getDate(),
  timestamp: timestamp(),
  ADDRESS: address,
  txHash: hash,
};
```

\***info** above schema can be any JSON object containing data in any format.

All new Seeds will be stored in a single MongoDB Collection “SEEDS”. However, every Address will be stored in a separate collection named after its corresponding Seed. Similarly, every transaction will be stored in a collection named after its corresponding Address. Corresponding Seed and Address collections will be auto generated by Asset Manager Module.

### 5.1 Data Dictionary

#### Middleware:

```
+ addAddress(String, String, String, JSON_Object, String): void
+ addSeed(String, String, String, String): void
+ addTransaction(String, String): void
+ fetchPrivateMAM(String, String): String
+ fetchPublicMAM(String): String
+ generateAddressLocally(String, int, int, String, String, JSON_Object): String
+ generateSeed(String, String, JSON_Object): String
+ getAllHash(String, Date): String[]
+ getDate(): String
+ getNode(): String
+ getSeed(String, String): String
```

```

+ getSingleData(String, Time): String
+ getSingleHash(String, Time): String
+ privateMAM(String, String): void
+ publicMAM(String, String): void
+ sendPrivateTransaction(String, String, String): void
+ sendPublicTransaction(String, String, String): void
+ timestamp(): String

```

**Doctor:**

```

+ AddPatient(String, String, int, String, String, String, String): boolean
+ SignUp(String, String, int, String, String, String, String): boolean
+ ForgotPassword(String): String[]
+ PatientsList(): String[]
+ LiveReadings(String): String[]
+ Login(String, String): Boolean
+ ViewPatientProfile(String): String[]
+ Profile(String): String[]
+ ViewHistory(String): <String[]>

```

**Patient:**

```

+ Profile(String): String[]
+ LiveReadings(String): String[]
+ qrScanner(String): Boolean
+ ViewHistory(String): <String[]>

```

## 6. Algorithm & Implementation

### 6.1 Register Doctor:

Go to login page

Register button click event

    Doctor\_name = user input

    Specialization = user input

Take other profile details (optional)

Create seed button click event

    Generate seed using seed\_generate() method

    Send seed, profile details and current\_time to the database

    Insert the sent data into database and register the doctor

Take doctor the landing page of the app

## 6.2 Receive real-time data:

Click Real-Time button

- Send data request to hardware device

- Wait for the hardware device to send data

- Receive the real-time readings

- Change the format of the data (if necessary)

- Display the readings on the screen

## 6.3 Receive batched data:

Click historic data button

Prompt user to insert the time and date

- Time = user input

- Date = user input

Send time and date to IRI node

Fetch the transaction address against the user provided date and time (if any)

Insert the transaction address to IOTA explorer

Fetch and return the transaction data back to doctor's app

Receive the data and clean it (if required) on doctor's app

Display the fetched data

## 6.4 Add Patient:

Click Add Patient button

Prompt input fields for the patient details

If no field is left empty

- Allow user to click 'Add' button

Add button event

Generate patient id using the doctor's seed and patient details

Send the doctor's seed, patient id and patient's details to database

Store the patient's details under the provided doctor's seed in database

Update the 'patients' section in web app with the new patient added

Show success prompt

## 6.5 Discharge Patient:

Go to the Patient's details in web app

Select the patient to discharge

Click discharge patient button under patient's profile

Prompt user to confirm

If yes, then send the query to delete the patient record for the given doctor id and patient id

Search the doctor and then patient for the given id in database

Delete the patient record from the database

Fetch the updated patients' details against the doctor's seed  
 Update the patient's details in doctor's web app  
 Prompt Patient removed  
 Else take user back to patients' details page

## 6.6 View Profile:

Click view profile button  
 Get the seed or id for the given account to database  
 Send query to get profile data for the available seed or id to database  
 Receive the result from the database  
 Display the information on the web interface

## 6.7 Data Visualization:

Click the Visualization button  
 Prompt user to input date  
 Fetch required data through middleware  
 Once processed, display the visual data on screen

# 7. Software requirements traceability matrix

**Table1- Requirements Traceability Matrix**

Req. Number	Ref. Item	Design Component	Component Items
FR01	DFD	Level 0,1,2	Send Temporal data
FR02	DFD	Level 2, 2.2	Receive data
FR03	DFD	Level 2, 2.2	Receive Encrypted Data
FR04	DFD	Level 2, 2.1	Create Seed
FR05	DFD	Level 2, 2.1	Create IOTA addresses
FR06	DFD	Level 2, 2.1	Track addresses
FR07	DFD	Level 2, 2.3	Add Patients
FR08	DFD	Level 2.2	Real time reading

FR09	DFD	Level 2.2	Discharge patient
FR10	DFD	Level 2.4	View Patient's profile
FR11	DFD	Level 2.4	Update login Credentials
FR12	DFD	Level 2, 2.2	Read Previous data
FR13	DFD	Level 2, 2.5	Visualize data
FR14	DFD	Level 2.1	Create/Register account
FR15	DFD	Level 2.6	Notification Alert
FR16	DFD	Level 2, 2.2	View Readings (Patient)
FR17	DFD	Level 2.2	View real time and batched data (Patient)
FR18	DFD	Level 2.2	Read Historic Data
FR19	DFD	Level 2.5	Visualize (patient)

## 8. Human interface design

There are two human interfaces for our project, one for doctor and other for patient. Both interfaces have descriptive and well-defined instructions what each component on the screen performs.

### 8.1.1 Web Interface:

Doctors can interact with the system using the web application. A doctor can login to the system using his token that was generated during sign up. Once logged in, system will show the doctor's profile including all his patients as well. By viewing a certain patient, doctor can view the patient's profile showing their vital signs in real time. Patient's profile will have options to view the recorded data and to visualize the data for the given time. Furthermore, doctor can add or remove patients from the system.

This web app will be connected to the IRI Node at the backend. During all these processes, web app will communicate with IRI node (sending requests) which will further request IoTA ledger if required. Web app will keep receiving the real time data of the patients as soon as the hardware device sends it for IoTA ledger.



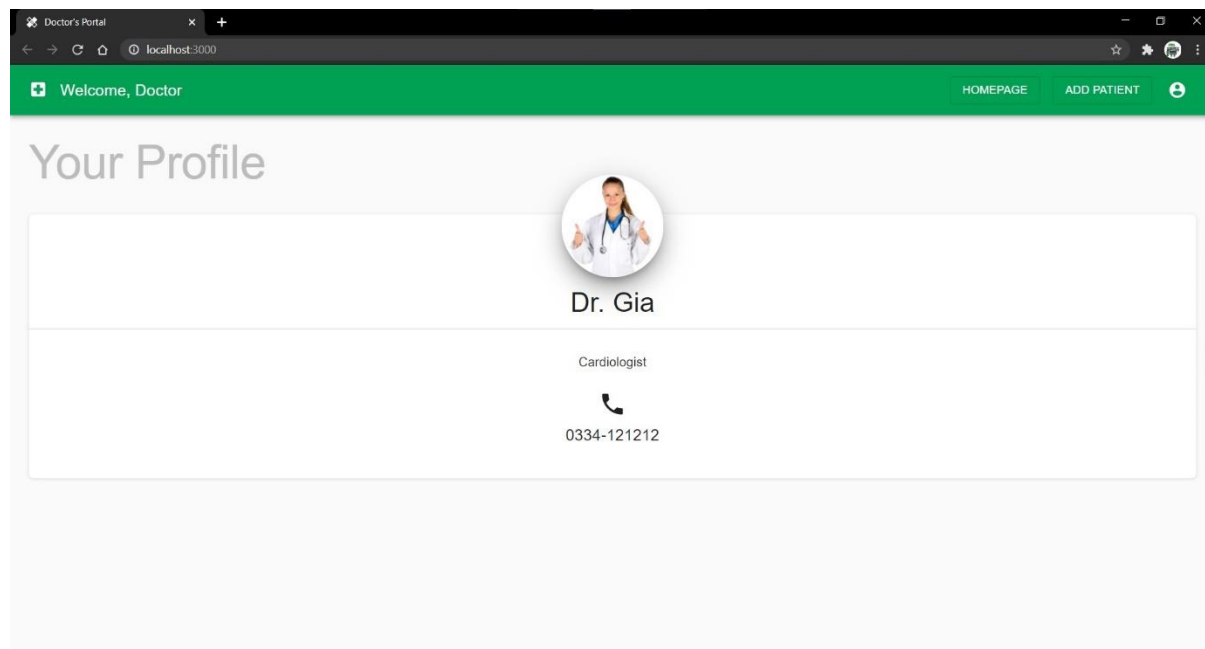
### 8.1.2 Mobile Interface:

Patients will have their own user interface on mobile phone. The mobile app will let patients sign in once they are added into the system by a doctor. Patient's functions are limited as compared to the doctors. A patient cannot add or remove himself from the system while a doctor can. So, once logged in, system will take the patient to their profile. Patient can also see their vital signs in real time. Same screen will also show the options of viewing the historic/batched data and visual data in case patient is interested.

Like the web interface, mobile app is also connected to IRI node using middleware. For any request, mobile app will have to request IRI node which will further request the IoTA ledger for the desired data. This mobile app will also receive the real time data as soon as hardware device sends it to the network.

## 8.2 Screen images

### 8.2.1 Web Application



The screenshot shows a web browser window titled "Doctor's Portal" with the address bar displaying "localhost:3000". The page has a green header bar with the text "Welcome, Doctor" on the left and "HOMEPAGE" and "ADD PATIENT" on the right. The main content area is titled "Patient's Profile" and displays the following information:

- Patient's Name: Wahaj Mustakeem
- Patient's Age: 69
- Vitals section containing:
  - Temperature: 100 F
  - BPM: 85
  - BP (mm/Hg): 120/80

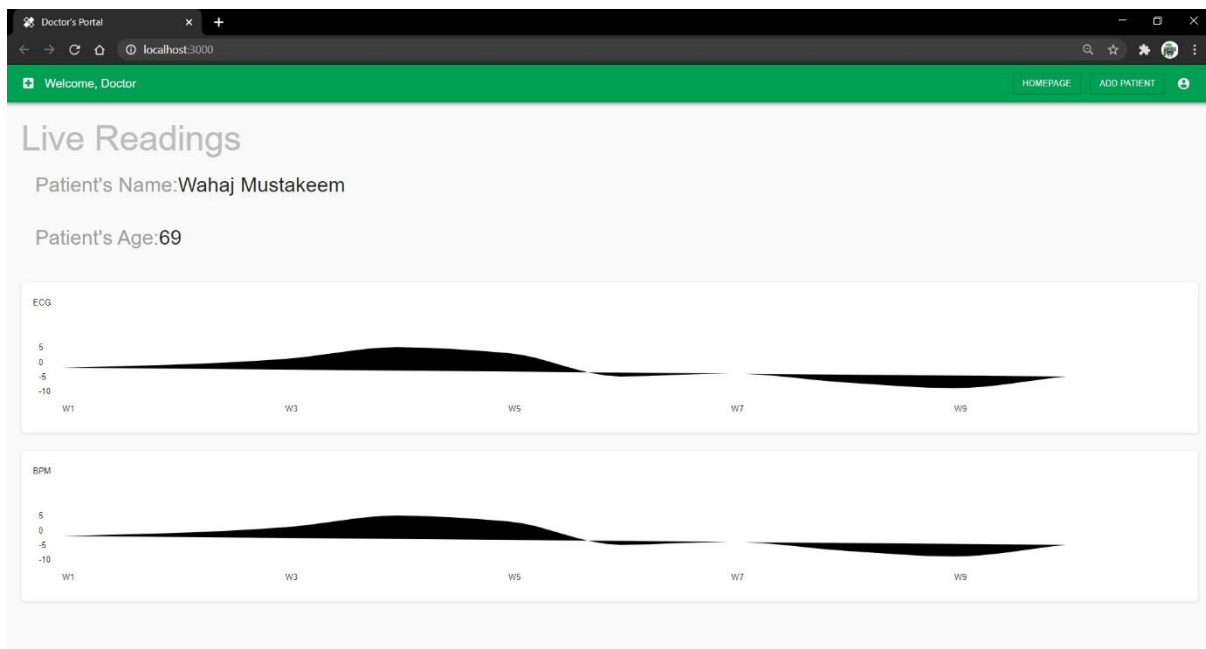
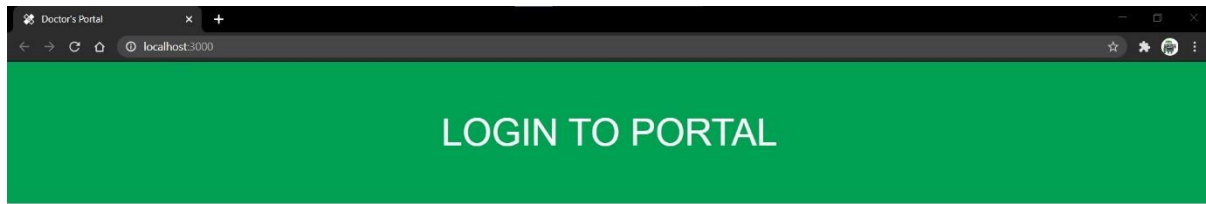
On the right side of the profile, there are four buttons stacked vertically:

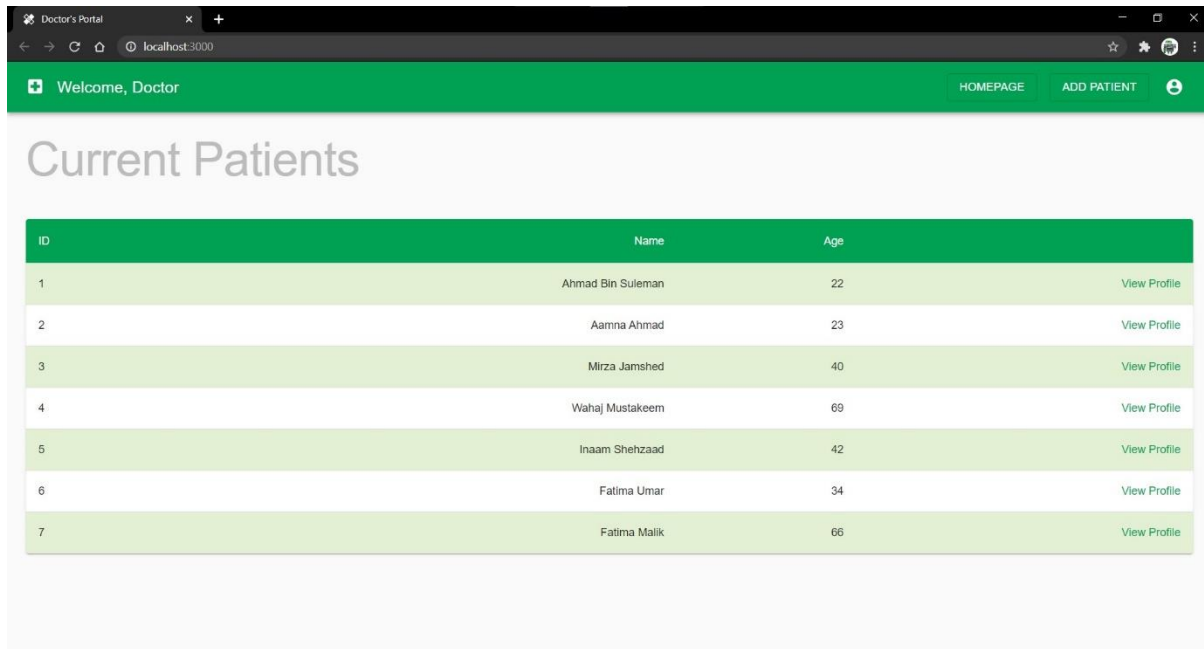
- GENERATE REPORT
- DISCHARGE
- VIEW HISTORY
- LIVE STATISTICS (highlighted in green)

The screenshot shows a web browser window titled "Doctor's Portal" with the address bar displaying "localhost:3000". The page has a large green header bar with the text "SIGN UP" in white. Below the header, there is a sign-up form with the following fields:

- First Name \*
- Last Name \*
- Email Address \*
- Password \*

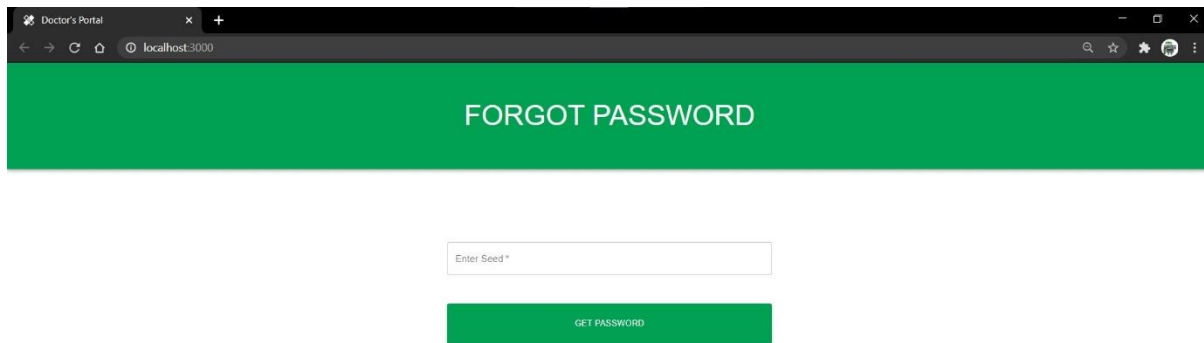
Below the form is a green button labeled "SIGN UP". At the bottom of the page, there is a link that says "Already have an account? Sign In".





The screenshot shows a web browser window titled "Doctor's Portal" with the address bar displaying "localhost:3000". The page has a green header bar with the text "Welcome, Doctor" on the left and two buttons, "HOMEPAGE" and "ADD PATIENT", on the right. Below the header, the main content area has a light gray background with the title "Current Patients" in a large, bold font. Underneath the title is a table with three columns: "ID", "Name", and "Age". The table contains seven rows of patient data, each with a "View Profile" link to its right.

ID	Name	Age	
1	Ahmad Bin Suleman	22	<a href="#">View Profile</a>
2	Aamna Ahmad	23	<a href="#">View Profile</a>
3	Mirza Jamshed	40	<a href="#">View Profile</a>
4	Wahaj Mustakeem	69	<a href="#">View Profile</a>
5	Inaam Shehzaad	42	<a href="#">View Profile</a>
6	Fatima Umar	34	<a href="#">View Profile</a>
7	Fatima Malik	66	<a href="#">View Profile</a>



The screenshot shows a web browser window titled "Doctor's Portal" with the address bar displaying "localhost:3000". The page has a solid green background with the text "FORGOT PASSWORD" in white, bold, uppercase letters. Below this, there is a white input field with the placeholder text "Enter Seed \*". Underneath the input field is a green button with the text "GET PASSWORD" in white, uppercase letters.

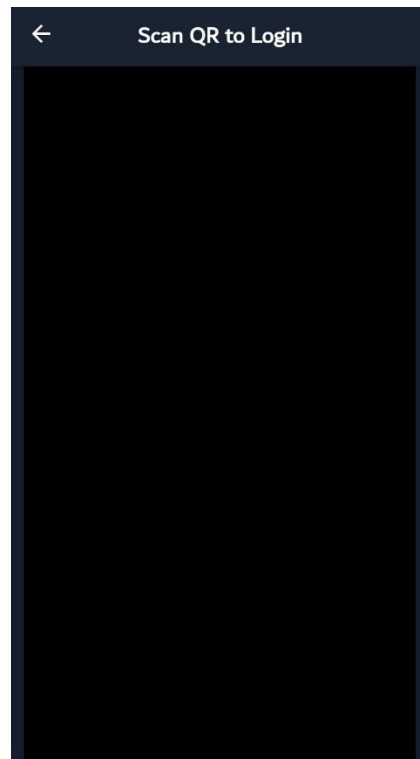
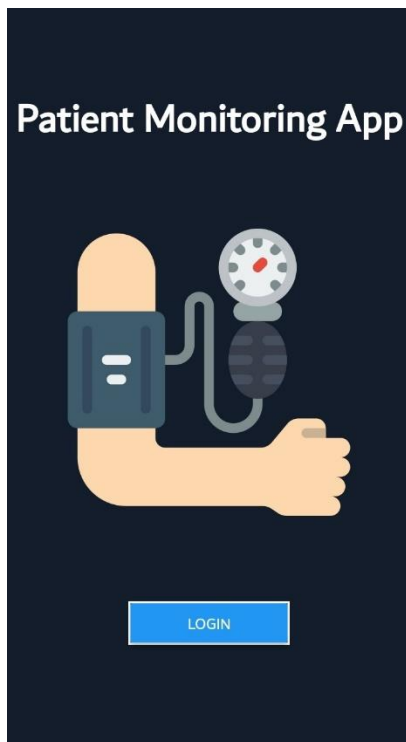
FORGOT PASSWORD

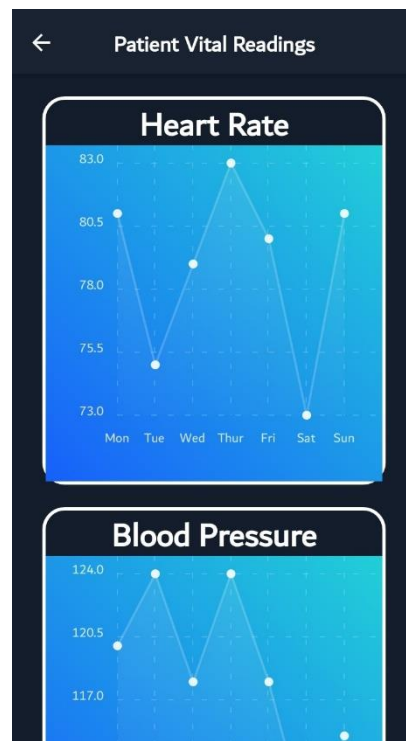
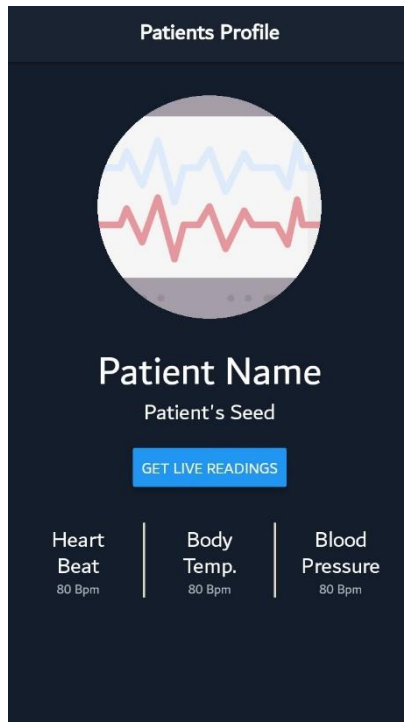
Enter Seed \*

GET PASSWORD

The screenshot shows a web browser window titled "Doctor's Portal" with the address bar displaying "localhost:3000". The page has a green header bar with the text "Welcome, Doctor" on the left and two buttons, "HOMEPAGE" and "ADD PATIENT", on the right. The main content area is titled "Add Patient" in a large, light gray font. Below the title, there is a form with three input fields: "First Name\*", "Last Name\*", and "Age in year(s)\*". A green button labeled "ADD PATIENT" is positioned below the form fields.

### 8.2.2 Mobile Application





### 8.3 Screen objects and actions

#### 8.3.1 Doctor:

- Create Account: Lets doctor to request for account by generating a unique seed and updating profile data to store it in Database
- View Profile: Shows the profile data
- Add Patient: Lets doctor add a patient under their seed, generates patient credentials
- Discharge Patient: Lets doctor discharge patient by removing their record associated with seed
- View Real time data: Allows doctor to ping the hardware device at any time to record and return the patient's vital signs at a given time
- View History: Allows doctor to request the saved data of any patient stored in IoTA Ledger
- Visualize Data: Doctor can use this to fetch data from ledger and visualize it
- Generate report: Get patient's report based on patient's data of specific time

#### 8.3.2 Patient:

- Login: Lets patient login to mobile app using credentials provided by doctor
- View Profile: Shows the patients' profile data
- View Real time data: Patients can see their real time data
- View History: Let patients to see their previous data

## 9. Appendix I

- IOTA ledger website: <https://www.iota.org/>
- IOTA MAM : <https://docs.iota.org/docs/client-libraries/0.1/mam/introduction/overview>
- Diagrams study : <https://tallyfy.com/uml-diagram/#:~:text=A%20UML%20diagram%20is%20a,document%20information%20about%20the%20system.>
- <https://www.lucidchart.com/pages/data-flow-diagram#top>
- <https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/>
- Tangle : <https://www.iota.org/>
- <https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4>
- Tangle Explorer: <https://thetangle.org/>
- IRI Node : <https://docs.iota.org/docs/node-software/0.1/iri/how-to-guides/install-iri>
- <https://docs.iota.org/docs/node-software/0.1/iri/introduction/overview>
- MongoDB : <https://www.mongodb.com/>
- Reverse Proxy Server : <https://www.ericzhang.me/reverse-proxy-ddos-protection/>

## 10. Demonstration Video Link

[https://drive.google.com/file/d/1HUEwdn\\_ZhTj99BymTGOxLiZFwhqeW\\_1l/view?usp=sharing](https://drive.google.com/file/d/1HUEwdn_ZhTj99BymTGOxLiZFwhqeW_1l/view?usp=sharing)

## 11. Github Links

### 11.1 Project Code Link

<https://github.com/foxx-2/Health-Monitoring-with-Theta-Middleware>

### 11.2 Hasnain Khawaja's Github Profile Link

<https://github.com/hasnainkhawaja47>

### **11.3 Noman Nasir Minhas's Github Profile Link**

<https://github.com/foxx-2>

### **11.4 Muhammad Wahaj Mubeen Github Profile Link**

<https://github.com/WahajMagazine341>