HOLISTIC HEALTH CARE APPLICATION

TEAM ELITE:

P. Ramya Devi [2019103568]

R.V. Kaviya [2019103538]

Thumula Praneeth Roa [2019103575]

TABLE OF CONTENTS:

ABSTRACT

- The main aim of the application is to provide holistic health care facilities to the user which includes doctor appointments,ordering medicines booking ambulance and alert system by the use of sensors.
- The major stakeholders are patient,doctor,pharamacy and ambulance service.
- The patient can book appointments,choose diet, book ambulance,order medicines through pharmacy,calculate BMI and they can also configure alert by giving the timings of their regular medicines.A unique e-health record will be assigned and maintained for storing the health details of the patient.The measurements of the vitals will we received through sensor connected to the application.
- The doctor can provide consultation and can also connect with the senior physician for higher medical advice.
- The pharmacy takes up the order and can maintain their own medicine catalogue depending on their medicine availability.
- The ambulance service provider can take up the bookings if they wish and can also access the emergency info of the patient.

SRS

1.INTRODUCTION

1.1 Identification

This document shall be identified as the Software Requirements and Specification of Smart HealthCare Application, Version 1.0 designed by the team Elite.

1.2 Purpose

The intended purpose of this software is to design a system that favours creating a digitized healthcare system, connecting available medical resources and healthcare services and enables remote health monitoring and emergency notification system.

1.3 Intended Audience

The intended audience of this system are Patients, Doctors, Pharmacy and Ambulance service.

1.4 Proposed Mini-Case Study Statement

The Smart HealthCare application is a system build with technologies of IoMT and AI. This system is designed to provide remote health monitoring, self-health mentorship to patients, fulfilling the overall medicinal needs of patients.

1.6 Benefits of the system

This software is user friendly in terms of ease of operation and it also provides a hub a health care services. It is multifunctional and discardes the need of using individual software for different health related services. It also keep continuous track of data's and provides alert ,thus delivering a cent percent complete health service package.

◦

2.1 KEY MODULES

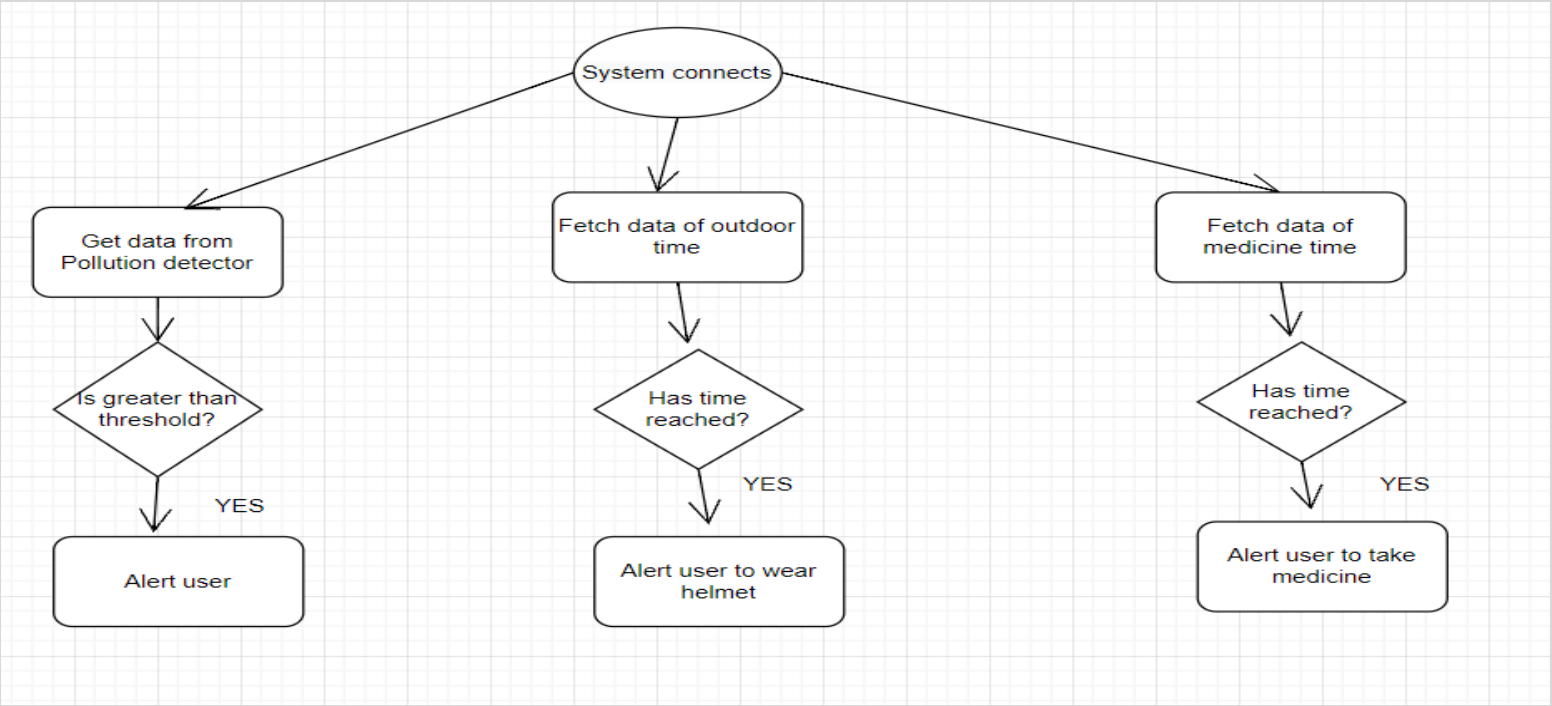
Code	Key Modules	Purpose	Use cases	Actors
A	General health regulation	Alert patients by computing fetched data from health record and sensors	Alert user, Detect pollution	System, sensor,Patient
B	Appointment handler	Allows patient to book doctor specific to their health issues	Book appointments	Patient, System,
C	Fitness Mentor	Computes BMI and suggests diet plan accordingly	Suggest diet, compute BMI	System,patient

A. General Health Regulation

A.1 Scope

This process involves the collection of data from sensors which includes vitals sensor,pollution detector and vehicle detector and the system alerts the patient accordingly such as to wear face mask if pollution detected and by fetching details of patients,alert patient to wear helmet and also give alerts to take medicine at due time.

A.2 Process flow



A.3 Process description

A.3.1 Key fields

Pollution detection

Medicine time

Outdoor time

A.3.2 Functional Specification

Patient screen: Patient enter their details using this screen and also updates and views the data stored in health record.

Alert message will be displayed in this screen by the system using those data for computation

A.3.3 Process Behaviour

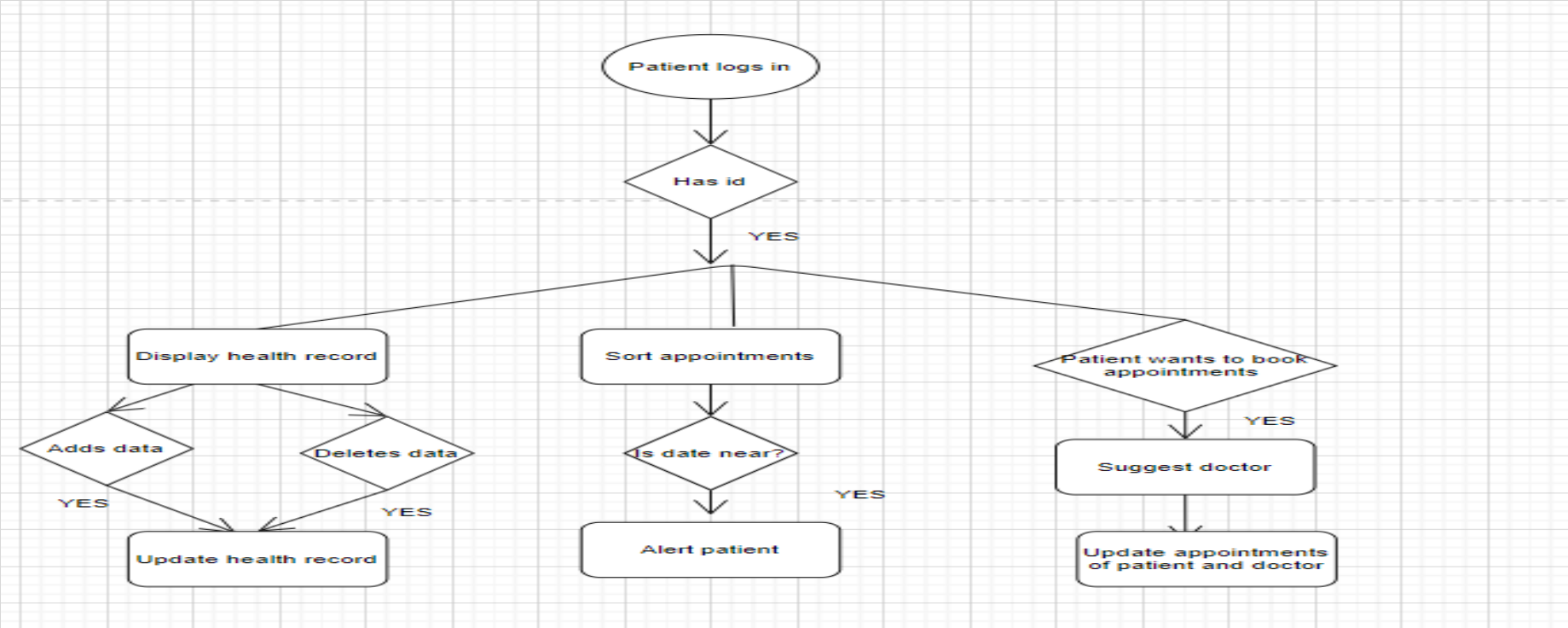
Action	Behaviour
On detecting pollution	Sensor detect pollution, this data gets collected and alert will be generated.
On update	Data gets updated now and then once the sensor send their data to the system

B.APPOINMENT HANDLER

B.1. Scope

The registered patient can add / delete the details of his health record and system will automatically update the record. If patient wants to book appoinment the system automically displays the domain of patient's health need. The system also sorts the appoinment based on date.Once date is near it alerts the patient

B.2 Process flow



B.3 Process Description

B.3.1. Key field

Patient details

Doctor detail

B.3.2 Functional specification

Patient screen: Display the patient details and if they want to update details they can and general alert will be shown here.

Appoinment screen: patient books their appointments through this screen and upcoming and appointment history can also be viewed.

B.3.3 Process Behaviour

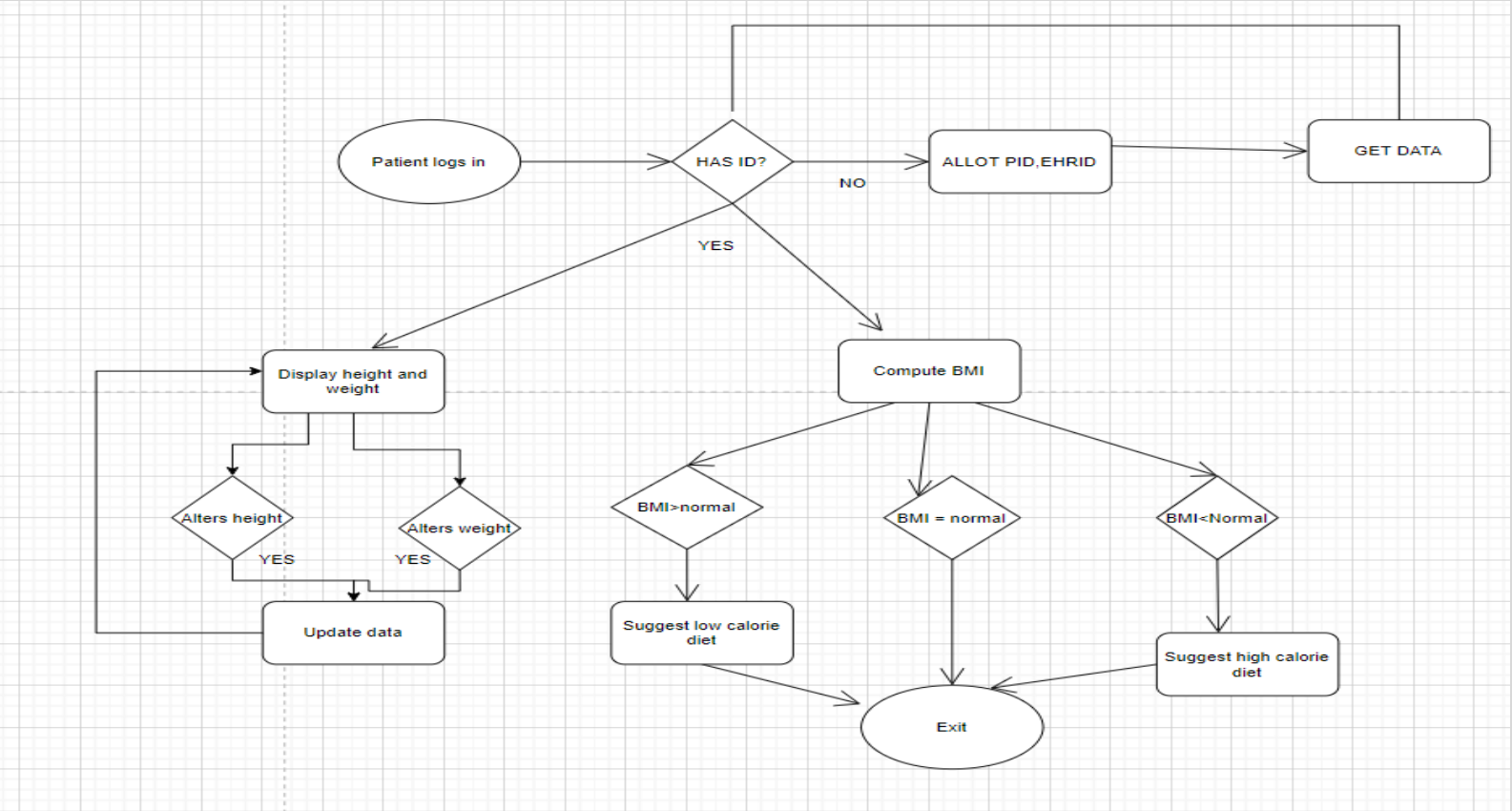
Action	Behaviour
On insertion	The system gets details and update them in Health records
On deletion	The system deletes the selected data from health records

C.FITNESS MENTOR

C.1. Scope

The patient BMI will be computed based on inputs given by the patient namely height and weight. According to the BMI value diet will be suggested.

C.2. Process flow



C.3. Process Description

C.3.1. Key field

Height, weight details

BMI value

Diet plan

C.3.2. Functional Specification

Patient screen: Patient gives their data and updates the data as per the needs through this screen.

Diet screen: The BMI value and the suitable diet will shown in this screen.

C.3.3 Process behaviours

Action	Behaviour
On update details	Patient can update their height; weight details and it will be stored in health record and by using those details system can compute BMI and based on its system suggest appropriate diet plan

FUNCTIONAL REQUIREMENTS

Security:

- Patient Identification: The system needs the patient to recognize herself or himself using the phone.
- Logon ID: Any users who make use of the system need to hold a Logon ID and password.
- Modifications: Any modifications like insert, delete, update can be synchronized quickly

Performance:

- Response Time: The system provides acknowledgment in just one second once the 'patient's information is checked.
- Capacity: The system needs to support at least 1000 people at once.
- User-Interface: The user interface acknowledges within five seconds.

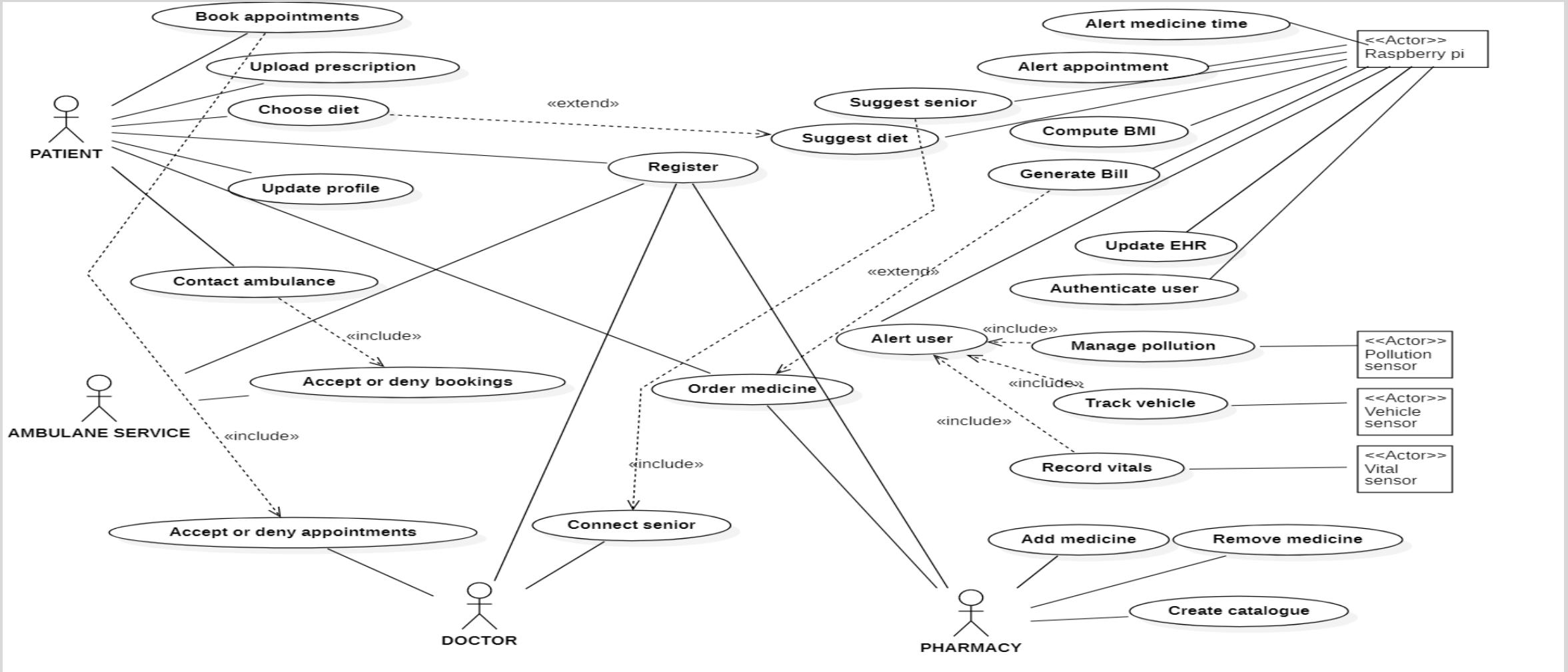
Maintainability:

- Back-Up: The system offers the efficiency for data backup.
- Errors: The system will track every mistake as well as keep a log of it.

Reliability:

- Availability: The system is available all the time.

USE CASE DIAGRAM



Overview

- The actors are patient,ambulance service,pharmacy,Raspberry pi ,pollution sensor,vehicle sensor and vital sensor.
- The patient can book appointments with the doctor an application will suggest the doctor based on the domain given by the patient.The patient registers himself and all his deatails are stored in health record which can be updated whenever required.The patient can also upload prescription or can himself add medicines for placing medicine order.He can also contact ambulance service.
- The doctor accepts or denies the appointments based on his/her availability and can also connect with the senior doctor of the same domain for medical advice.
- The pharmacy member can add or remove medicine based on the stock availability and can also maintain their catalogue.The pharmacy can also view the prescription uploaded by the patient and can add up medicine to the patient' cart.
- The Raspberry pi is coupled with the devices through the sensors/wires.It is the computational node where all kind of computation like alert generation,bill generation,BMI calculation,updation of health record ,authentication of user,suggestion of senior doctor and diet takes place
- The pollution sensor keeps track of the level of pollutants and generates an alert when it goes beyond threshold,the vehicle sensor alerts to wear helmet when vehicle is turned on.The vitals sensor reads the value of the vitals and send it to the system where computation takes place and alert is created when vitals go beyond normal

Use case Name	Compute BMI
Description	This use case allows the system to compute BMI from the data given by patients.
Primary Actor	System
Precondition	1.The user should be registered and autheticated. 2. All required data should be given by the patients. 3.Standard BMI values and method for computing BMI should be predefined.
Main flow	1.patient log in to their home page 2. if he/she is new creates account and system autheticates 3. System computes the BMI by dividing the weight in kg by square of height in metres. 4.If the BMI value is greater than normal BMI for the patient's gender and age it will store the patient's condition as 'obese' and if it is less than normal the condition will be computed as 'underweight' and if it's equal to normal it will be stored as 'normal'.
Alternate flow	Missing patient details such as height or weight. 1.System prompt for details which you have missed 2.Usecase resume at step 3 3b. Invalid height and weight details 1.The system display "Invalid Entry". 2.System prompts for the valid details 3.Use case resume at step 3

Use case Name	ALERT USER
Description	This use case allows system to alert user whenever data sent by the sensor goes beyond normal level.
Primary actor	System
Include use case	1.Record vitals 2.Manage pollution 3.Track vehicle
Precondition	1.The critical values for vitals and pollution level will be preset 2. Data about the outgoing time will also be obtained from the user. 3. Sensors are autheticated and set in pace for measuring data
Postcondition	1.Alert message gets displayed on the user screen and in case of vitals alert. Then alert message will also send to ambulance service
Main flow	1.System gets data from sensor and stores. 2. If vitals do beyond normal alert the patient 3. If the level of nitrous oxide is high checks for level of sulphur oxide. If it is also high goes for checking of carbon monoxide.If all these are conditions are satisfied alert goes for patient to wear mask 4. The vehicle detector present in the bike returns 1 if vehicle starts and 0 if vehicle is off. The system receives the boolean value and if it receives 1 from the detector it alerts the user.
Alternate flow	1.if computed data is not high, system not give any alerts it keeps on computing the data from the sensor 2.not give any alerts if there is no deduction of pollution 3.not give alerts if there is no deduction of vehicles

Use case Name	Upload prescription
Description	This use case allows the patient to upload the prescribed medicine list
Primary actor	Patient
Precondition	1.The patient should be registered and authenticated 2.The patient should also allow access to the camera

Use case Name	Add medicine
Description	This use case allows the pharmacy to add medicine in medicine catalogue based on the availability of the medicine stock
Primary actor	Pharmacy
Precondition	1.Pharmacy can register and authenticate 2Patient can register and authenticate 3. Pharmacy maintains a record of medicine list included with the cost.
Main flow	If the pharmacy purchases a new medicine it can add it in the existing list. The cost and description of the new medicine can be added to the specified fields.

Use case name	Update health record
Description	The patient can add or delete or alter the details of the health record whenever required and the application automatically updates it.
Primary actor	Patient
Precondition	user should be registered and authenticated as patient and the previous details of the health record will be displayed for the view of patient
Main flow	1.Patient logs in to their home page 2. The system authenticates the user 3.The user clicks the data he/she wants to changes and enters the new data 4.system updates respective patient records based on their entered data.
Alternate flow	1a. Invalid password or userid entry 1.System prompts for correct password or userid 2.use case resume at step 3 If the user is new he registers himself and gets autheticated and enters his health details.Once he is done he can update the details whenever required.

Use case name	Order Medicine
Description	This use case allows patient to order their medicine in their desired pharmacy
Primary actor	Patient
Precondition	The pharmacies should have already uploaded their items present in their medicine catalogue.
Extend use case	Generate bill
Main flow	1.Patient chooses the medicine and adds into their cart 2.He can add or delete the items in the cart 3.The pharmacy can send confirmation mail on receiving the payment for the order

Use case Name	Book appointment
Description	This use case allows the patient to book their desired doctor's appointment
Primary actor	Patient
Include use case	Accept/deny appointments
Precondition	user should be registered and authenticated as patient
Main flow	The patient chooses the doctor he want to book appointment The patient then waits for the acceptance of appointment by the doctor Once confirmed the details of the appointment will be stored in database

Use case Name	Authenticate user
Description	The system could authenticates all the user who logs in by crosschecking the credentials given by them
Primary actor	System
Precondition	The user should have given their details for crosschecking
Main flow	1.The system prompts for user's userid and password 2.User enters the details 3.The system checks the entered details with the previous registered details 4.If the string matches the user will be automatically directed to the home page.
Alternate flow	If the String does not match the system prompts a message The steps resume from step 2 The user can have a try out for 3 times.At 4th time if he enters wrong details the user's account permission will be cancelled and he should go for creating a new account

Use case Name	Suggest Diet
Description	The system suggests the user with a diet depending on the health details in the patient's profile
Primary actor	System
Extend use case	Choose diet
Main flow	1.Patient's food preference will be collected 2.Appropriate diet plan will be suggested 3.The overall calorie will be calculated for the items choosen by the patient

Use case Name	Suggest senior
Description	The system suggests the senior doctor in the field same as the doctor requesting it
Primary actor	System
Include use case	Connect senior
Precondition	All the doctors should have given their field and designation during registration The doctor makes request for senior doctor's consultation
Main flow	1.System takes the field of the user who is requesting senior doctor as string 1 2.It compares the string 1 with all the other doctors field. 3.If it matches looks for doctor's designation if it is senior it prompts the doctor's details to the the user

Use case Name	Alert medicine time
Description	The system generates an alert when the time of the medicine is near
Primary actor	System
Precondition	The time of the medicines taken regularly by the patient should have already been stored in database. The starting date and ending date of the medicines should have been stored already
Main flow	The system checks the present time and date of all medicines If it matches it prompts a notification

Use case name	Generate Medicine bill
Description	Once the patient paces the order and the payment is made the bill will be generated by the system
Primary actor	System
Precondition	The order should be placed by the patient
Main flow	The system calculated the individual medicine cost based on the count and prices and finally add the cost of all medicines to generate a bill

Use case Name	Update profile
Description	If the patient wishes to update his details such as age,address,mobile number he could do and the system will update the profile in the database
Primary actor	Patient,System
Main flow	1.The patient logs in 2.The patient enter his/her profile details such as name,age,gender and address. 3.If the details are already there,the patient selects the field which he wants to update and enters the new data in the specified field 4.Once entered the system automatically updates it.

Use case Name	Accept/Deny appointments
Description	This use case allows the Doctor to accept or deny appointment
Primary actor	Doctor
precondition	The patient requests for the appointment of the doctor
Main flow	1.the user could register and authenticate as doctor 2.The newly booked appointmnet will be shown at the top 3.The doctor can either accept or deny the appointments

Use case Name	Create catalogue
Description	This use case allows pharmacy staff create a catalogue based on the medicines they have
Primary actor	Pharmacy
Precondition	The user could register and authenticate as pharmacy staff.
Main flow	1.user log in to their Pharmacy home page 2.The user can create their own medicine catalogue 3.The medicine specification for each medicine in the medicine catalogue is to be given.

Use case Name	Record vitals
Description	The sensor sends the recorded vital details such as O2 level,Body temperature and pulse
Primary actor	Sensor
Precondition	Sensor should have made a conection with the application
Main flow	The sensor sends the recorded data to the system The system stores it in patient's health record The system keeps a track of vitals and compares it with the standard normal vitals.

Use case Name	Connect senior
Description	The doctor can connect with the doctor of senior level in the same field
Primary actor	Doctor
Precondition	The field and designation of the doctor should have been stored earlier
Main flow	1.Doctor logs in using the credentials 2.The doctor can have a look on the doctors suggested by the system 3.The doctor can book his appointment with the senior doctor.

Use case Name	Manage pollution
Description	The pollution level will be send to the system by the sensor and system keeps on moniotoring the levels
Primary actor	Sensor
Main flow	The sensor will send a signal to the system whenever pollution is detected in the surroundings of the user

Use case Name	Track vehicle
Description	The sensor keeps an track on the on and off status of the vehicle
Primary actor	Sensor
Main flow	The sensor sends the signal to the system whenever the vehicle gets on

Use case Name	Choose diet
Description	The patient chooses the diet based on the suggestions given by the app
Primary actor	Patient
Pre condition	The food preferences of the patient is got through input
Main flow	1. The patient logs in 2. System suggests the diet based on the patient's health details 3. The patient can either accept the suggested diet or he can choose his own diet
Post condition	The calories for the chosen diet is calculated and displayed.

Use case Name	Register
Description	All the user are supposed to create and account and register themselves and give their details
Primary actor	All the users
Main flow	The user gives their id and password which will be stored by the system The patient,doctor,pharmacy,ambulance service should fill their details respectively

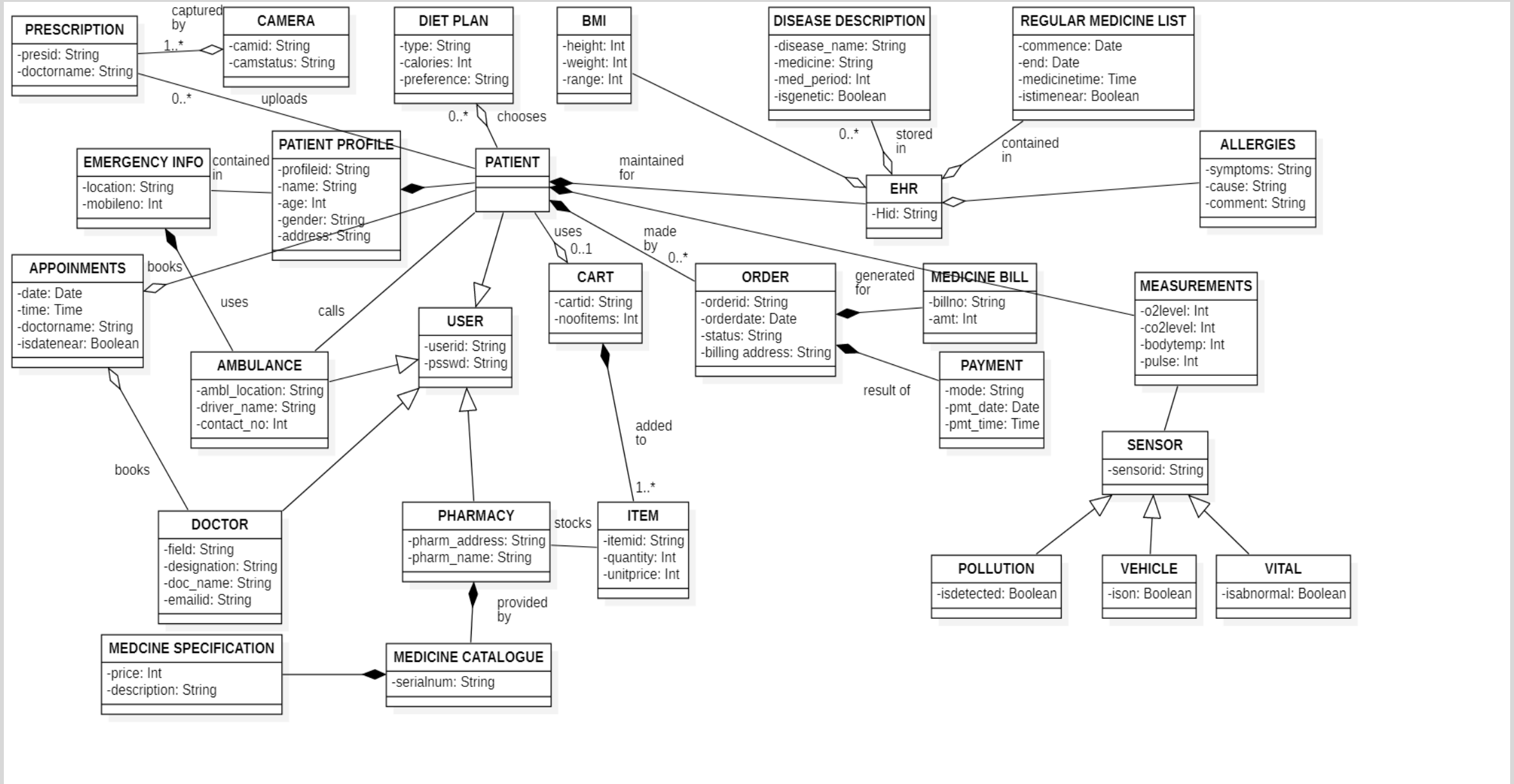
Use case Name	Contact ambulance
Description	The patient can book an ambulance whenever he/she requires.
Primary actor	Patient
Include use case	Accept/deny bookings
Main flow	1. The patient logs in 2. Patient chooses the ambulance he/she wants to book. 3. The amublance service accepts the booking. 4. The patient receives the confirmation mail
Post condition	The patient emergency info such as gps location is collected and made available to the ambulance service

Use case Name	Accept/deny bookings
Description	The ambulance service can either accept or deny the booking he has got.
Primary actor	Ambulance service
Pre condition	The Ambulance is registered and authenticated The details of the ambulance is already got through input The emergency info of the patient is made availabe to the ambulance service
Main flow	1. The user logs in 2. System shows the newly got booking based on the calculated nearest location 3. The ambulance service accepts the appointment 4. Confirmation mail is send to the patient

Use case Name	Remove medicine
Description	This use case allows the pharmacy to remove medicine in medicine catalogue based on the availability of the medicine stock
Primary actor	Pharmacy
Precondition	1.Pharmacy can register and authenticate 2Patient can register and authenticate 3. Pharmacy maintains a record of medicine list included with the cost.
Main flow	If the pharmacy goes out of stock of a medicine it can remove it from the existing list. The updation will be automatically made available in the database.

Use case Name	Alert appointment
Description	The system generates an alert when the time of the appointments is near
Primary actor	System
Precondition	The appointment is booked by the patient and the doctor has confirmed the appointment
Main flow	The system checks the present date and time with the upcoming appointmnets If it matches it prompts a notification

DOMAIN MODEL


PRESCRIPTION
-presid: String
-doctorname: String
captured by
1..*
CAMERA
-camid: String
-camstatus: String
0..*
uploads
DIET PLAN
-type: String
-calories: Int
-preference: String
0..*
chooses
BMI
-height: Int
-weight: Int
-range: Int
DISEASE DESCRIPTION
-disease_name: String
-medicine: String
-med_period: Int
-isgenetic: Boolean
REGULAR MEDICINE LIST
-commence: Date
-end: Date
-medicinetime: Time
-istimenear: Boolean
0..*
stored in
contained in
EMERGENCY INFO
-location: String
-mobileno: Int
contained in
PATIENT PROFILE
-profileid: String
-name: String
-age: Int
-gender: String
-address: String
PATIENT
maintained for
EHR
-Hid: String
ALLERGIES
-symptoms: String
-cause: String
-comment: String
uses
0..1
made by
0..*
APPOINMENTS
-date: Date
-time: Time
-doctorname: String
-isdatenear: Boolean
books
uses
calls
CART
-cartid: String
-noofitems: Int
ORDER
-orderid: String
-orderdate: Date
-status: String
-billing address: String
generated for
MEDICINE BILL
-billno: String
-amt: Int
MEASUREMENTS
-o2level: Int
-co2level: Int
-bodytemp: Int
-pulse: Int
USER
-userid: String
-psswd: String
AMBULANCE
-ambl_location: String
-driver_name: String
-contact_no: Int
result of
PAYMENT
-mode: String
-pmt_date: Date
-pmt_time: Time
added to
1..*
SENSOR
-sensorid: String
books
DOCTOR
-field: String
-designation: String
-doc_name: String
-emailid: String
PHARMACY
-pharm_address: String
-pharm_name: String
stocks
ITEM
-itemid: String
-quantity: Int
-unitprice: Int
POLLUTION
-isdetected: Boolean
VEHICLE
-ison: Boolean
VITAL
-isabnormal: Boolean
provided by
MEDCINE SPECIFICATION
-price: Int
-description: String
MEDICINE CATALOGUE
-serialnum: String

Data dictionary:

OBJECT	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
1.PATIENT PROFILE	Profile_id	String	It contains the complete details of patients
	Name	String	
	Age	Int	
	Gender	String	
	Address	String	
2.USER	User_id	String	Every user has user id and password which is verified
	Psswd	String	each and every time they login
3.EHR	Hid	String	Every patient is assigned a e-hid and is maintained a health record
4. MEASUREMENT	O2_level	Int	The readings measured by the sensor are stored and
	CO2_level	Int	and checked for abnormalities
	bodytemp	Int	

OBJECT	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
	Pulse	int	
5.REGULAR MEDICINE LIST	Commence	date	The list of medicines takes regularly by the patient is stored
	End	date	When the time is near the patient is notifies
	medicinetime	time	
	istimenear	boolean	
6.ALLERGIES	symptoms	string	The allergies of the patient are stored in health record.
	cause	String	The doctor can have a look at them before proceeding for prescribing meds
	comments	String	
7.BMI	height	int	The BMI value of the patient is computed and checked against normal range
	weight	int	

OBJECT	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
	Range	String	
8.SENSOR	Sensorid	String	
	Isdetected	Boolean	Pollution sensor detects pollution
	Isabnormal	Boolean	Vital sensor sends the level of various vitals
	ison	Boolean	Vehicle sensor sends the status of the vehicle
9. PRESCRIPTION	prseid	String	The prescription is uploaded by the patient for buying medicines
	doctorname	String	
10.CAMERA	camid	String	The camera is connected to capture the image of the prescriptiom
	Cam_status	String	
11.DISEASE DESCRIPTION	Disease_name	String	The list of previous diseases the patient has is got and made available to the doctor through E-health record

OBJECT	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
	Medicine	String	
	Med_period	Int	
	isgenetic	boolean	
12.APPOINTMENTS	Date	Date	The details of all the appointments the patieny has is stored
	Time	Time	At respective date and time the patient is notified
	isdatenear	Boolean	
	doctorname	String	
13.DIET PLAN	Type	String	The food items are made available for the patient to create their own diet plan. In addition the system itself suggests the complete diet plan
	Calories	int	
	preference	String	

OBJECT	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
14.EMERGENCY INFO	Location	String	The emergency details of the patient which is going to be accessed by the ambulance provider
	Mobileno	Int	
15.BILL	Billno	String	For each and every order the total amount is calculated and bill is generated
	Amount	Int	
16.DOCTOR	Field	String	It the basic profile of the doctor
	Designation	String	
	Doc-Name	String	
	Email_id	String	
17.AMBULANCE	Ambl_loc	String	It contains the basic details of the ambulance service provider
	drivername	String	

	ATTRIBUTE	DATATYPE	DESCRIPTION
	Contact_no	Int	
18.PHARMACY	Pharm_name	String	The pharmacy basic details are got and registered
	Pharm_address	String	
19.ITEM	Item_id	String	The list of medicines available for sale
	Quantity	Int	
	unitprice	Int	
20.CART	Cartid	String	The basket in which the patient could pick up the medicines and edit the quantity
	Noofitems	Int	
21.ORDER	Orderid	String	The medicine order placed by the user
	orderstatus	String	

	ATTRIBUTE	DATA TYPE	DESCRIPTION
	orderdate	Date	
	billingaddress	String	
22.MED CATALOGUE	serialnum	String	It the complete list of medicine available in the pharmacy along with it's price
23.SPECIFICATION	price	Int	It's the specification for each and every medicine available in the pharmacy
	description	String	
	mode	String	
	Pmt_date	Date	
	Pmt_time	time	

Relationship and multiplicity

1.Relationship name:	Stocks
Meaning:	The pharmacy stocks the item
Multiplicity:	1. 1 to many pharmacies can stocks 1 to many number of items. 2. 1 to many number of items can be stocked by 1 to many number of pharmacies

2.Relationship name:	captured by
Meaning:	The prescription is captured by the camera
Multiplicity:	1. A camera can capture 1 to n number of prescriptions 2. 1 to n number of prescriptions can be captured by a camera

3.Relationship name:	provided by
Meaning:	The medicine catalogue is provided by the pharmacy
Multiplicity:	1.1 A pharmacy can provide one medicine catalogue 1.2 A medicine catalogue can be provided by a pharmacy

4.Relationship name:	Added to
Meaning:	The item is added to the cart
Multiplicity:	1. 1 to n number of items can be added to the cart 2. A cart has 1 to n number of items

5.Relationship name:	Maintained for
Meaning:	A E-health record is maintained for a patient
Multiplicity:	1. A E-health record will be maintained for a patient 2. A patient can have only one EHR

6.Relationship name:	contained in
Meaning:	The emergency info is contained in patient profile class
Multiplicity:	1.1 A patient profile contains a emergency info 1.2 A emergency info is contained in a patient profile

7.Relationship name:	Stored in
Meaning:	The disease description is stored in EHR
Multiplicity:	1. Many disease descriptions can be stored in a EHR 2. A EHR can store multiple disease descriptions

8.Relationship name:	Chooses
Meaning:	The patient chooses the diet plan
Multiplicity:	1.1 A patient can choose 0 to n number of diet plans 1.2 0 to n number of diet plans can be chosen by a single patient

9.Relationship name:	Generated for
Meaning:	A medicine bill is generated for a order
Multiplicity:	1.1 A order results in generation of single medicine bill 1.2 A single medicine bill is a generated for a order

10.Relationship name:	Result of
Meaning:	The payment is the result of order.
Multiplicity:	1. A order results in a single payments 2. A payment is a result of a single order

11.Relationship name:	Books
Meaning:	The user books the appointments
Multiplicity:	1.1 A user can book 1 to n number of appointments 1.2 1 to n number of appointments can be booked by a user

12.Relationship name:	Calls
Meaning:	The patient calls the ambulance
Multiplicity:	1. A patient calls the ambulance 2. A ambulances can be called by a patient

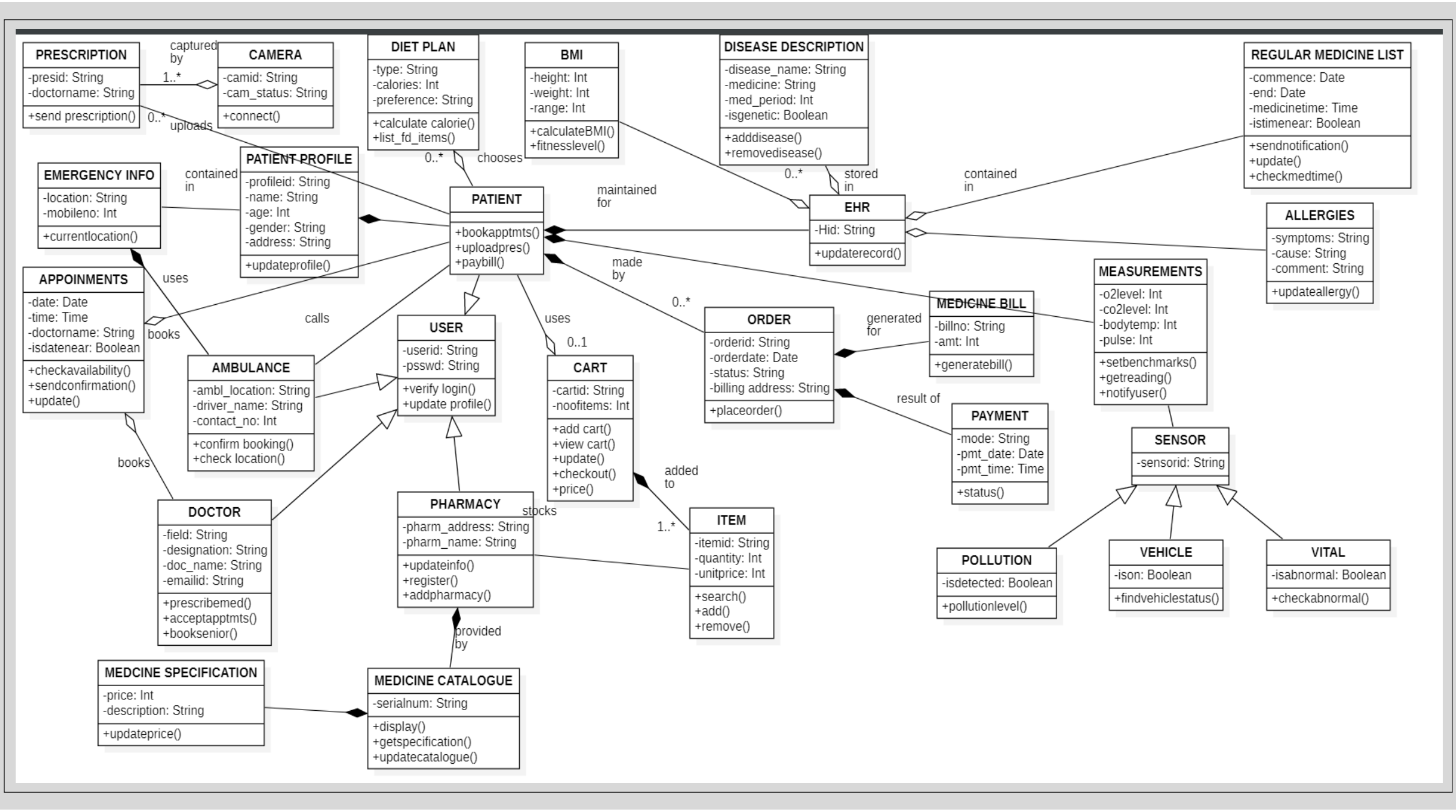
13.Relationship name:	Uploads
Meaning:	Patient can upload many prescriptions
Multiplicity:	1. A patient can upload 1 to many prescription 1.2 1 to n number of prescriptions can be uploaded by the patient

14.Relationship name:	Contained in
Meaning:	Regular medicine list is contained in EHR
Multiplicity:	1. A EHR can contain 0 to many regular medicine list 2. 0 to many regular medicine lists can be contained in a EHR

15.Relationship name:	uses
Meaning:	The patient uses the cart
Multiplicity:	1. 0 to 1 cart can be used by a patient 2. A patient can use 0 to 1 cart

16.Relationship name:	Made by
Meaning:	The orders are made by the patient
Multiplicity:	1. 0 to many orders can be made by a patient 2. A patient can make 0 to many orders

CLASS DIAGRAM


PRESCRIPTION
-presid: String
-doctorname: String
+send prescription()
captured by
1..*
CAMERA
-camid: String
-cam_status: String
+connect()
0..*
uploads
DIET PLAN
-type: String
-calories: Int
-preference: String
+calculate calorie()
+list_fd_items()
BMI
-height: Int
-weight: Int
-range: Int
+calculateBMI()
+fitnesslevel()
DISEASE DESCRIPTION
-disease_name: String
-medicine: String
-med_period: Int
-isgenetic: Boolean
+adddisease()
+removedisease()
REGULAR MEDICINE LIST
-commence: Date
-end: Date
-medicinetime: Time
-istimenear: Boolean
+sendnotification()
+update()
+checkmedtime()
EMERGENCY INFO
-location: String
-mobileno: Int
+currentlocation()
contained in
PATIENT PROFILE
-profileid: String
-name: String
-age: Int
-gender: String
-address: String
+updateprofile()
0..*
chooses
PATIENT
+bookapptmts()
+uploadpres()
+paybill()
maintained for
0..*
stored in
EHR
-Hid: String
+updaterecord()
contained in
ALLERGIES
-symptoms: String
-cause: String
-comment: String
+updateallergy()
uses
APPOINMENTS
-date: Date
-time: Time
-doctorname: String
-isdatenear: Boolean
+checkavailability()
+sendconfirmation()
+update()
books
calls
AMBULANCE
-ambl_location: String
-driver_name: String
-contact_no: Int
+confirm booking()
+check location()
USER
-userid: String
-psswd: String
+verify login()
+update profile()
uses
0..1
CART
-cartid: String
-noofitems: Int
+add cart()
+view cart()
+update()
+checkout()
+price()
made by
0..*
ORDER
-orderid: String
-orderdate: Date
-status: String
-billing address: String
+placeorder()
generated for
MEDICINE BILL
-billno: String
-amt: Int
+generatebill()
MEASUREMENTS
-o2level: Int
-co2level: Int
-bodytemp: Int
-pulse: Int
+setbenchmarks()
+getreading()
+notifyuser()
result of
PAYMENT
-mode: String
-pmt_date: Date
-pmt_time: Time
+status()
SENSOR
-sensorid: String
books
DOCTOR
-field: String
-designation: String
-doc_name: String
-emailid: String
+prescribemed()
+acceptapptmts()
+booksenior()
PHARMACY
-pharm_address: String
-pharm_name: String
+updateinfo()
+register()
+addpharmacy()
stocks
added to
1..*
ITEM
-itemid: String
-quantity: Int
-unitprice: Int
+search()
+add()
+remove()
POLLUTION
-isdetected: Boolean
+pollutionlevel()
VEHICLE
-ison: Boolean
+findvehiclestatus()
VITAL
-isabnormal: Boolean
+checkabnormal()
provided by
MEDCINE SPECIFICATION
-price: Int
-description: String
+updateprice()
MEDICINE CATALOGUE
-serialnum: String
+display()
+getspecification()
+updatecatalogue()

CRC

CLASS	ATTRIBUTES	RESPONSIBILITIES	COLLABORATORS
User	- id - password	+ verify login() + update profile()	Patient Doctor Pharmacy Ambulance
Patient		+ book appointments() + Upload prescription() + Pay medicine bill()	Patient profille Appointments Prescription Diet plan EHR Cart Order Measurement Ambulance
Patient profile	-Profile id -Name -Age - Gender -Address	+Update profile	Emergency info
Ambulance	-Location -Driver Name -contact Number	+Check location() +confirm booking()	Patient Emergency info
Doctor	-Field -Designation -Name	+Prescribed med() +Accept appointments() +Book senior appointment()	Appointments

7.	**Pharmacy**	**-Address** **-Name**	**+Update info()** **+Register()** **+Add pharmacy()**	**Item** **Medicine catalogue**
8.	Prescription	-Prescription id -Doctor name	+Send prescription()	Camera Patient
9.	Cart	- Cart Id -No of items	+Add cart item() +Update quantity() +View cart() +Check out() +Cart price()	Patient Item
10.	Medicine catalogue	-Serial Number	+Display() +Get specification() +Update catalogue()	Pharmacy Medicine Specification
11.	EHR	-H id	+Update record()	BMI Patient Allergies Regular medicine list Disease description
12.	Order	-order id -order date -order Status -Billing address	+Place order()	Patient Medicine Bill Payment
13.	Sensor	-Sensor id		Pollution Vehicle Vital Measurement
14.	Measurement	-O2 level	+Sent benchmark()	Sensor

Classes and their collaborators description:

i. The patient books appointments,calls ambulance,uses cart,uploads prescription, chooses diet and has a profile and has a set of measurements
ii. EHR is maintained for a patient
iii. BMI,disease description,regular medicine list,allergies are stored in the EHR
iv. Emergency info is contained in the patient profile and is used by the ambulance
v. The prescription is captured by the camera
vi. The doctor books his/her appointment with the senior
vii. The users namels ambulance,patient,doctor,pharmacy are registered and their login is verified
viii. The pharmacy stocks the items which is added to the cart and provides medicine catalogue which in turn has medicine specification
ix. Order is made by the patient which results in payment and generation of medicine bill.
x. The three sensors namely pollution,vehicle,vital sends data which is shown to the patient

Description:

1.Patient profile :

It is identified as a class which contains the complete details of patients. And it could be viewed by patients after logging into the system by referring his/her profile they can get clear view of their health conditions

2.Regular Medicine List

This class contains the details of medicine which has to be taken every day ,Details include the time at which the patient should take up those medicines, and start and end date of taking medicine, so by computing those data the system could alert patients appropriately

3.Allergies

This class includes the details of Patient's allergies. Details contains the symptoms, cause and comments .

4.Disease's description

This class contains the name of disease which the patient has been diagnosied earlier and prescribed medicines for that disease , and medicine time and whether the disease has any genetic cause or not.

5.BMI

This class contains the height and weight details of the patients, so that the system could compute the BMI and range of BMI values for computing the health condition of the patient.

6.Diet plan

The diet plan is suggested which includes amount of calorie and type of diet as whether underweight/overweight diet and preference of food as veg or non veg indian/western/italian etc

7.Emergency info

Contact number and location of the patient are included in this class.This will be used by the ambulance service when the patient books an ambulance.

8.Ambulance

Id and location of ambulance details specifically contained in this class.The location will be computed for finding the nearby ambulance from the patient location.

9.EHR

It maintains the complete health details of the patient. Each and every user will have a unique id which holds the information of the patient. The doctors can also view the health record of their patient.

10.Measurements

This class includes vitals measurements of the patients sent by the sensors like o2 level,co2 level, body temperature, pulse rate. The system does a computation by comparing the values with the normal values and alerts the user accordingly.

11.Sensors

Sensors collect data and those data details will be stored in system database for computation process. It involves a group of three sensors

12.Vital sensor:

This class records the vitals of the patient and detects the abnormal levels of vitals.

13.Vehicle sensor:

This class contains the info of the vehicle i.e whether vehicle is in on state or off state.

14.Pollution sensor

This class includes the pollution level of the outside environment and detection of alarming pollution level.

15.Doctor

It is a class that includes Field, designation, name and email id details of the doctor. He/she can accepts appointments make appointments senior doctor.

16.Appoinments

On the patient side, they can make a request of appointments with their desired doctors. And on the doctor's side they can also make appointments for to connect with senior doctor for getting any required consultation.The appointment date and time and doctor name will be recorded and also it includes boolean condition of whether the date of appointment is near or not.

17.Pharmacy

Pharmacy maintains details like name, address in the system, so that patients can get the services efficiently.

18.Medicine catalogue

It is a class which contains serial number of each and every medicine the respective pharmacy has.

19.Medicine Specification

Medicine price and it's description like chemical compound which would be it made from and it's manufacturing details will be contained

20.Item

It is a class which contains medicine's id,its quantity and price and this item can be added to the cart by the patient or the pharmacy.

21.Cart : This class has cart id, no of items, as its attribute. Patient can add/delete or update or checkout the cart.

22.Order

This class contains the status of order,order id and order date and billing address as its attribute and patient orders details are maintained through this class.

23.Prescription

It is a class that contains Prescription id and doctor name. And these prescriptions could be uploaded by patients into the system.

24.Camera

By giving access to the camera the the prescription can be captured by the camera.

25.Medicine bill

The bill number and total amount will be displayed for the patient once the order is confirmed and payment is made.

26.Payments

The mode of transaction, date and time of payments will be recorded.This is vital for order to be confirmed

27.Patient

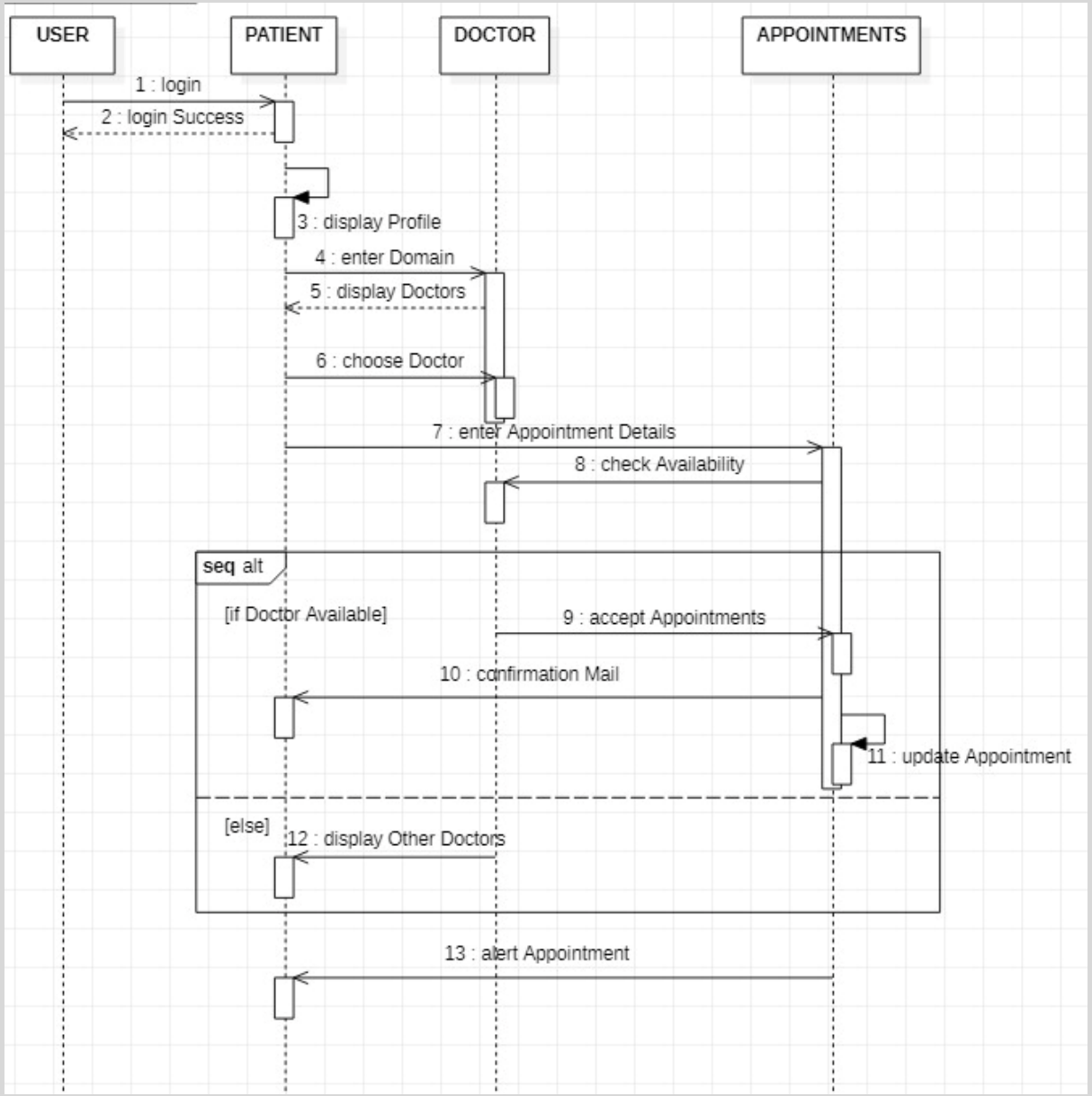
It is identified as class and has functionalities namely patients can book appointments, pay medicine bill, upload prescription

28.User

This class has User id and password as attributes, so that the users can log into system by giving valid login details and the details will be verified.

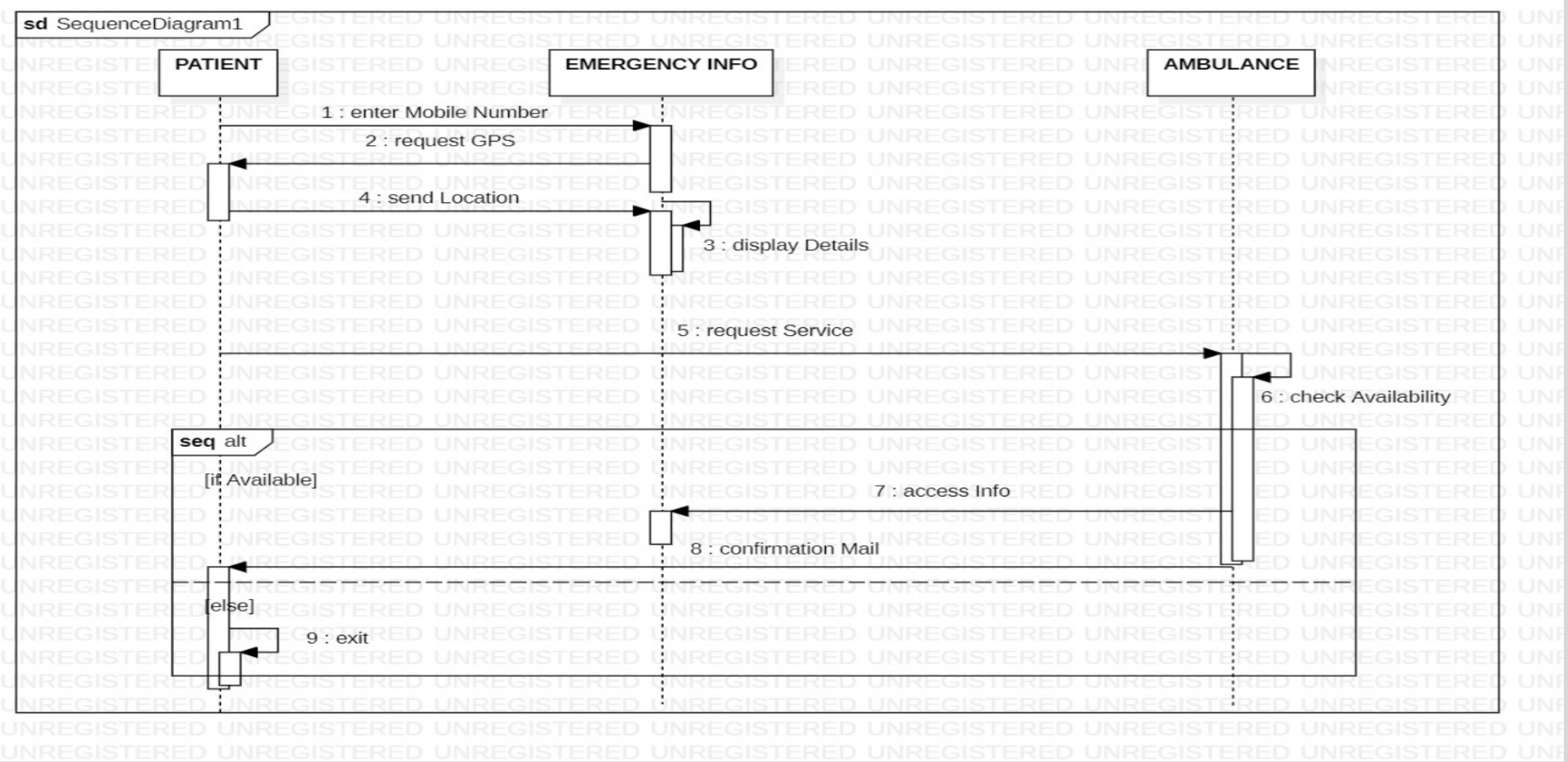
SEQUENCE DIAGRAM

1.Book appointments



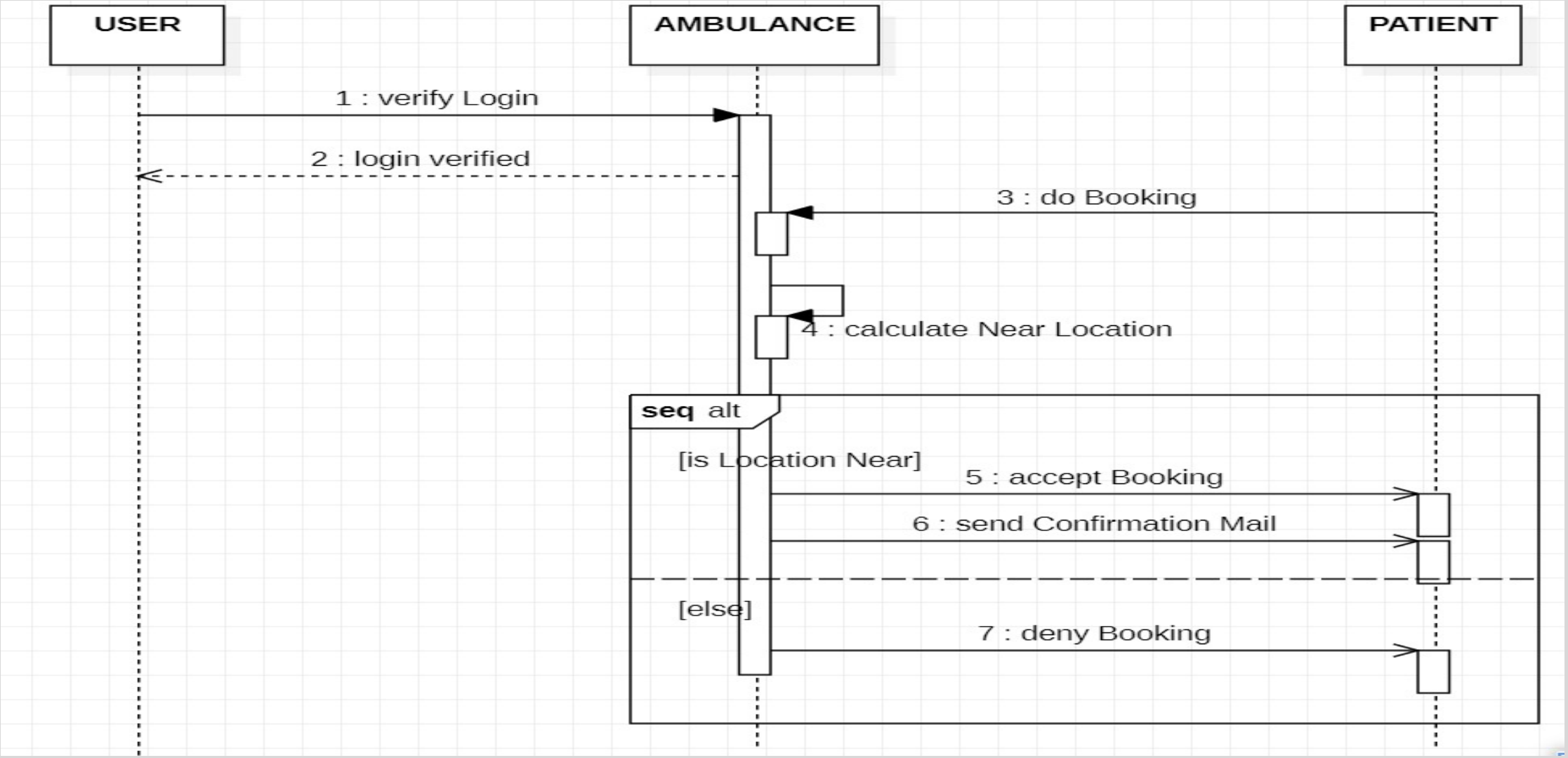
- The user enters his credentials and logs in as patient which is verified. The profile is displayed to the patient.
- The patient enters the domain of the doctor to book appointment . The application checks for the domain in the doctors profile and if the string matches with the patient's search it displays the doctors.
- The patient now picks up the doctor and gives the details such as date and time for booking.If the doctor is available at the patient required time the appointment is accepted or else denied.
- Once the appointment is accepted confirmation mail is send to the user and appointments list of the user is updated with the new appointment.
- Once the appointment date and time is reached the app notifies the user.

2.Contact ambulance



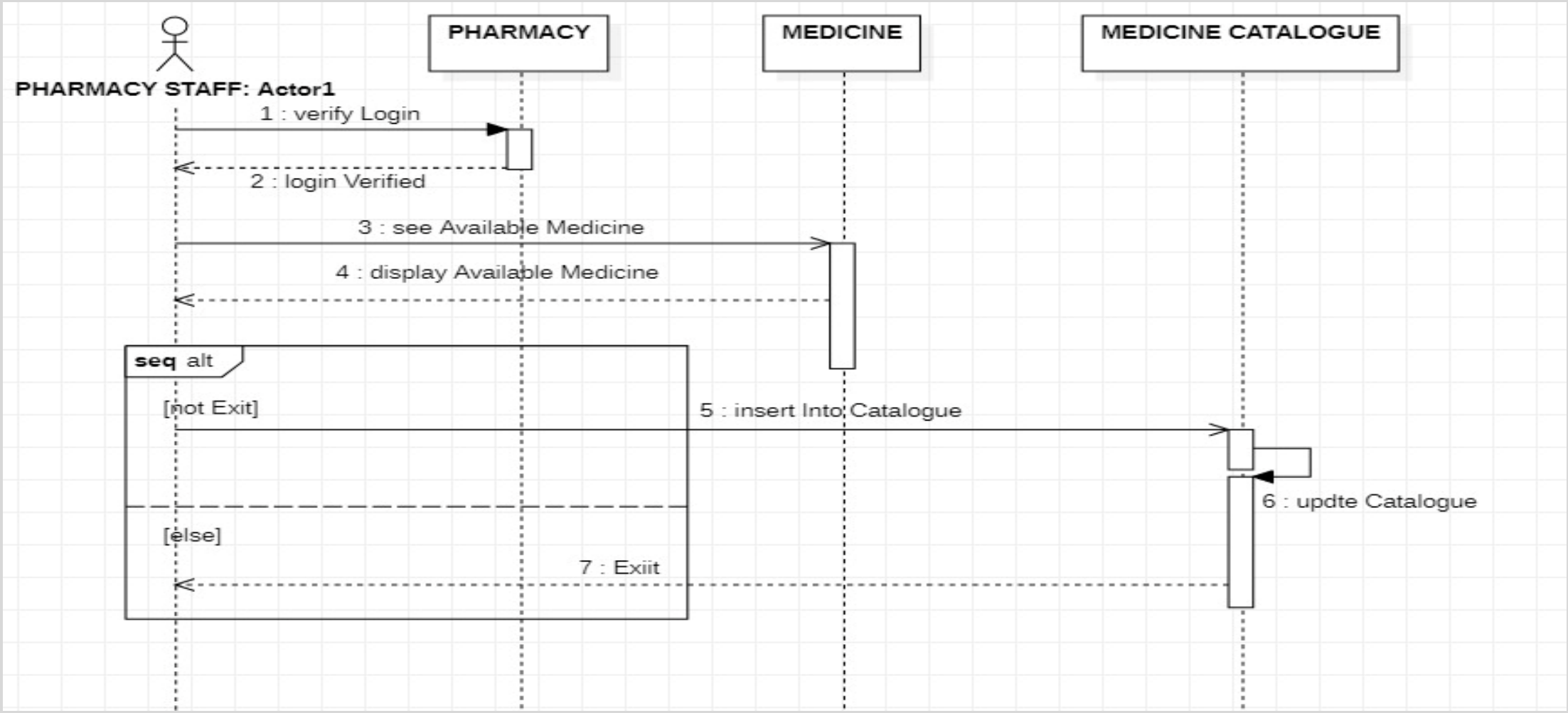
- The emergency details of the patient such as mobile number and access to GPS is got .
- Now the GPS is turned on in the user's device and location is read and is sent and stired in the emergency info object.
- Once user makes a request for ambulance service the availability and consent of the ambulance is received and suitable action is taken
- If booking is confirmed ,a confirmation mail is sent to the user.

3.Accept/deny bookings



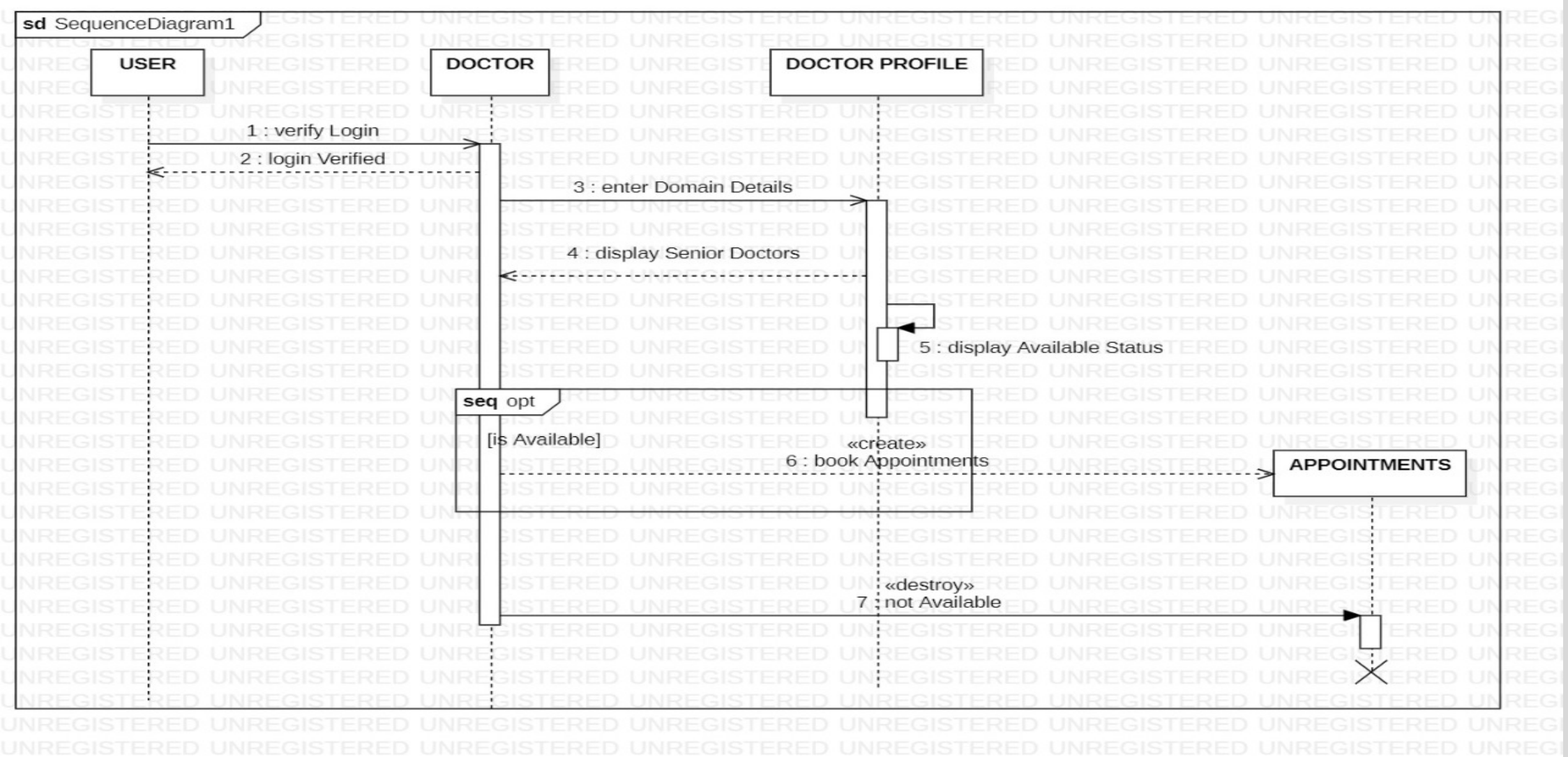
- Once the patient makes a request for ambulance service the following sequence of events takes place
- The ambulance service provider logs in and his login is verified.Now based on the location of the service demander the nearby ambulance service is searched and request is made by the app.
- If the ambulance service is willing to take bookings, the booking is confirmed and a confirmation mail is send to the user.
- Else the app searches for other booking.

4.Add medicine



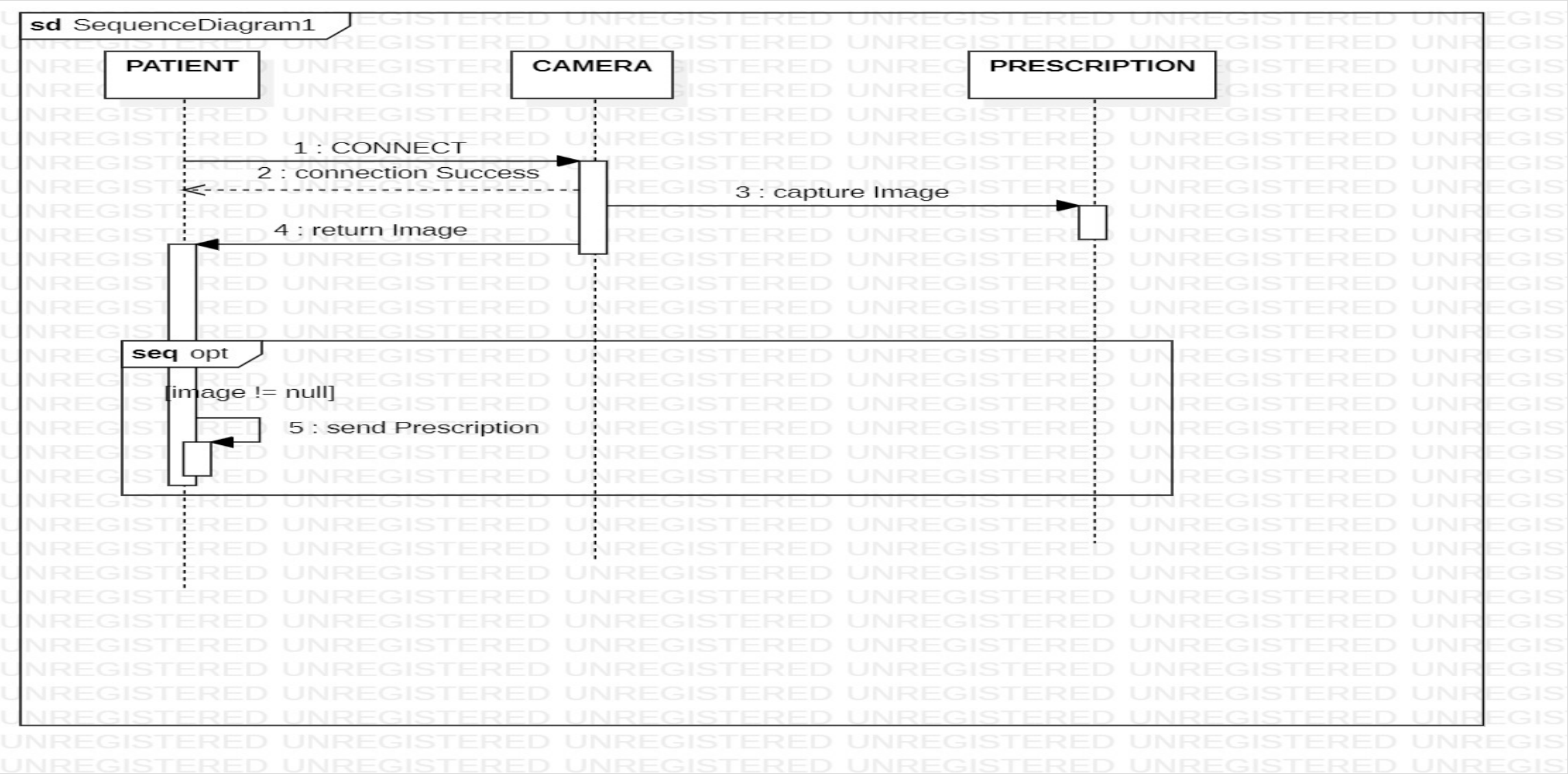
- The pharmacy staff logs in and his/her login is verified by the application.Now the list of available medicines is shown to the user.
- If the user wants to add new medicine which is not in list,he adds them into catalogue and the catalogue is updated with this new medicine.
- If the medicine already exists in the catalogue the the process exits.

5.Connect senior



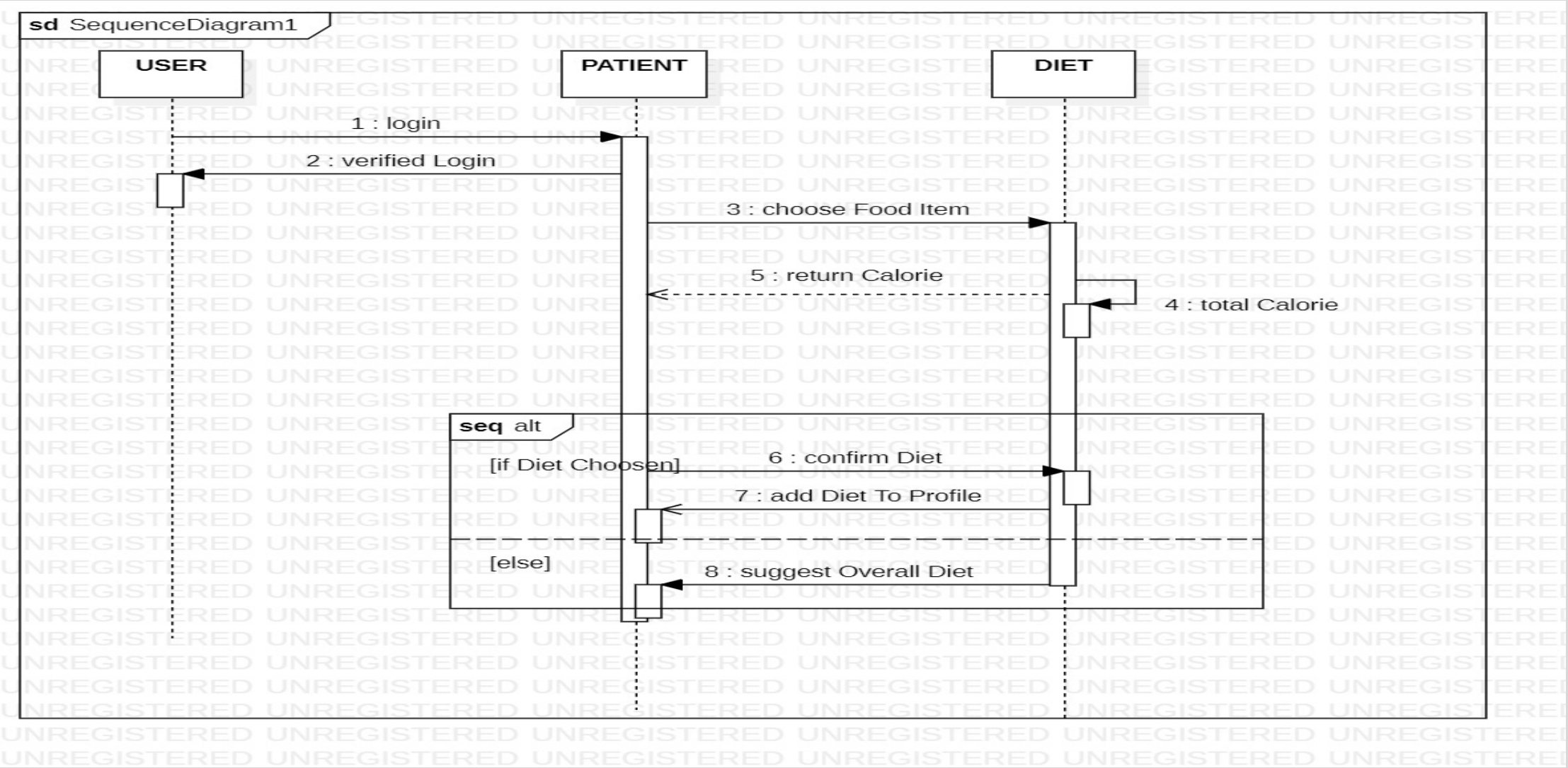
- The doctor logs in and his/her login is verified by the application.
- The doctor gives his domain details such as his field and designation.
- The application now looks for doctors with and same field and designation as senior.If match is found is suggests the senior to the doctor
- If the senior doctor is available on the specified date and time appointments are take in else the events are aborted.

6.Upload prescription



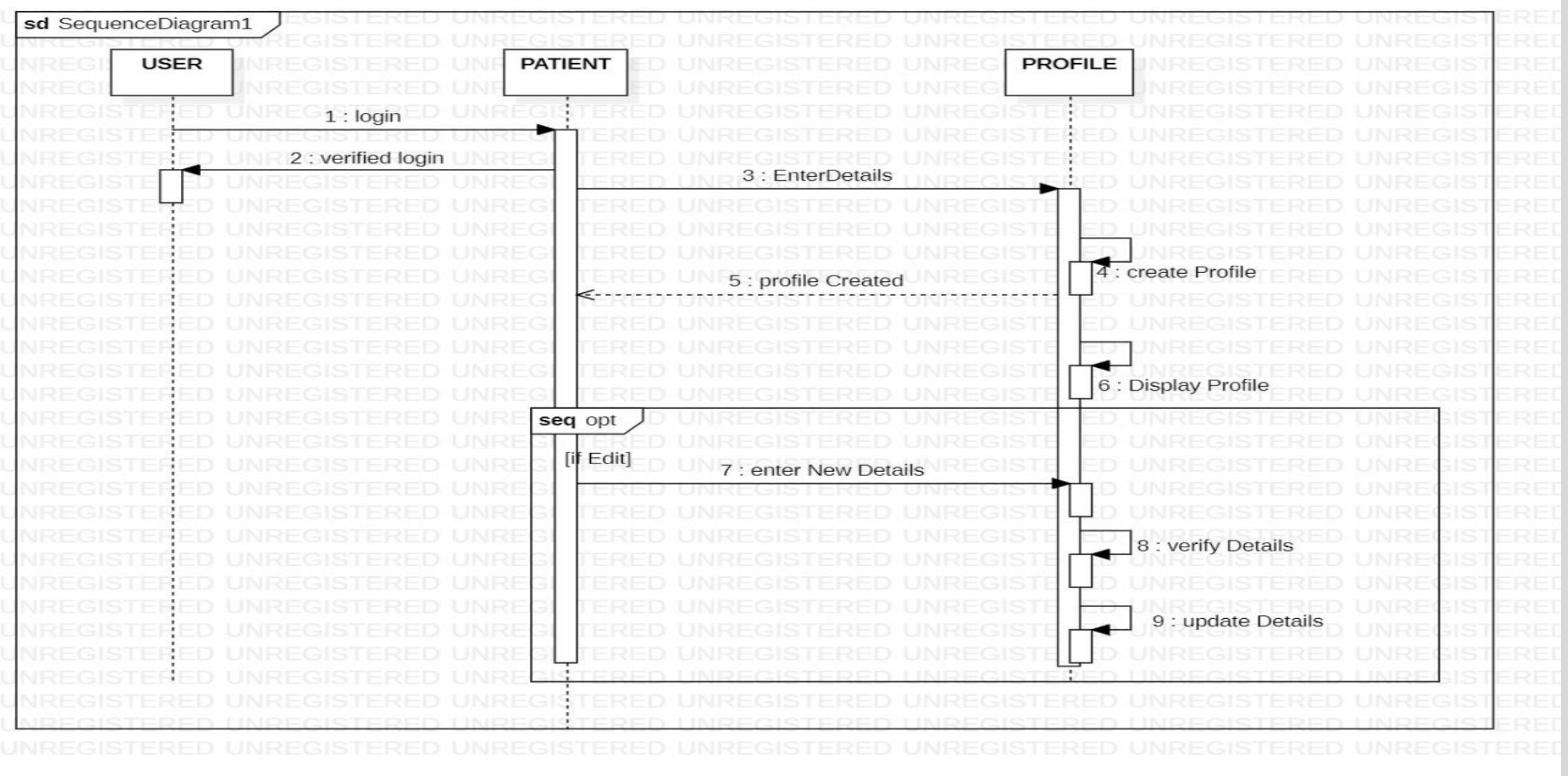
- Once the patient gives permission to access the camera the camera is connected and connection status of the camera is returned to the user.
- If the connection is success the image of the prescription is captured.
- The image captured is returned to the user and is uploaded for ordering medicine.

7.Choose diet



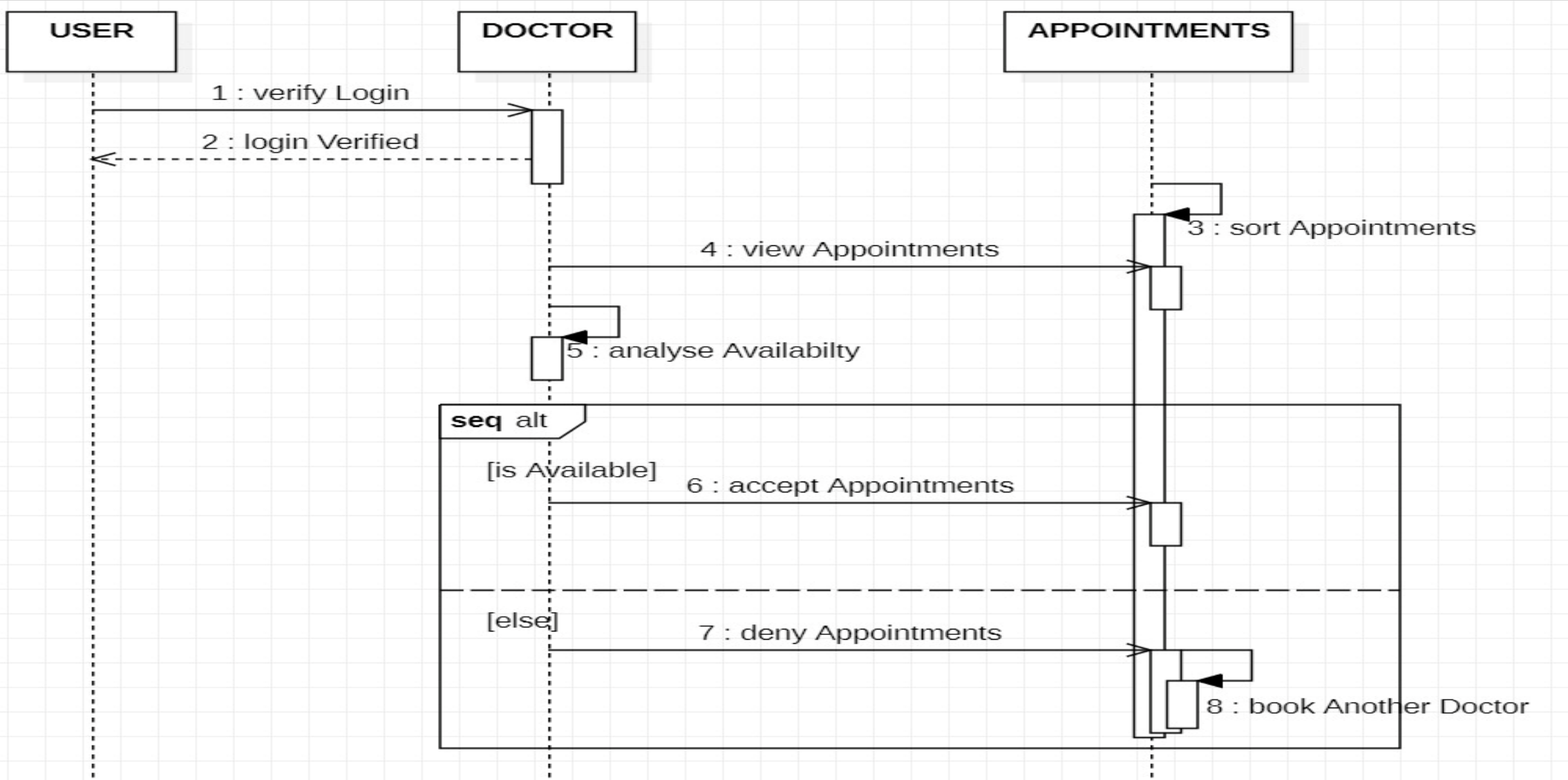
- The user if he wishes to pick the food items by himself,chooses the food items available in the database
- Once food items are picked,total calories for a day is calculated and is returned to the diet.
- If the user is fine the diet is confirmed and the diet plan is added to the user's profile
- Else, the system based on the health condition of the patient of the patient suggests the overall diet andf if the user wishes can take up the diet plan.

8.Update profile



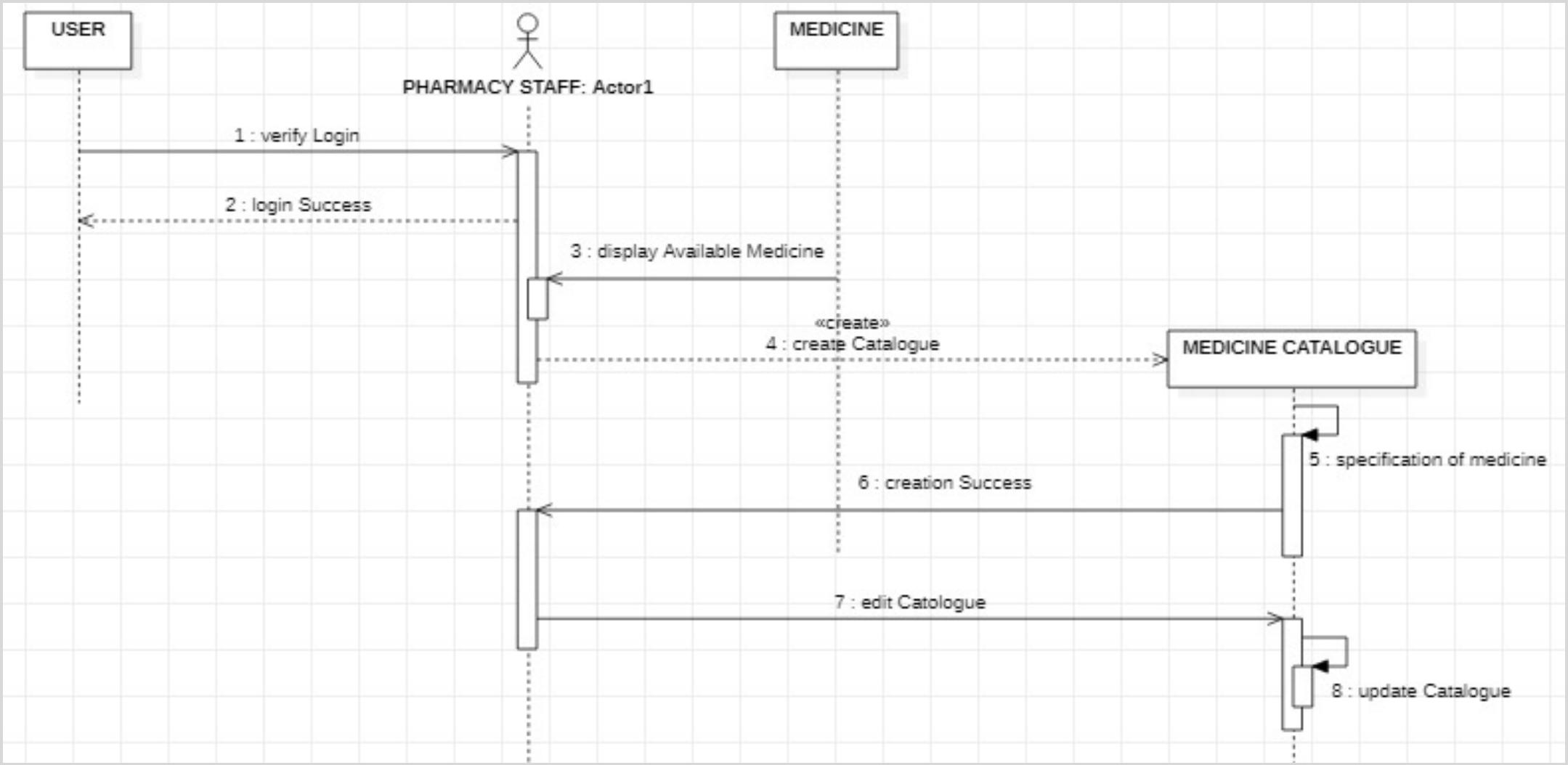
- The user logs in and his/her login is verified by the application.Now the user enters the details such as name,age,gender and address and
- Now the profile is created and a unique profile id is assigned to the user which will be be made available for display whenever the user wants.
- If the user wants to edit the already existing profile,he can choose the respective field and can enter the new details and the details after verification will be updated in the database.

9.Accept/deny appointments



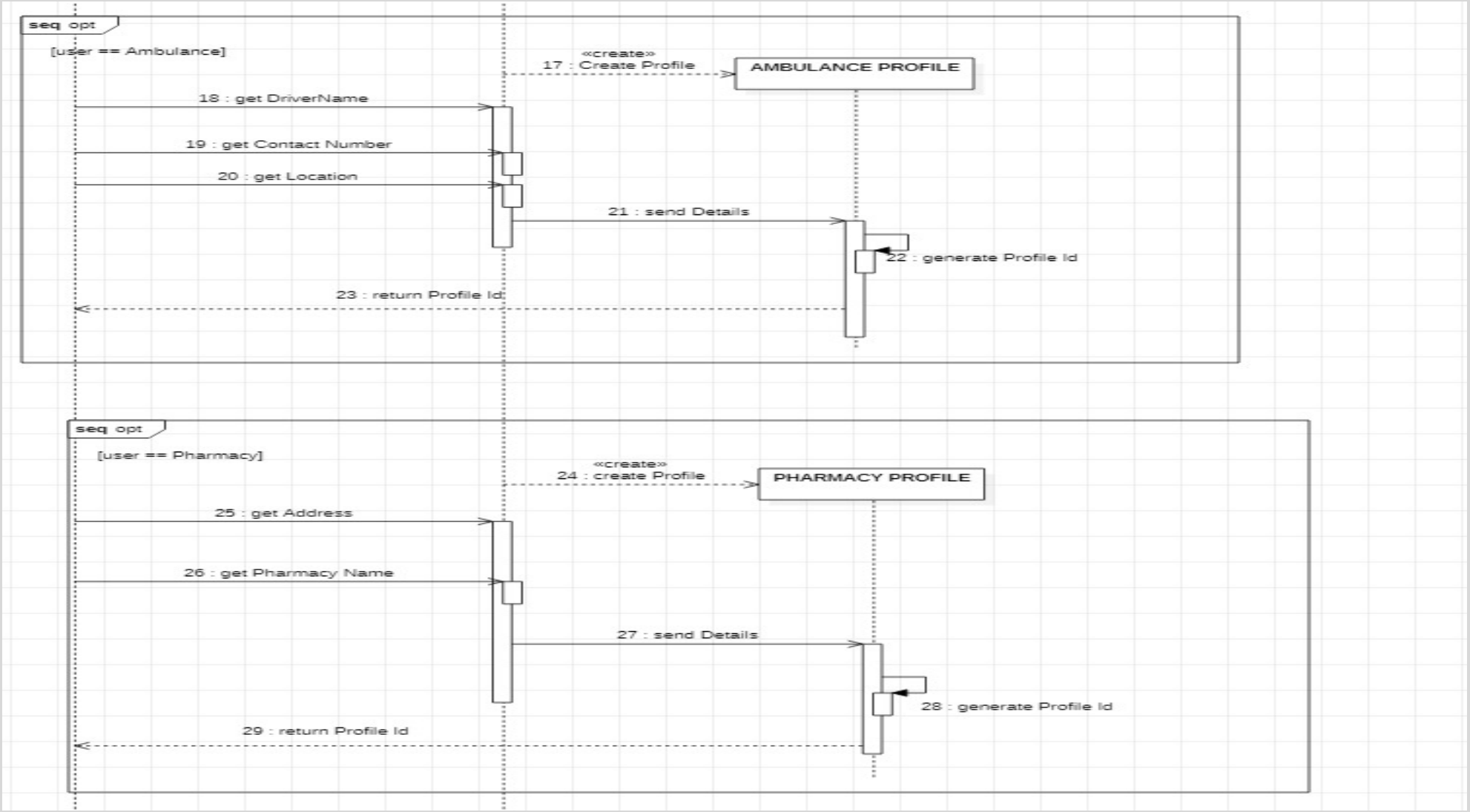
- The doctor logs in and his/her login is verified by the application.
- The appointments which the doctor already has is sorted and displayed.
- The availability of the doctor is analysed against the patient requested date and time
- If the doctor is available and he wishes to take up the appointment,the appointment is booked.
- Else,the appointment is denied and some other available doctor is suggested to the patient.

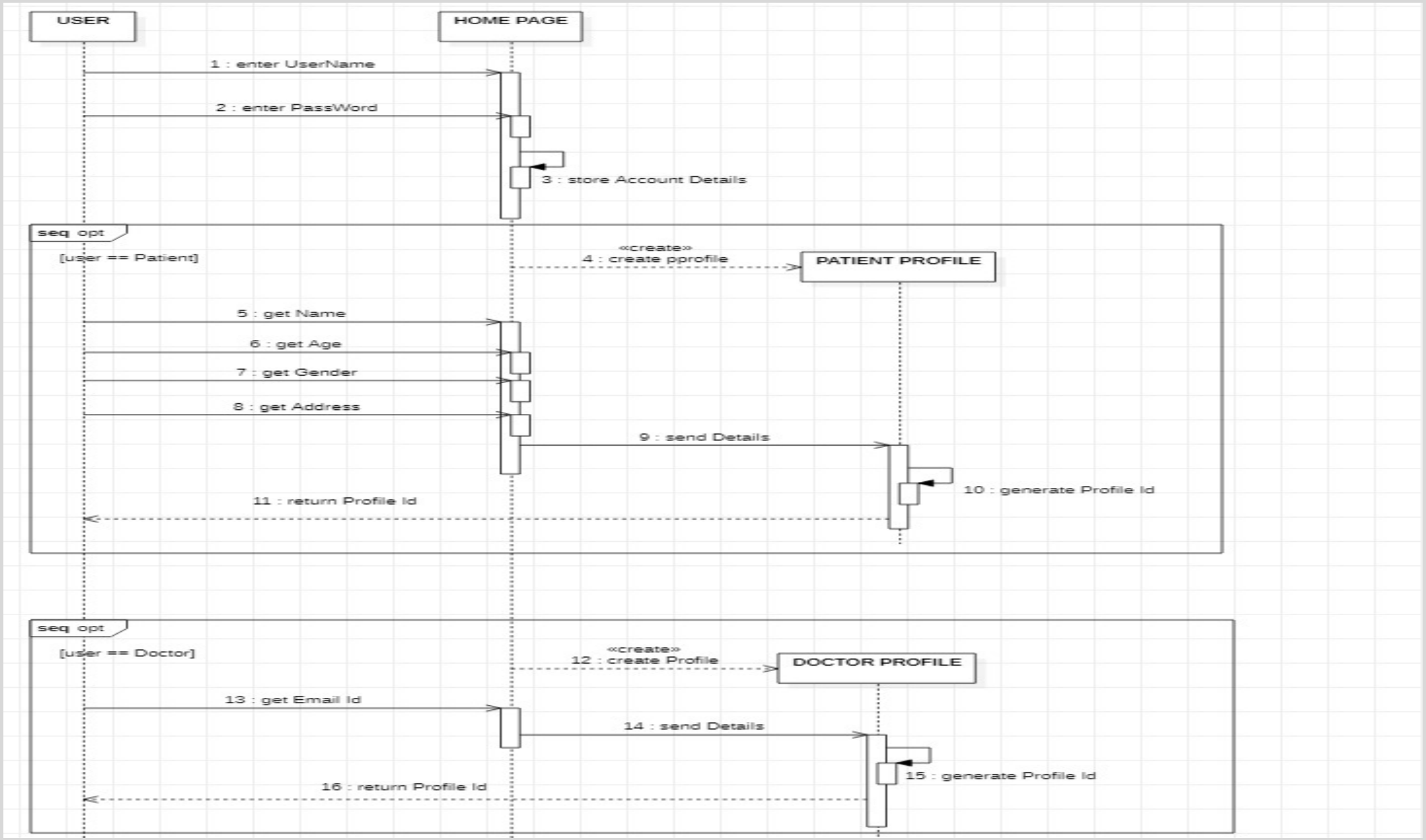
10.Create catalogue



- The pharmacy staff logs in and his/her login is verified by the application.
- The available list of medicine is displayed to pharmacy and a catalogue is created specific for a particular pharmacy.
- The medicine specification which includes description and unit price for each and every medicine in the catalogue is added and is stored in the pharmacy database.
- Or else if the catalogue is already created,the pharmacy person can edit the catalogue which will be automatically updated.

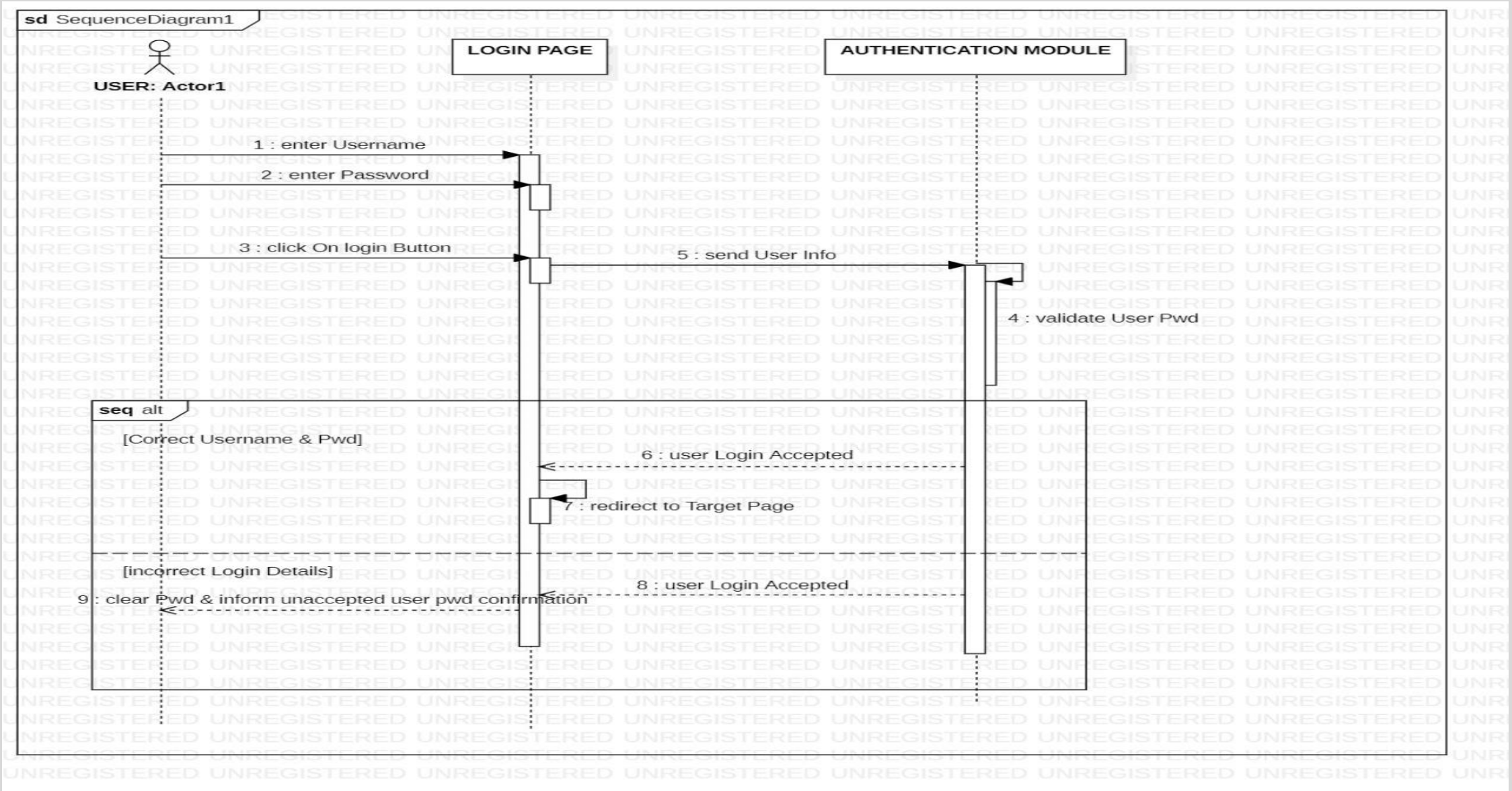
11.Register


seq opt
[user == Ambulance]
«create»
17 : Create Profile
AMBULANCE PROFILE
18 : get DriverName
19 : get Contact Number
20 : get Location
21 : send Details
22 : generate Profile Id
23 : return Profile Id
seq opt
[user == Pharmacy]
«create»
24 : create Profile
PHARMACY PROFILE
25 : get Address
26 : get Pharmacy Name
27 : send Details
28 : generate Profile Id
29 : return Profile Id


USER
HOME PAGE
1 : enter UserName
2 : enter PassWord
3 : store Account Details
seq opt
[user == Patient]
«create»
4 : create pprofile
PATIENT PROFILE
5 : get Name
6 : get Age
7 : get Gender
8 : get Address
9 : send Details
10 : generate Profile Id
11 : return Profile Id
seq opt
[user == Doctor]
«create»
12 : create Profile
DOCTOR PROFILE
13 : get Email Id
14 : send Details
15 : generate Profile Id
16 : return Profile Id

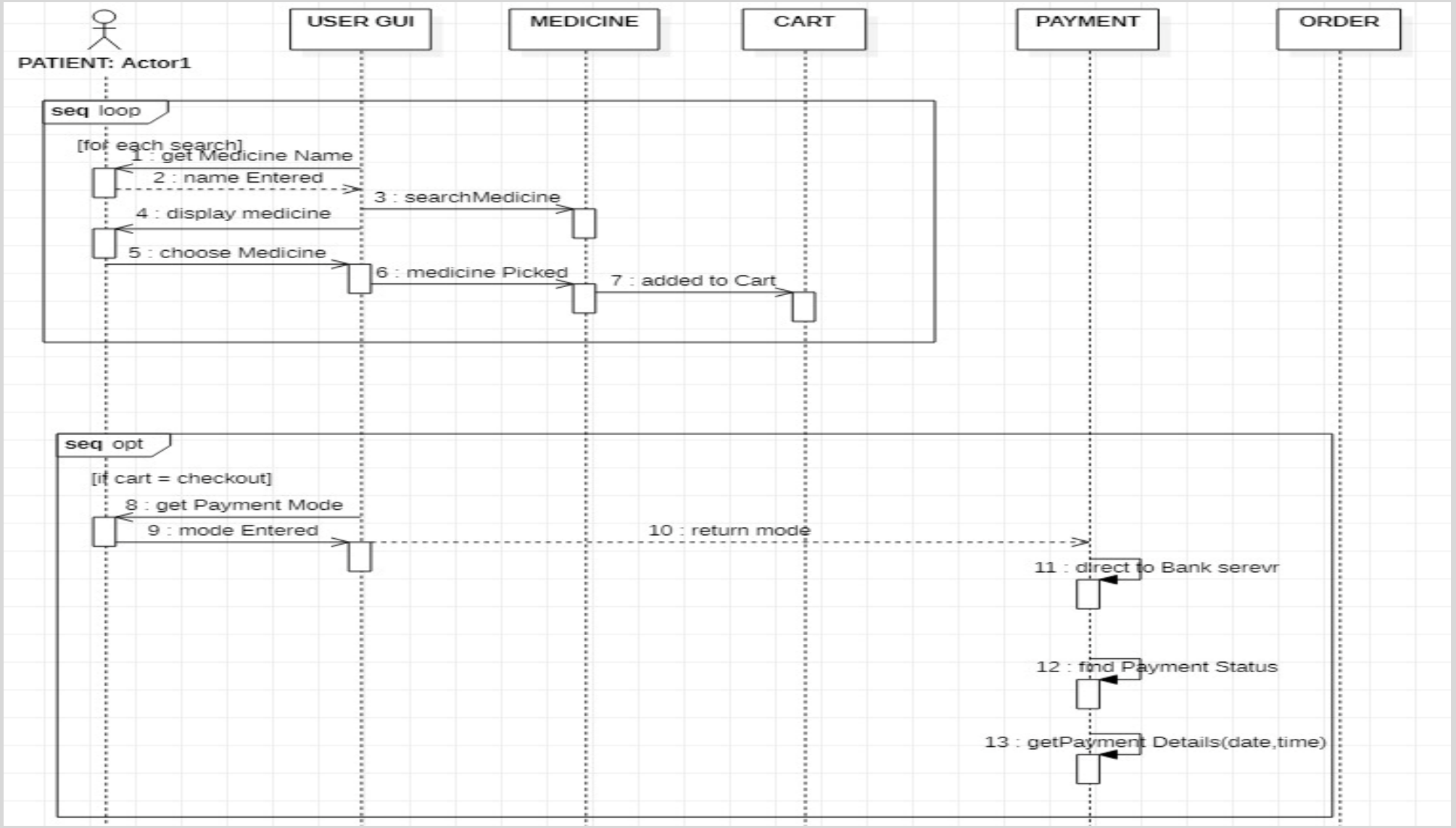
- The User logged into system and then Account details will be stored
- After that Profile will be created , if the user logged as Patient then the appropriate details will be got like name ,age, gender and address have been got ,stored those details and profile Id will generated
- then the generated profile Id return to the user
- If the user logged as Doctor then the Email id will be got and stored and profile Id will generated and it is return to the user
- If the user logged as Ambulance service, then profile creation have done by getting details like get driver name, contact number and location
- After receiving those details then Profile Id will be generated and it have returned to the user
- If the user logged as Pharmacy staff, then the profile will be created by getting details like Address and Pharmacy name
- After that profile Id will generated and return to the user

12.Authenticate user



- The user have done the login process by entering Username and password
- After getting logi details , validation of password have done
- If the username and password is correct so returns message as Login accepted and redirect to target page
- If the username and password is incorrect so returns message as login Unaccepted and asking the user to enter correct login details

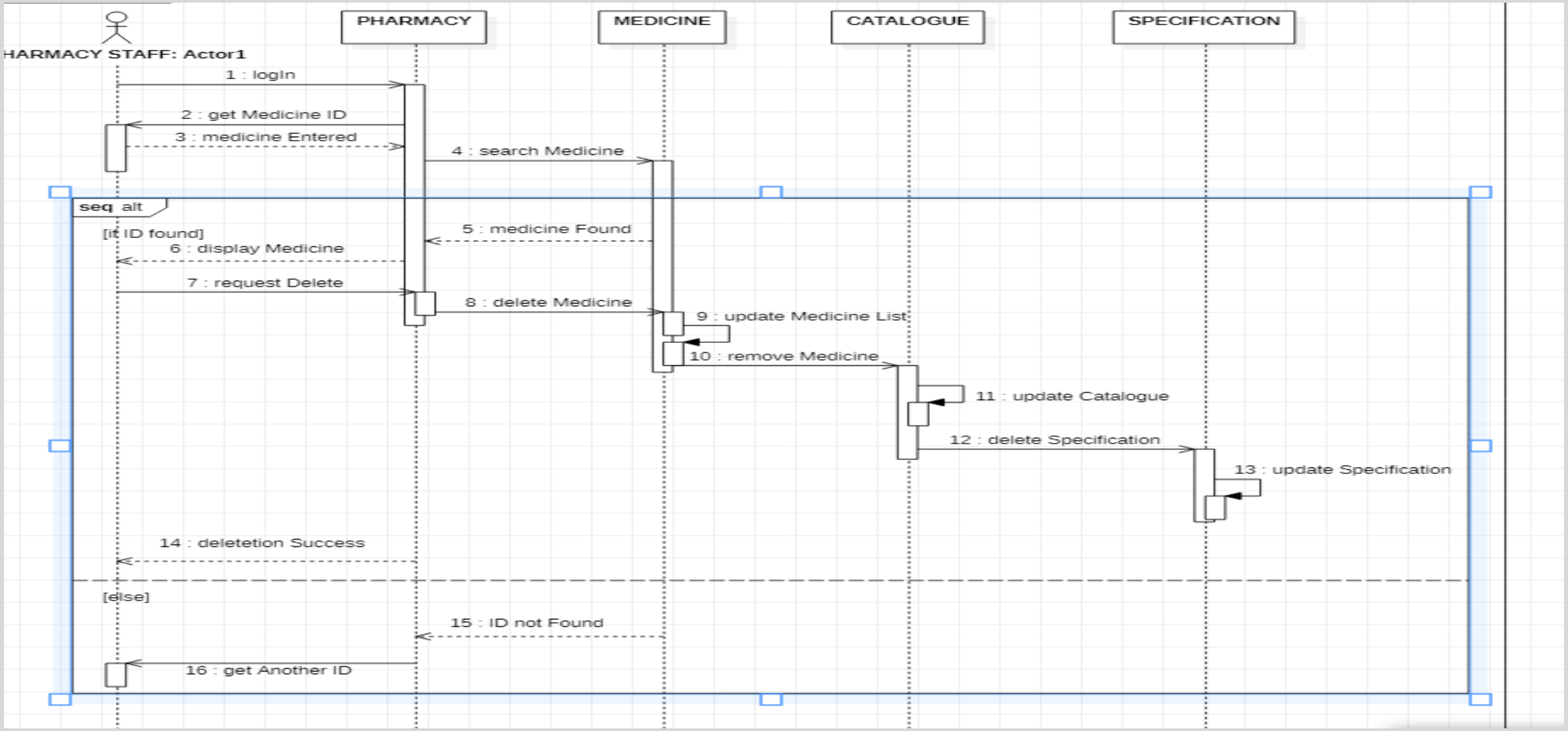
13.Order medicine




seq opt
14 : place Order
15 : success Message
16 : order Status
17 : ordre confirmed

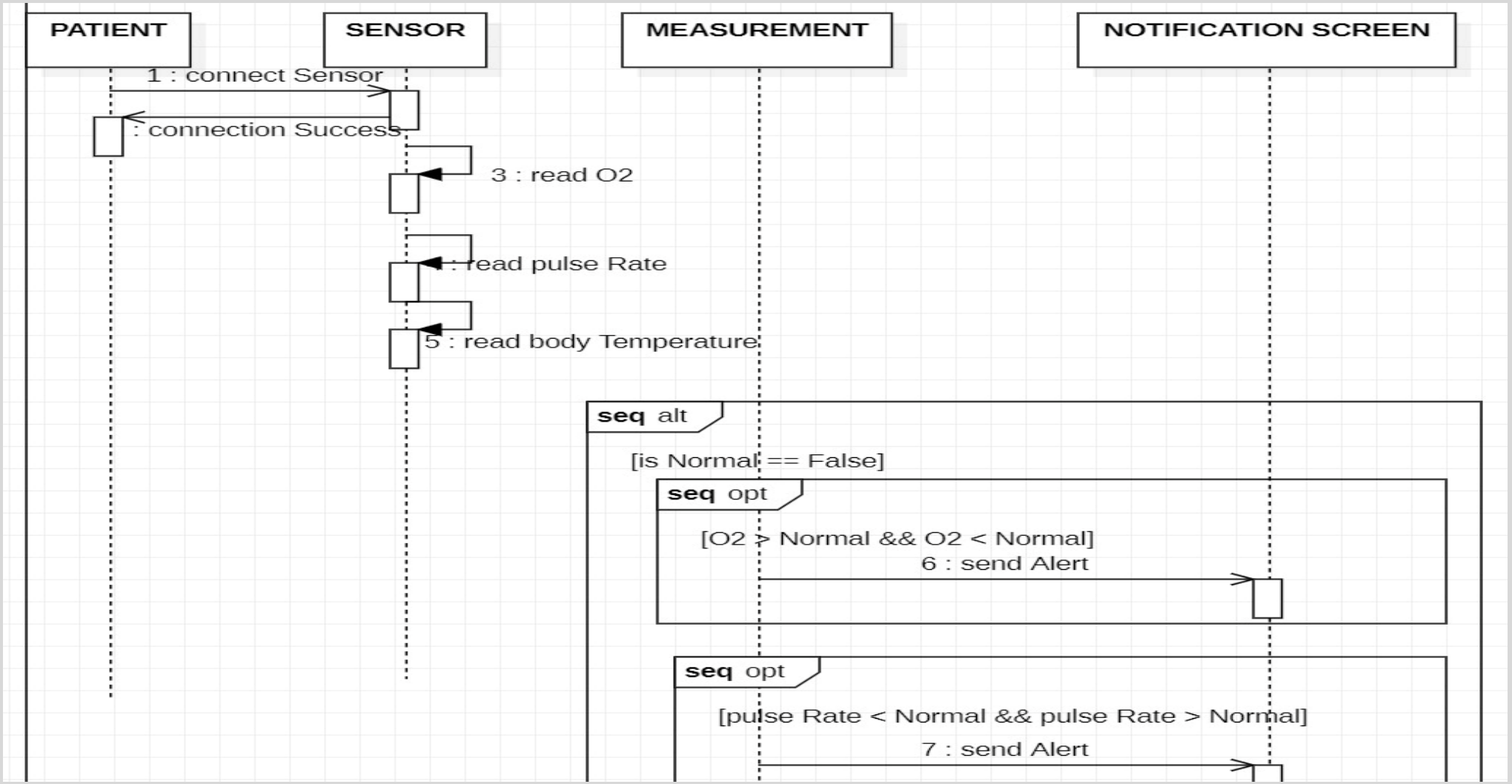
- User asked to enter the medicine name they want to purchase
- After the searching of particular medicine have done , if the medicine is found it is displayed to user
- so user added those medicine to the cart
- the above mentioned process have to be run for each and every medicine purchase
- if cart undergo checkout process then the user asked to enter payment mode so that entered mode will returned to payment
- after completion of payment process the order will be placed
- Finally the order status will be identified and confirmation of order send to user

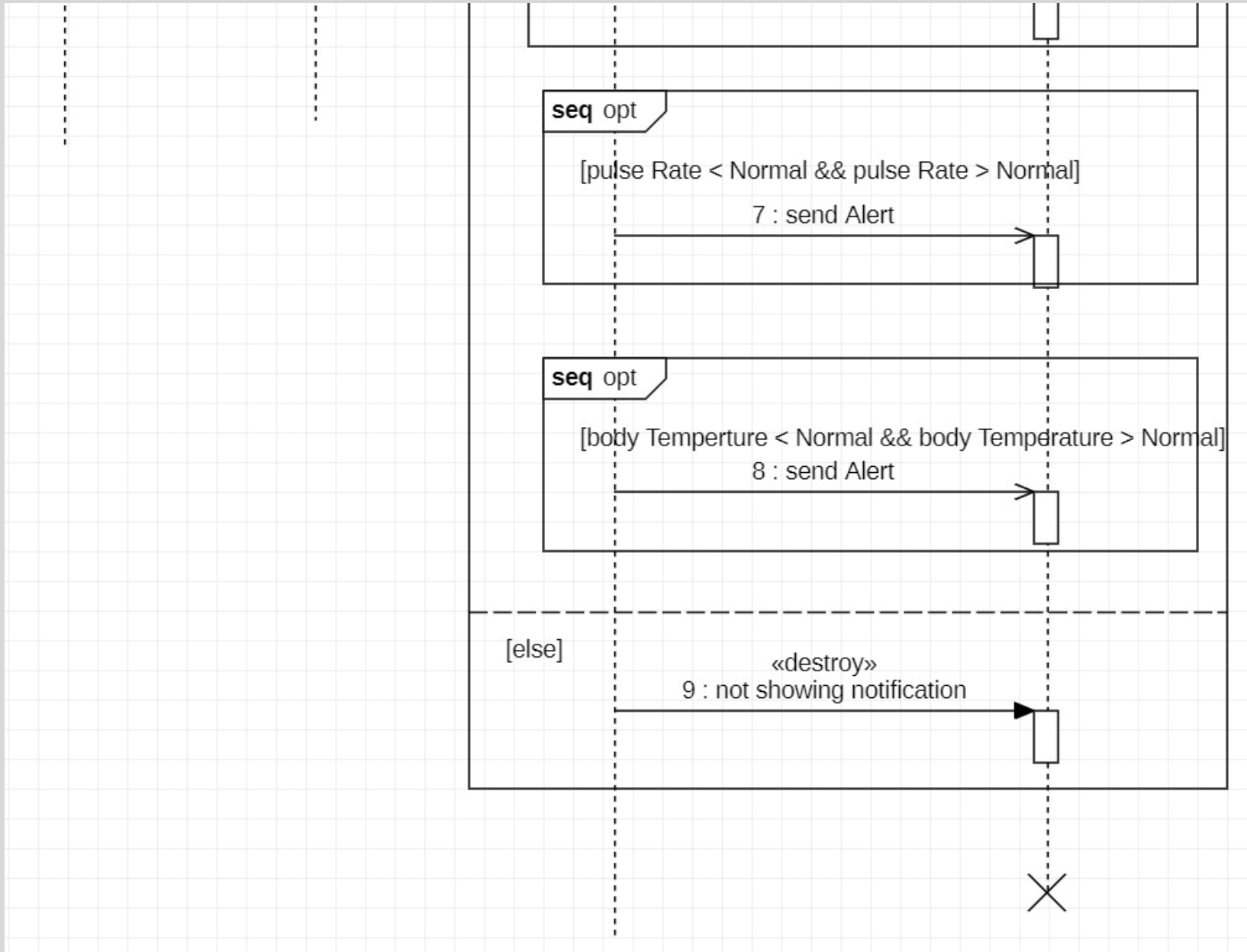
14.Remove medicine



- The Pharmacy staff login into system and asked to enter medicine Id want to delete
- After getting medicine id the searching of medicine have done
- if medicine is found the deletion of medicine from catalogue and specification have done
- After the deletion catalogue and medicine specification will be updated and finally deletion success message will be send to Pharmacy staff
- If the medicine not found so it returns medicine not found message to Pharmacy staff and prompts for another Id

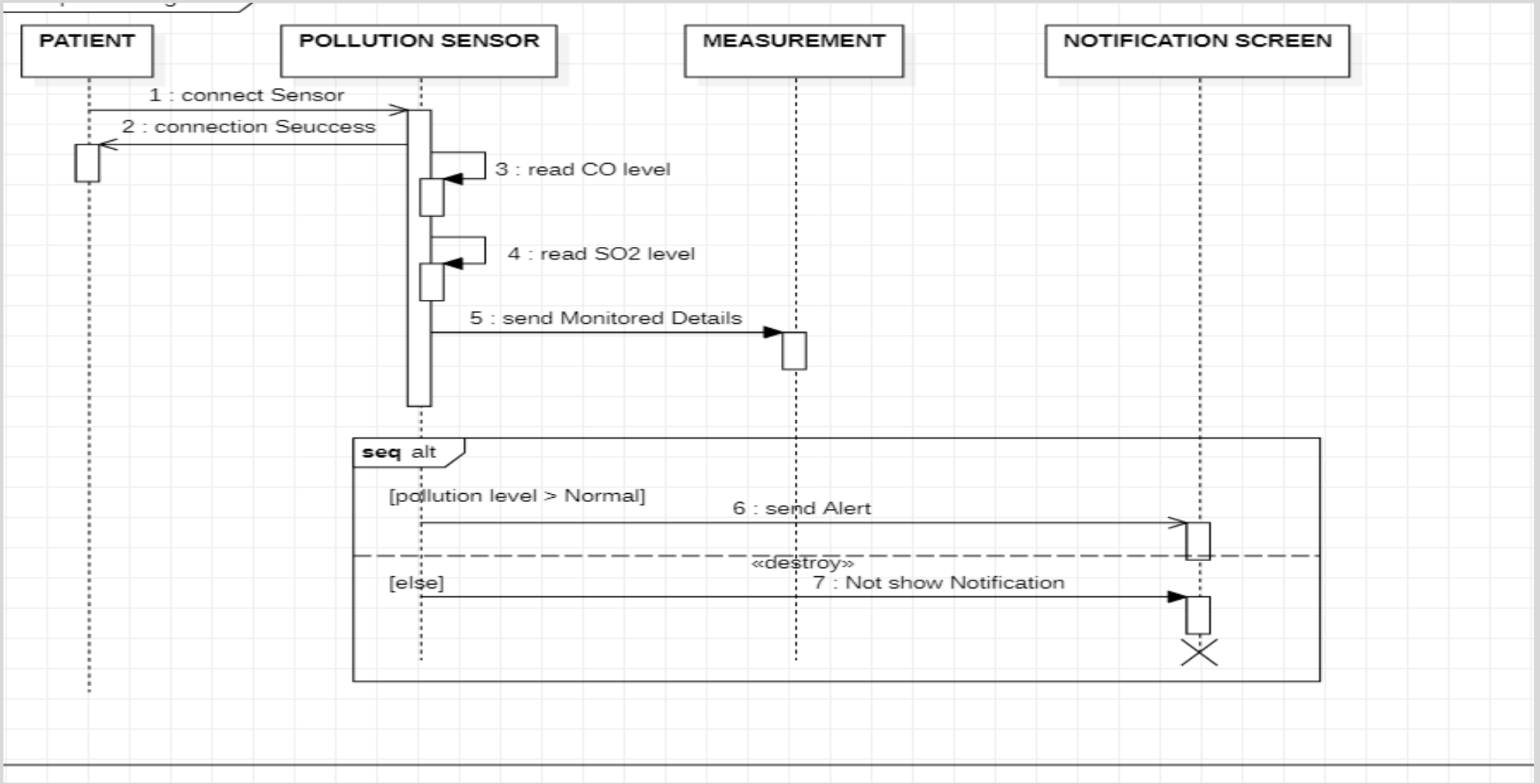
15.Record vitals




seq opt
[pulse Rate < Normal && pulse Rate > Normal]
7 : send Alert
seq opt
[body Temperture < Normal && body Temperature > Normal]
8 : send Alert
[else]
«destroy»
9 : not showing notification

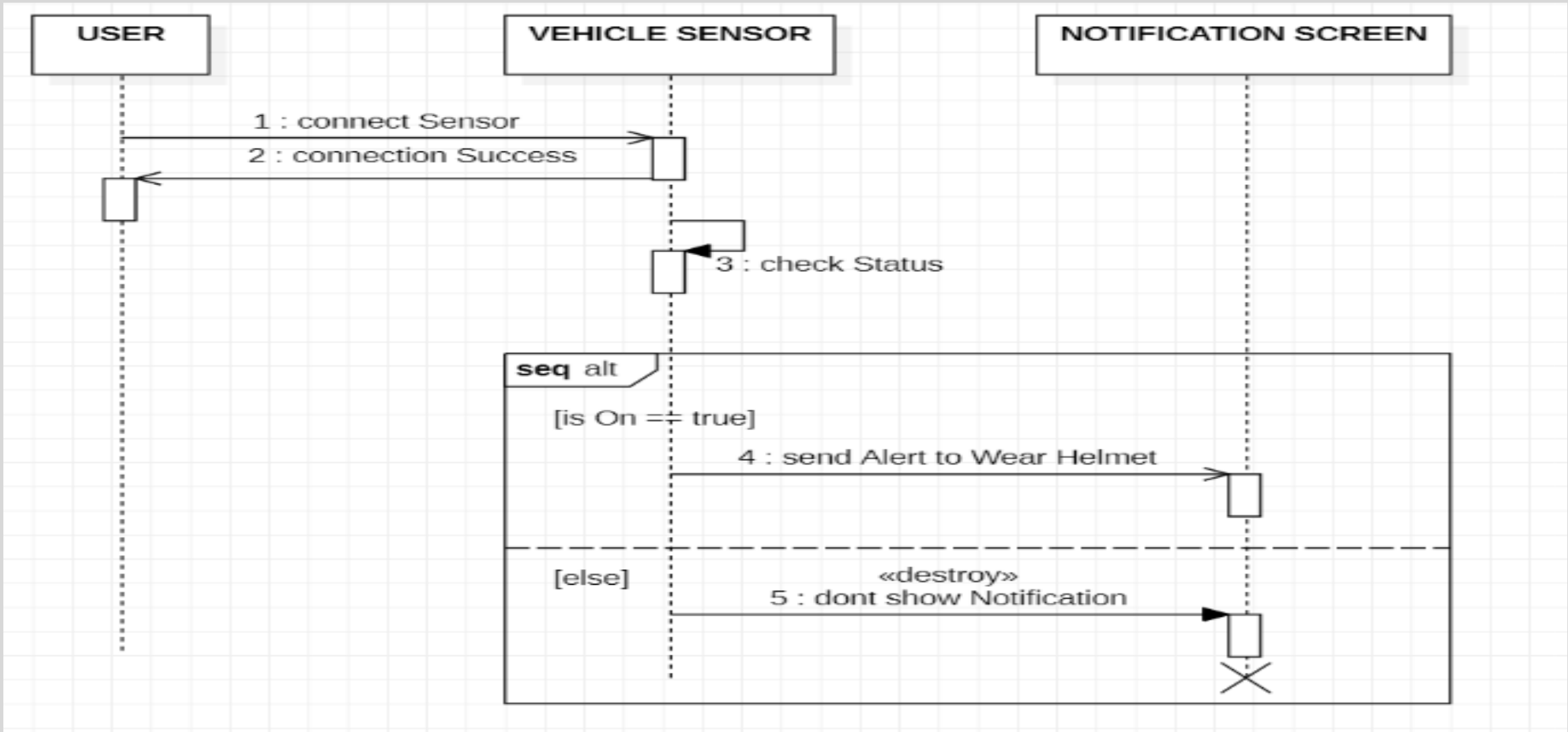
- The user connects to sensor
- The vital sensor reads data like o2 level , body temperature and pulse rate
- If the gathered data is above the normal then the notification message send to user

16.Manage pollution



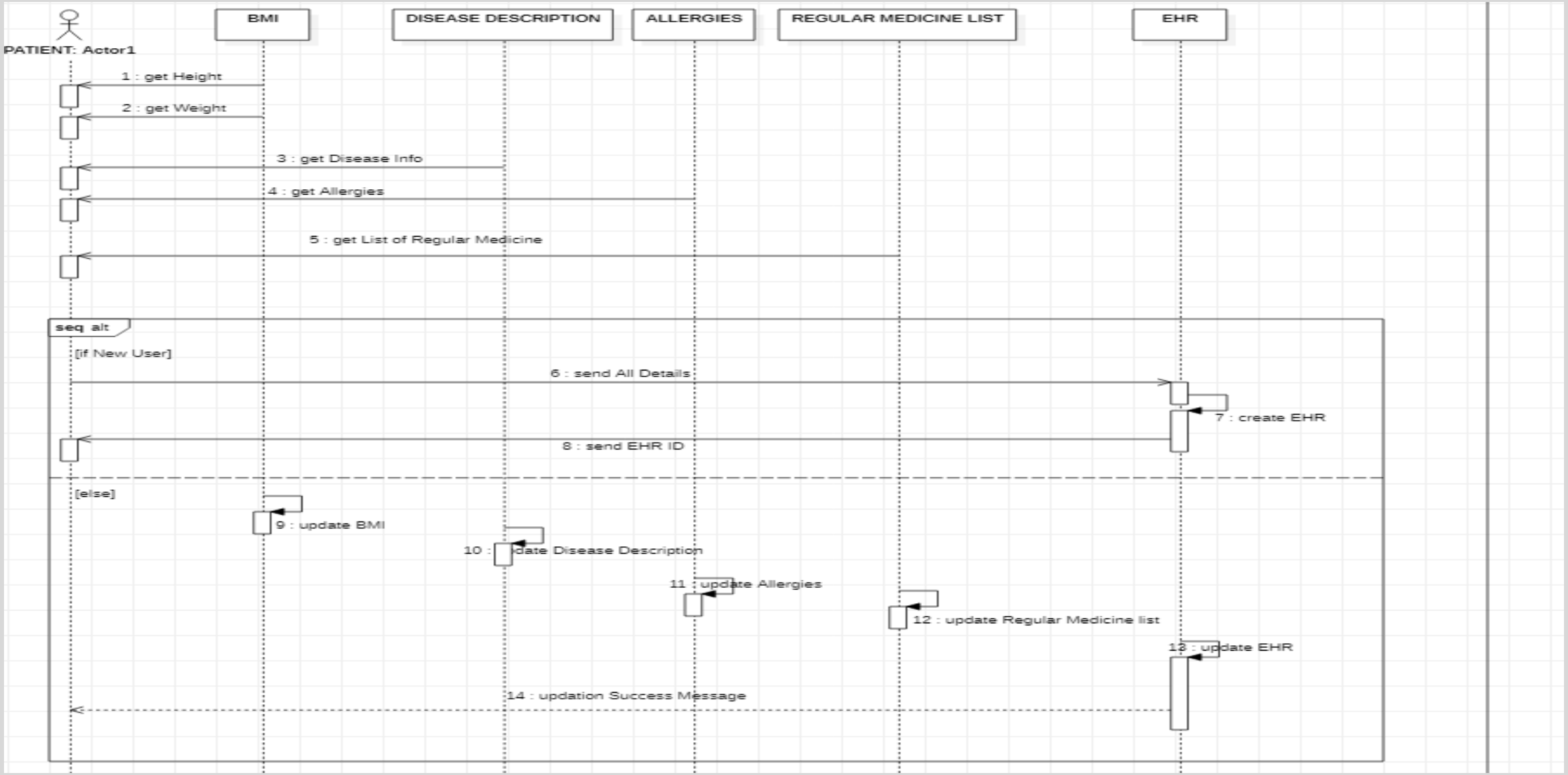
- The user connects to sensor
- The Pollution sensor reads data like CO level and So2 level
- If the gathered data is above the normal then the notification message send to user

17.Track vehicle



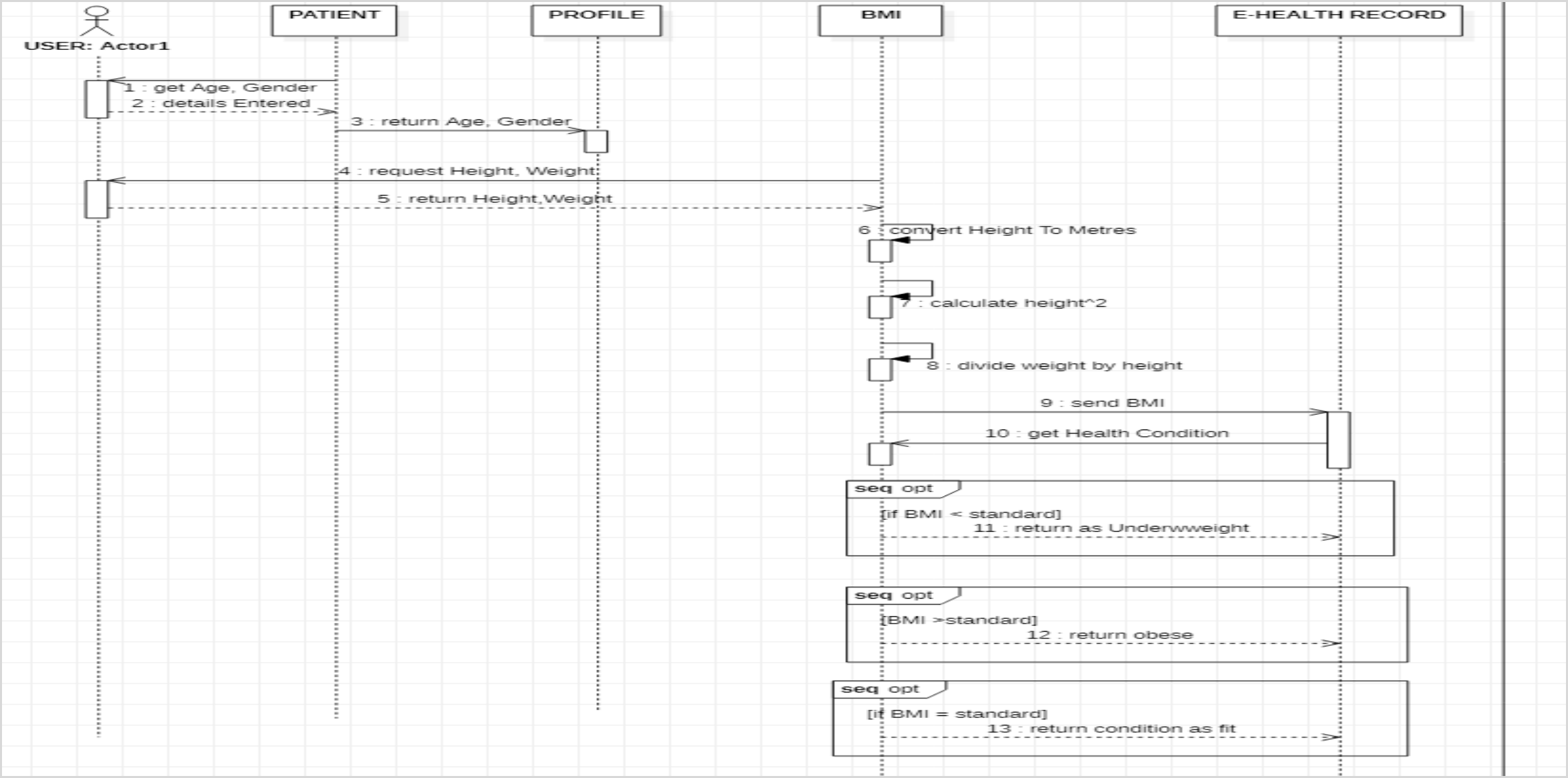
- The user connects to sensor
- The Vehicle sensor detects if the Vehicle is on or not
- If vehicle is on then the notification message send to user

18.Update e-health record



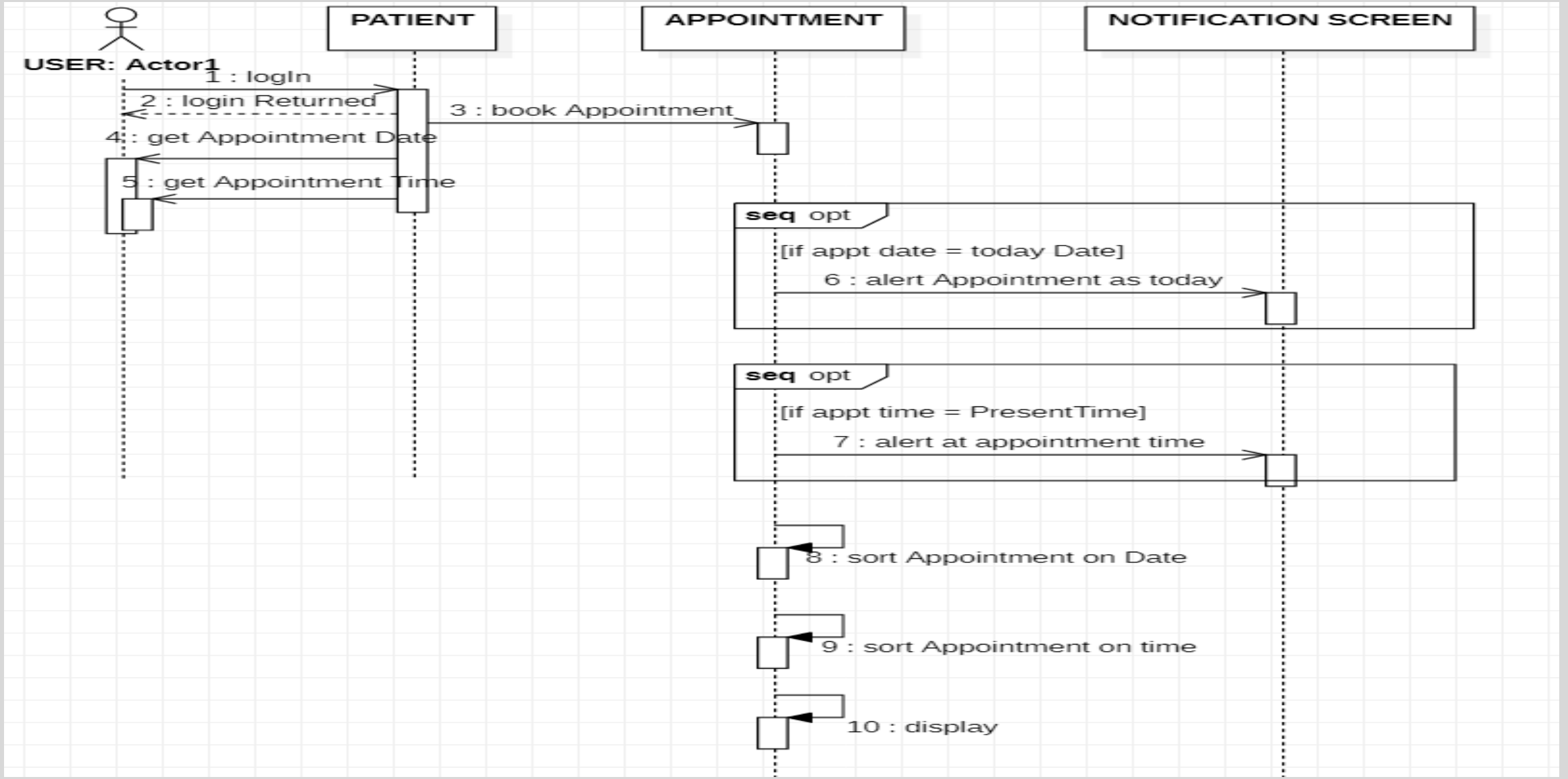
- The user have to enter height weight , regular medicine and llergies they got
- if the user is new then creation of E-Health record have done by getting those info and return E-HR Id
- Else update E-Health record of a user

19.Compute BMI



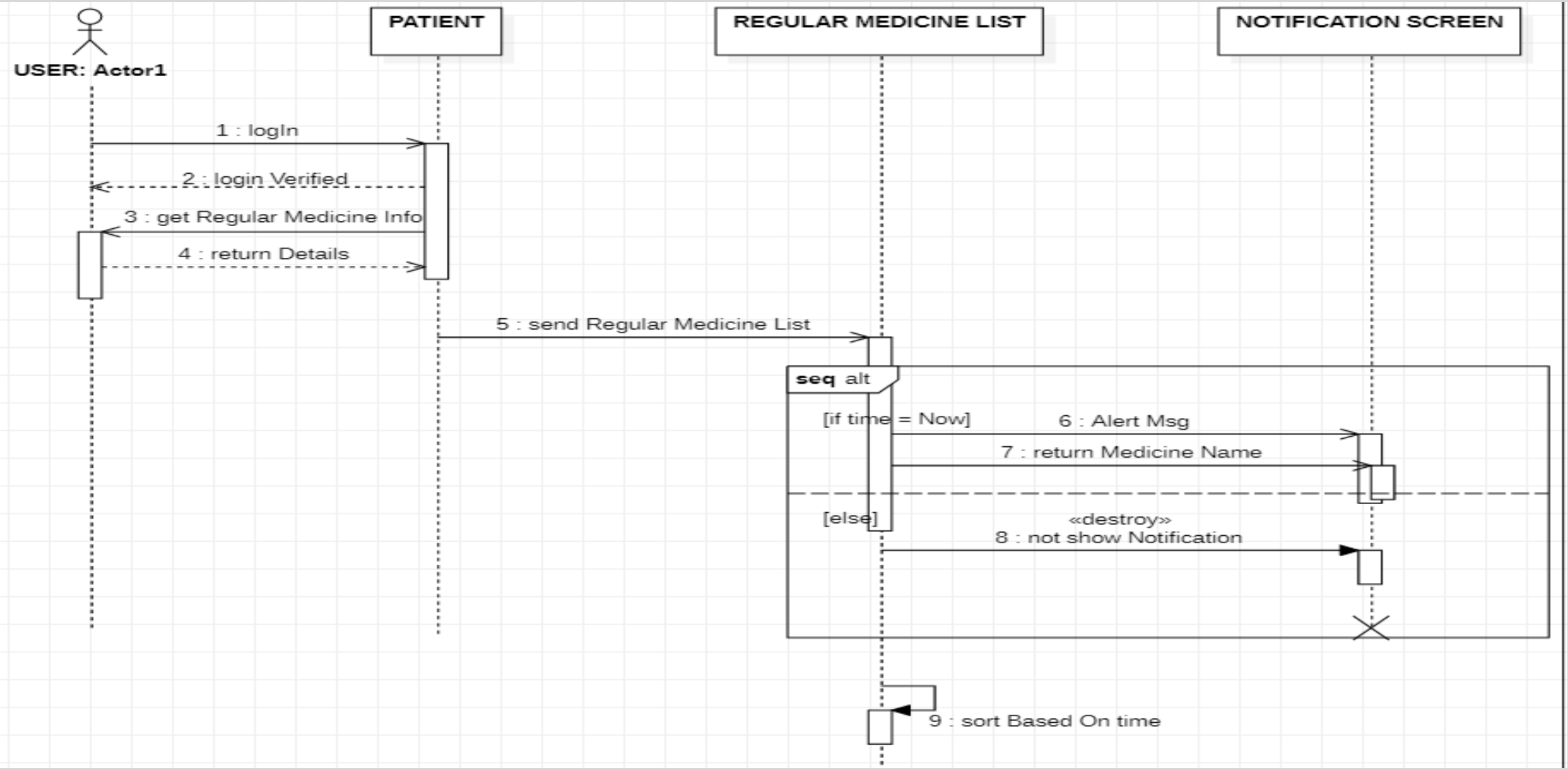
- User asked to enter height and weight after getting details
- height is converted into metres and calculate height square and divide the weight by height and BMI is calculated
- if BMI value is greater that standard so returns as obese
- If BMI value is lesser than standard so returns as underweight
- if BMI value is equals to standard value so returns as fit

20.Alert appointment



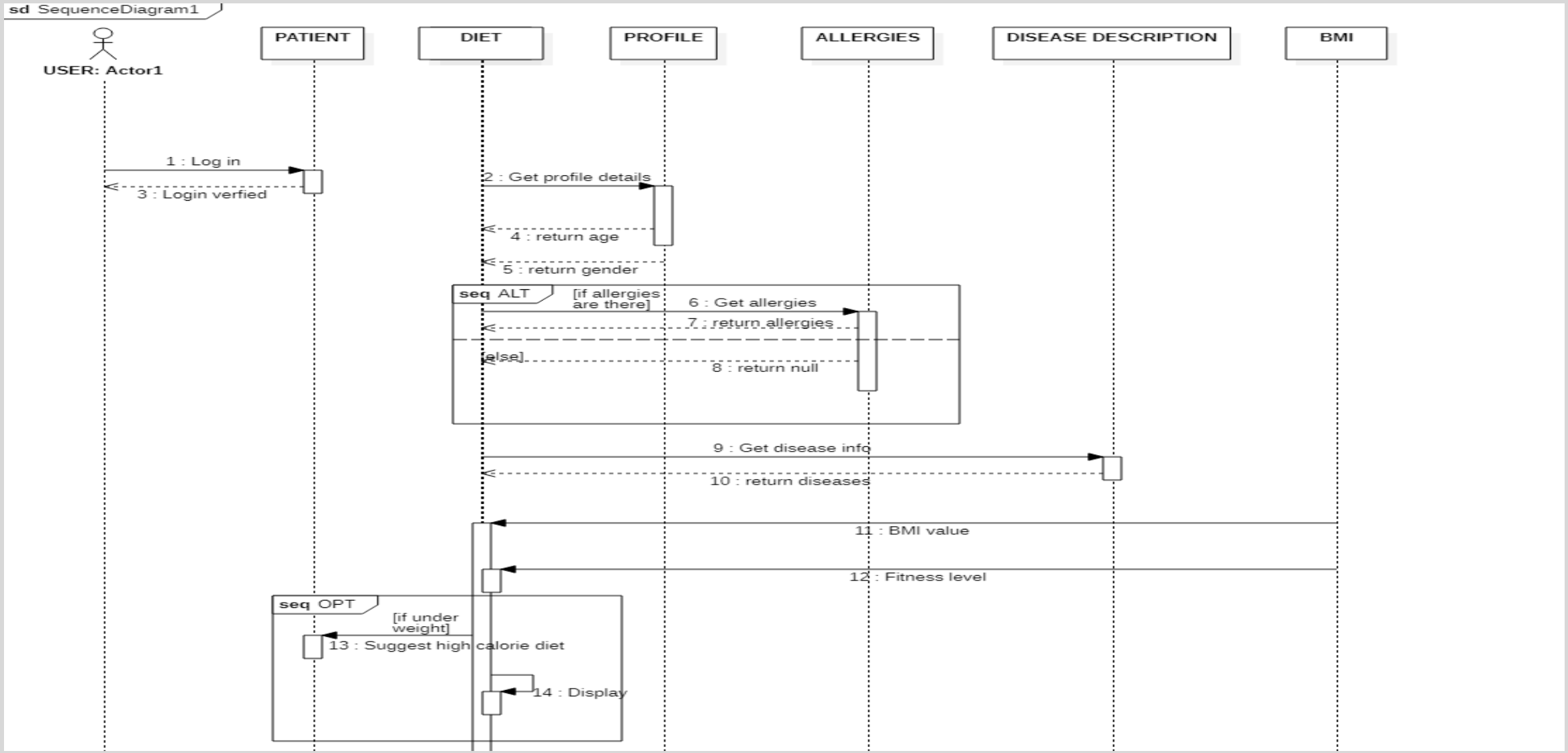
- User logged into system and book appointments
- user have to enter appointment date and time
- if Appointment date is equals to current date then alerts the user
- if Appointment time is same as current time then alerts the user
- Else sorting of appointments have done

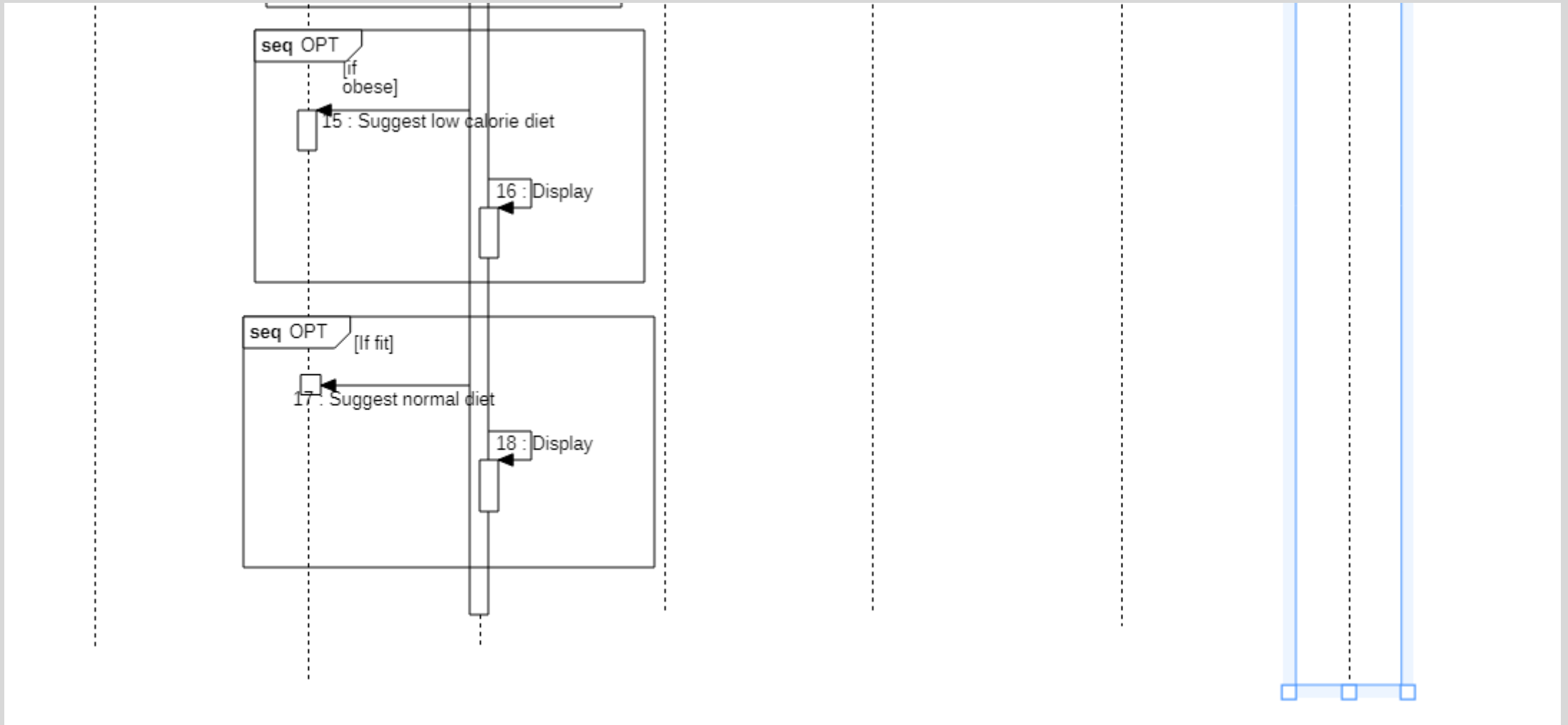
21.Alert medicine time



- User logged into system and asked to enter regular medicine details
- After getting those details will be stored
- if Medicine time is equals to current time then alert message will send to user
- Else user doesn’t get any notification

22.Suggest diet

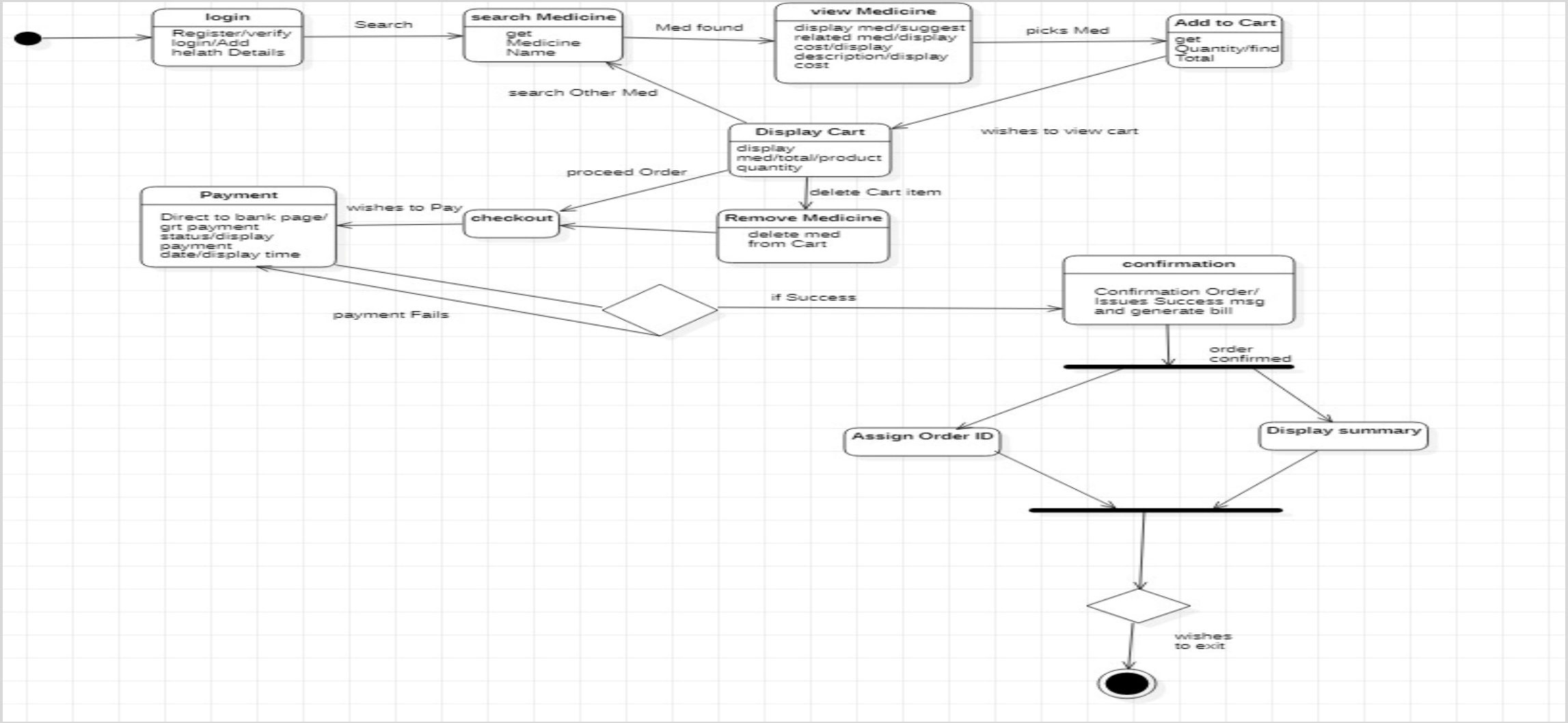



seq OPT
[if obese]
15 : Suggest low calorie diet
16 : Display
seq OPT
[If fit]
17 : Suggest normal diet
18 : Display

- User logged into system and give profile details
- get Allergies and disease details
- After getting all necessary info the BM will be computed
- Finding of fitness level have done
- If fitness level is obese then suggest low calorie diet
- If fitness level is underweight then suggest high calorie diet
- If fitness level is fit then suggest normal diet

STATE DIAGRAM

1.Ordering medicine



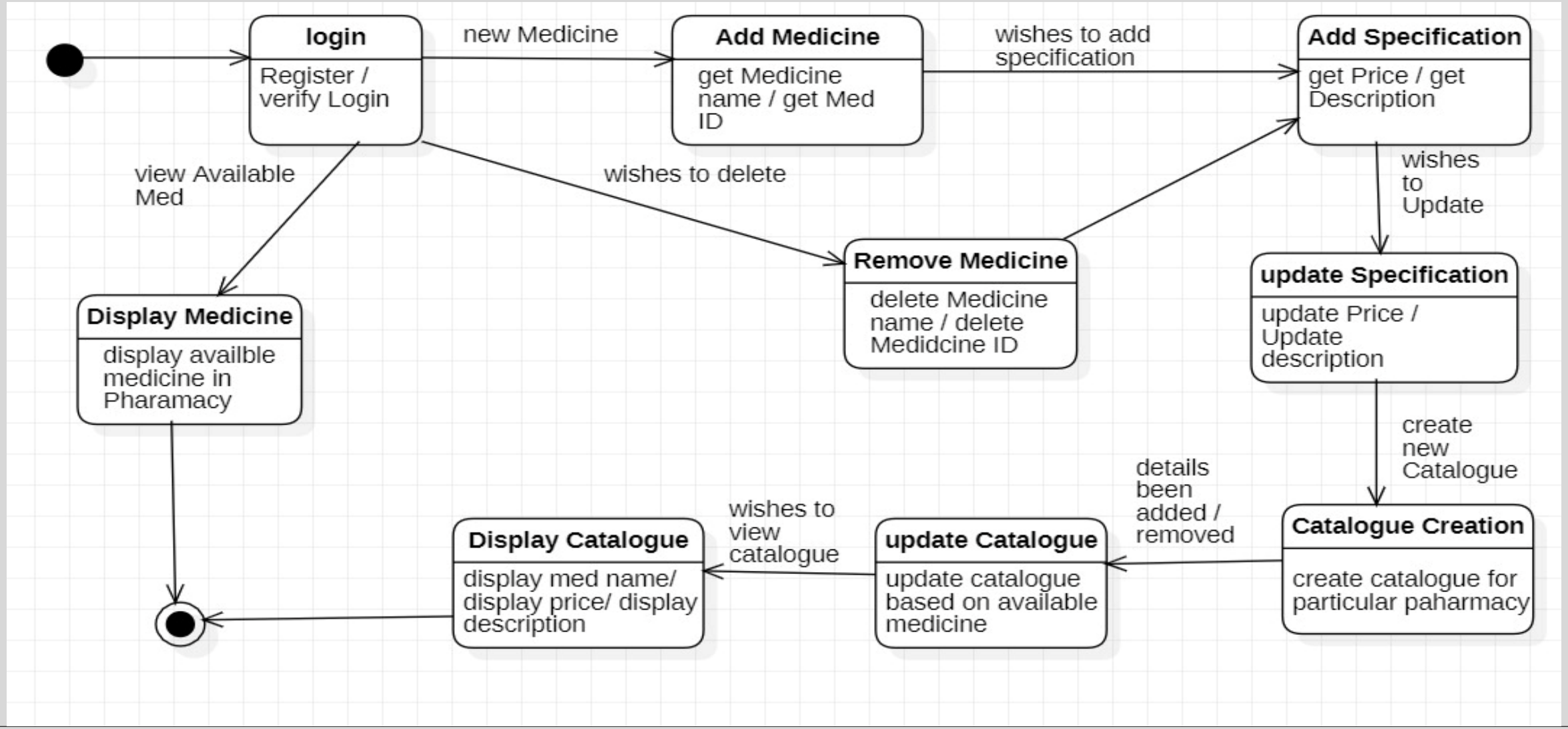
- From the start state, user activate by doing login in system
- After logged into system, the user search for desired medicine
- This leads to the next state called view medicine if the search medicine ids found
- If medicine is not found it returns back to search medicine state
- After viewed the medicine details, those medicines added to cart
- This leads to state called Display cart which displays the cart details
- This leads to checkout state and Remove state
- If user is okay with cart items then proceed checkout state
- If user want to delete cart item then go with Remove Medicine state after proceed with checkout sate
- By doing checkout, user make Payment if he/she wishes to pay
- here We have choices they are if payment success or Fails
- If decision result on Success leads to Confirmation state and here we have Fork operation

    ->Assign Order ID

    -> Display summary

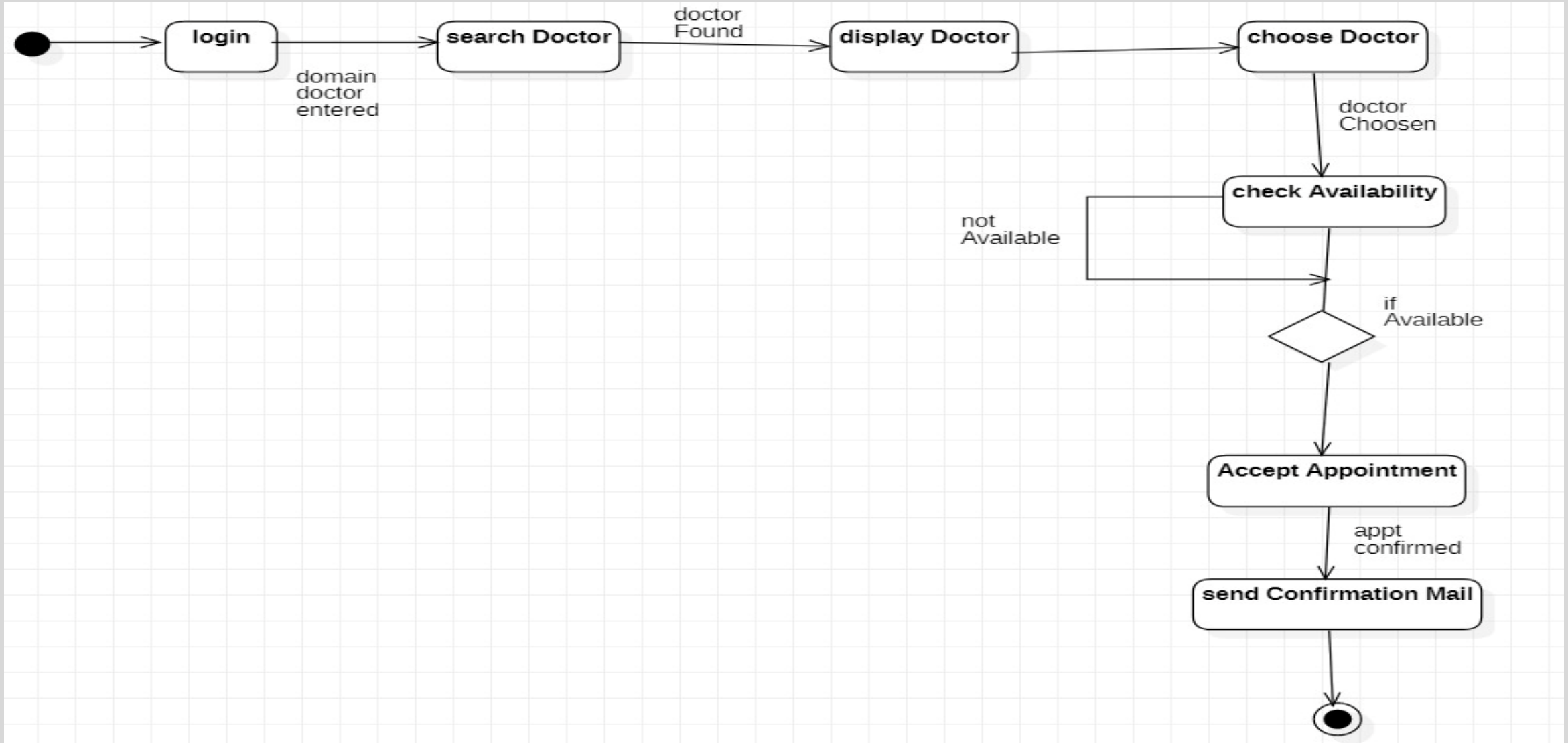
- These 2 states joined together results exit state that is Final state

2.Catalogue creation



- From the start state, Pharmacy staff activates by doing login into system
- Pharmacy staff has new medicine this leads to Add medicine state
- If the staff wants to delete medicine then leads to Remove Medicine state
- If Staff wants to view Available medicine then leads to Display Medicine state
- They wishes to add specifications then leads to new state called Add specification
- And if they want to update specification leads to update Specification
- Then it leads to Catalogue creation state
- Deletion/ addition of details have been done leads to Update catalogue state
- Staff wants to view Catalogue this leads to Display catalogue state and enter into Final state

3.Booking appointment

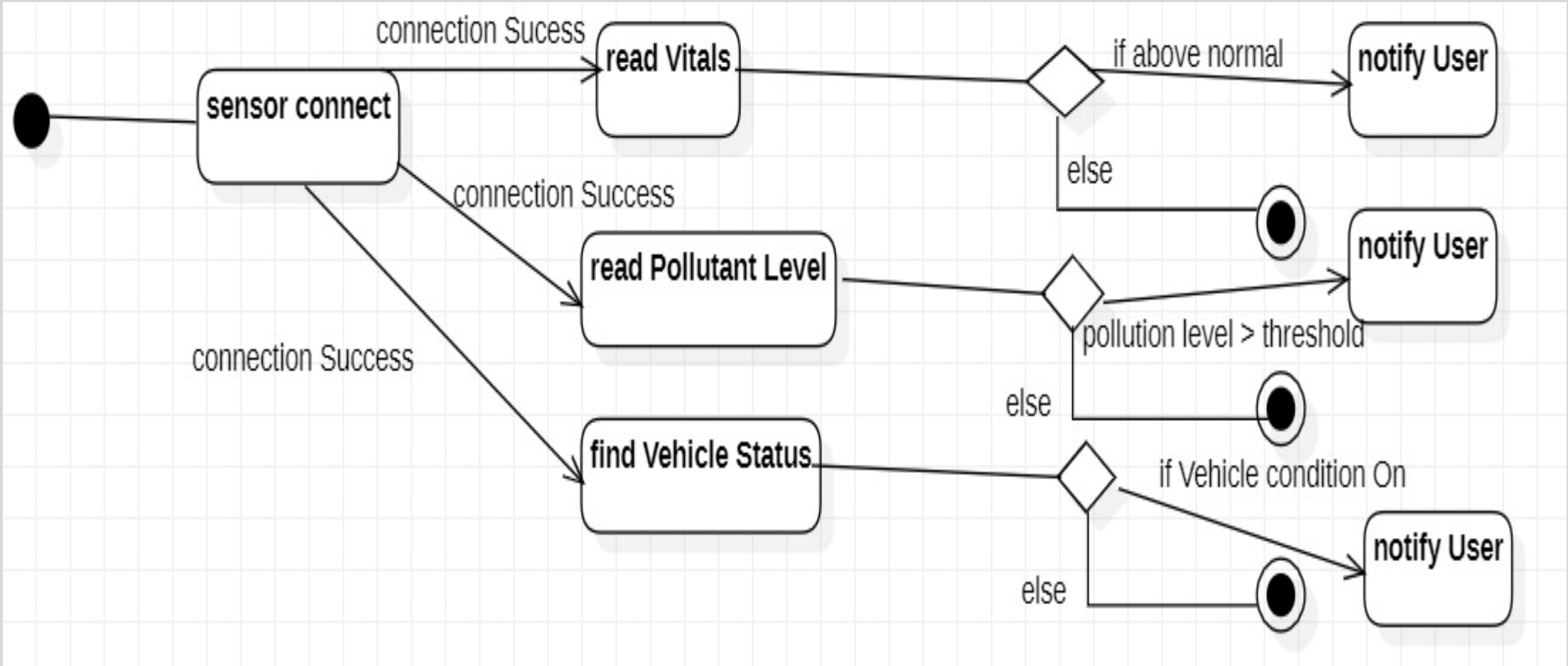


- From the start state the user activates by doing login in system
- User enter desired Doctor's domain
- After that searching of doctor in respective domain have been done
- Display doctor’s of desired domain
- And the user choose a desired doctor from list of doctors
- checking the availability of doctor have been done
- if the decision results s unavailable then it returns back to search for another doctor sate
- here if the decision results as doctor is available then it leads to Accept appointment state

and then it leads to send confirmation mail state

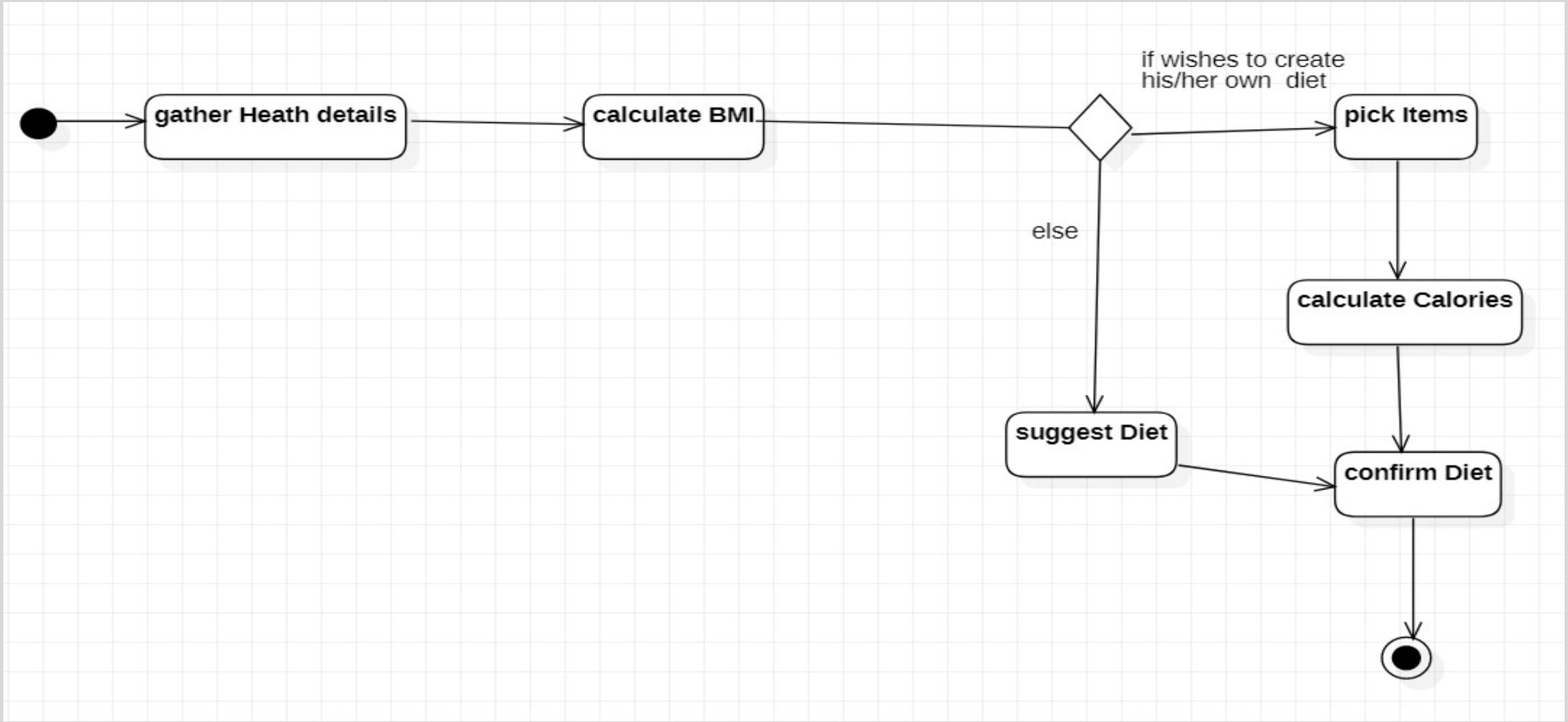
- And get into final state

- 4. Alert creation for abnormal readings



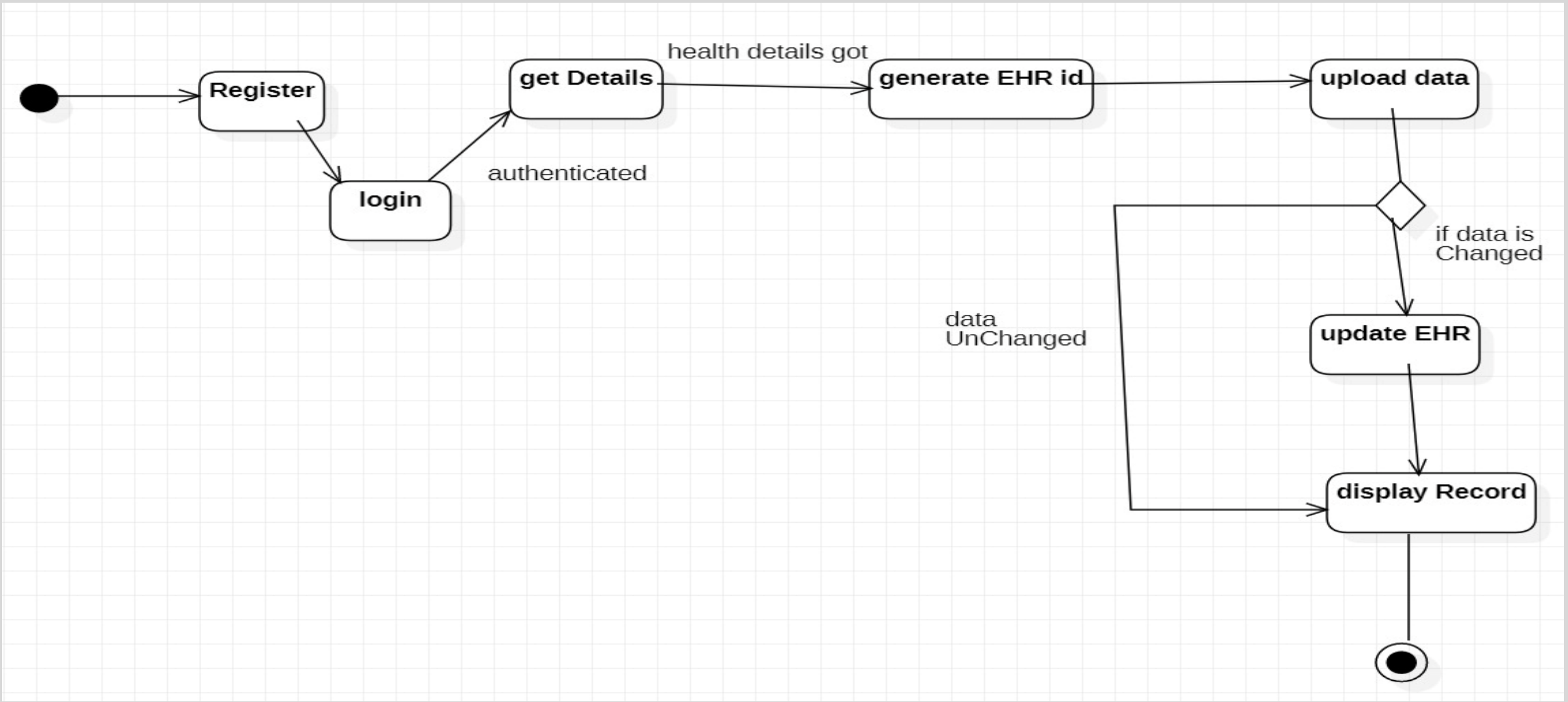
- From the start state sensor initiates by establishing connection
- The connection have been done with vital sensor, vehicle sensor, and pollution sensor
- here we have 2 decision they are above normal and normal
- if recorded vitals is above the normal then leads to notify user state and else gets into final state
- If Pollution level is greater than threshold then it leads to notify user state and else gets into final state
- If Vehicle condition is on then it leads to notify user state and else gets into final state

5. Diet choosing process



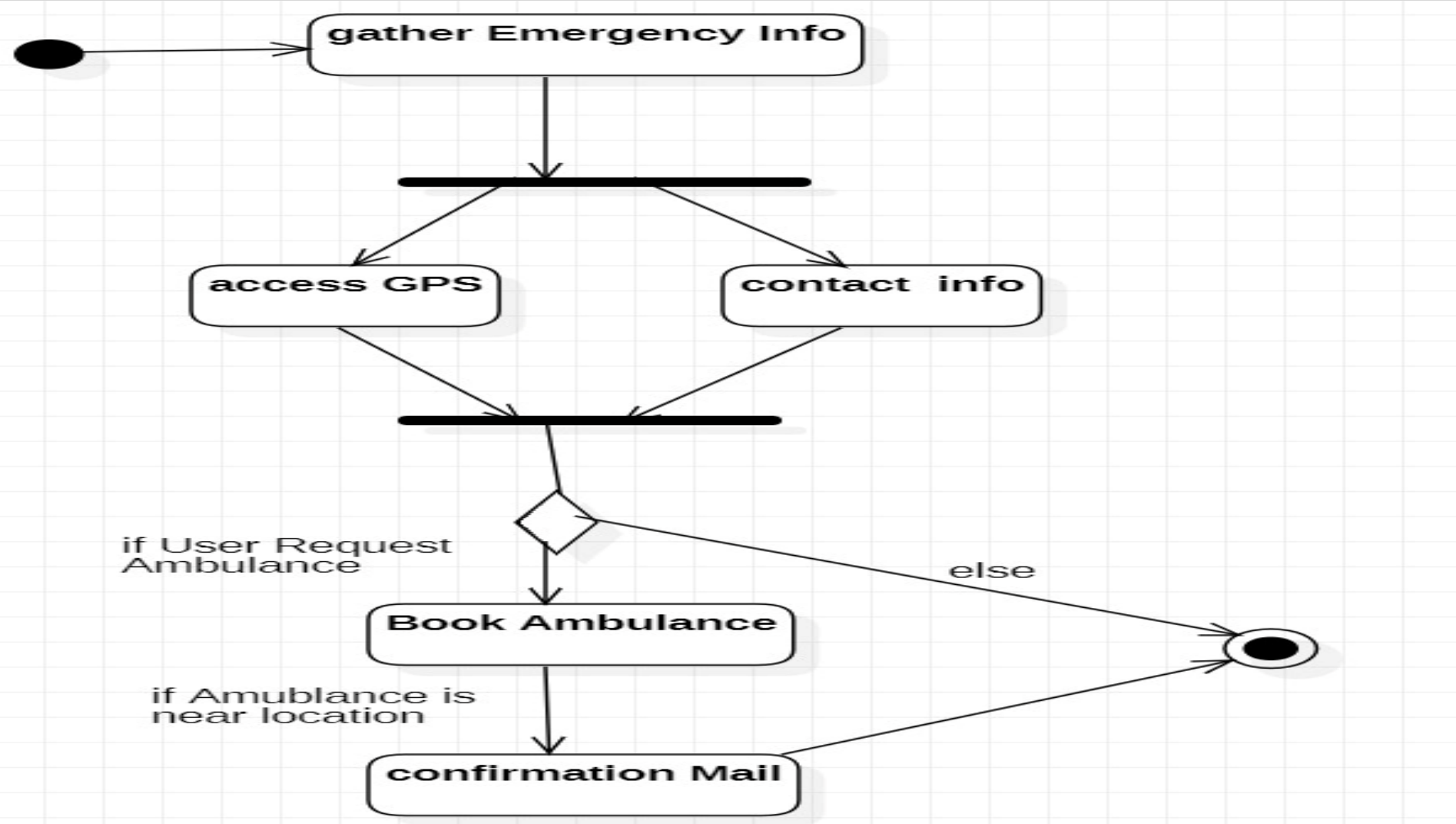
- From the start state the user activates by entering health details
- It leads to Calculate BMI state which have calculated from the health details
- In next state we have two decisions they are Pick items and suggest diet
- If user wants to create his/her own state then they have to go with pick item state
- After picked up items the calories will be calculated for those items
- it leads to confirm diet state and get into final state
- Else if the appropriate diet will be suggested by system itself and then user have to confirm the suggested diet and gets into final state

6.Health record creation



- From the start state the user initiates by doing Register
- it leads to login state
- After doing authentication the details will be got from the user
- This leads to get details state in that the details have got
- Once the details have been got it leads to generate HER id
- then it leads to the received data will be stored in E-Health record
- here we have decisions like Update HER if user want to change the uploaded data then this state take part
- Else if that data remains unchanged then it leads to Display record state and gets into final state

7. Booking of ambulance



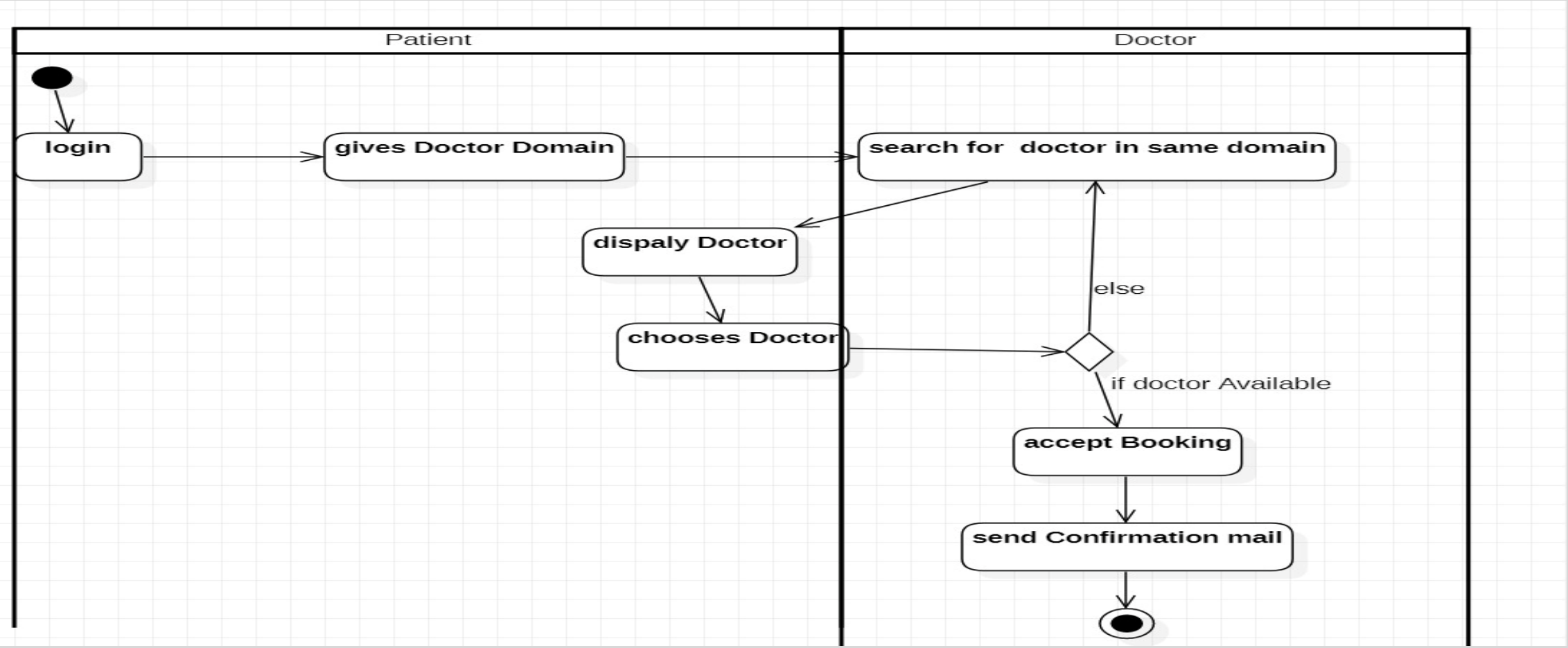
- From the start state the user initiates by entering Emergency info
- this leads to fork operations:

    -> Access GPS

    ->Contact Info
- these two states joined together and results new state called Book Appointment state
- here we have choices like user wants Ambulance service or not
- If user wants ambulance service then it leads to Book appointment state
- Else user don’t want Ambulance service then it result final state
- if Ambulance location is near then it leads to Confirmation mail send state
- Gets into final state

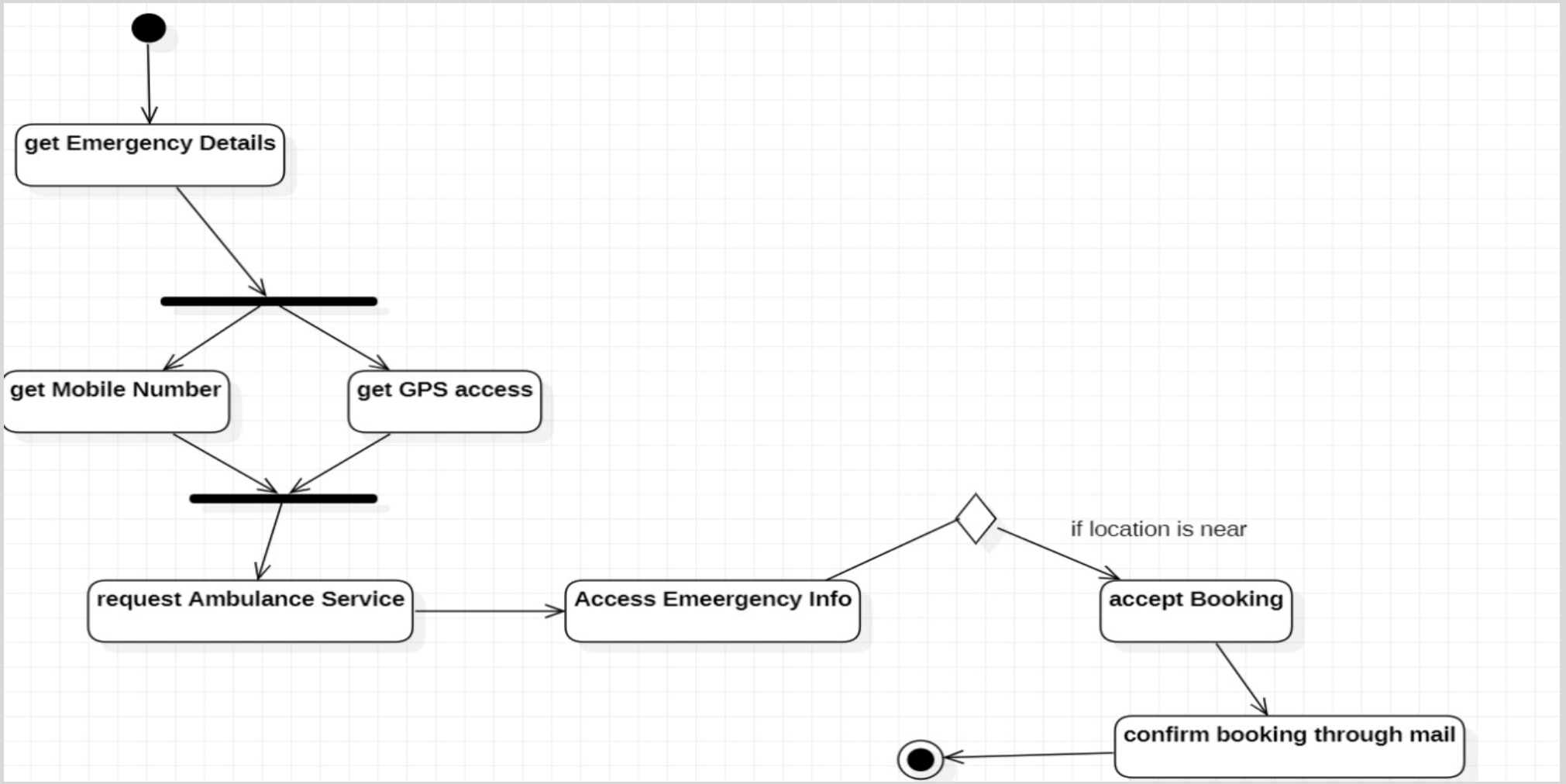
ACTIVITY DIAGRAM

1.Appointment booking



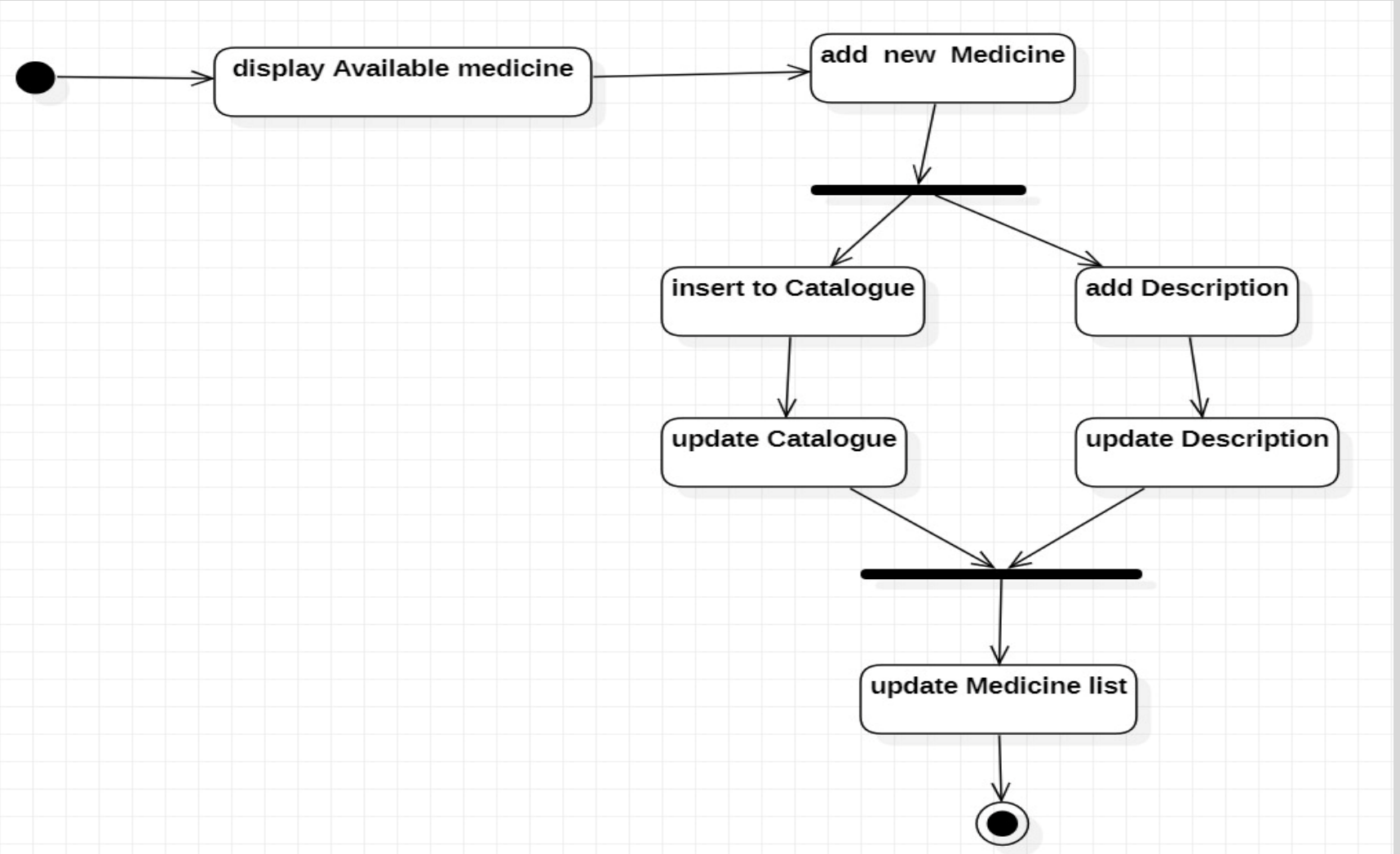
- This activity model uses a swim lane diagram and the two objects used here are **user** and **system**
- From the initial point, the user activates **system**
- Once the user has entered the doctor's domain
- System search doctor's of same domain which have entered by user
- In next state user gets the doctor list
- By referring the list, user can choose the desired doctor from the list
- Then we have a activity of checking availability where there will be choices of available and unavailable
- If the doctor is available then booking has been accepted and confirmation mail has sent
- If the doctor is unavailable then the patient have to choose for another doctor from the list

2.Ambulance booking



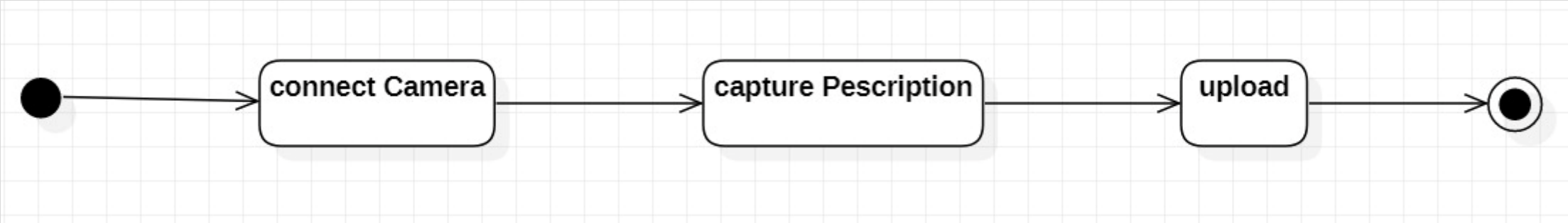
- From the initial point, the Emergency details has been received
- We have a Fork operation where the get Mobile number and get GPS access
- Fork operation:
- Get mobile number
- Get GPS access
- Then these 2 activities will be joined together
- Then the user request the Ambulance service
- Once the Ambulance service gets the user's request, they get the access of user's Emergency details which has been entered by user above
- By using this, if user's location is near , they accept user's ambulance service request
- And they send the confirmation mail to user then the activity will be terminated.

3. Medicine list updation



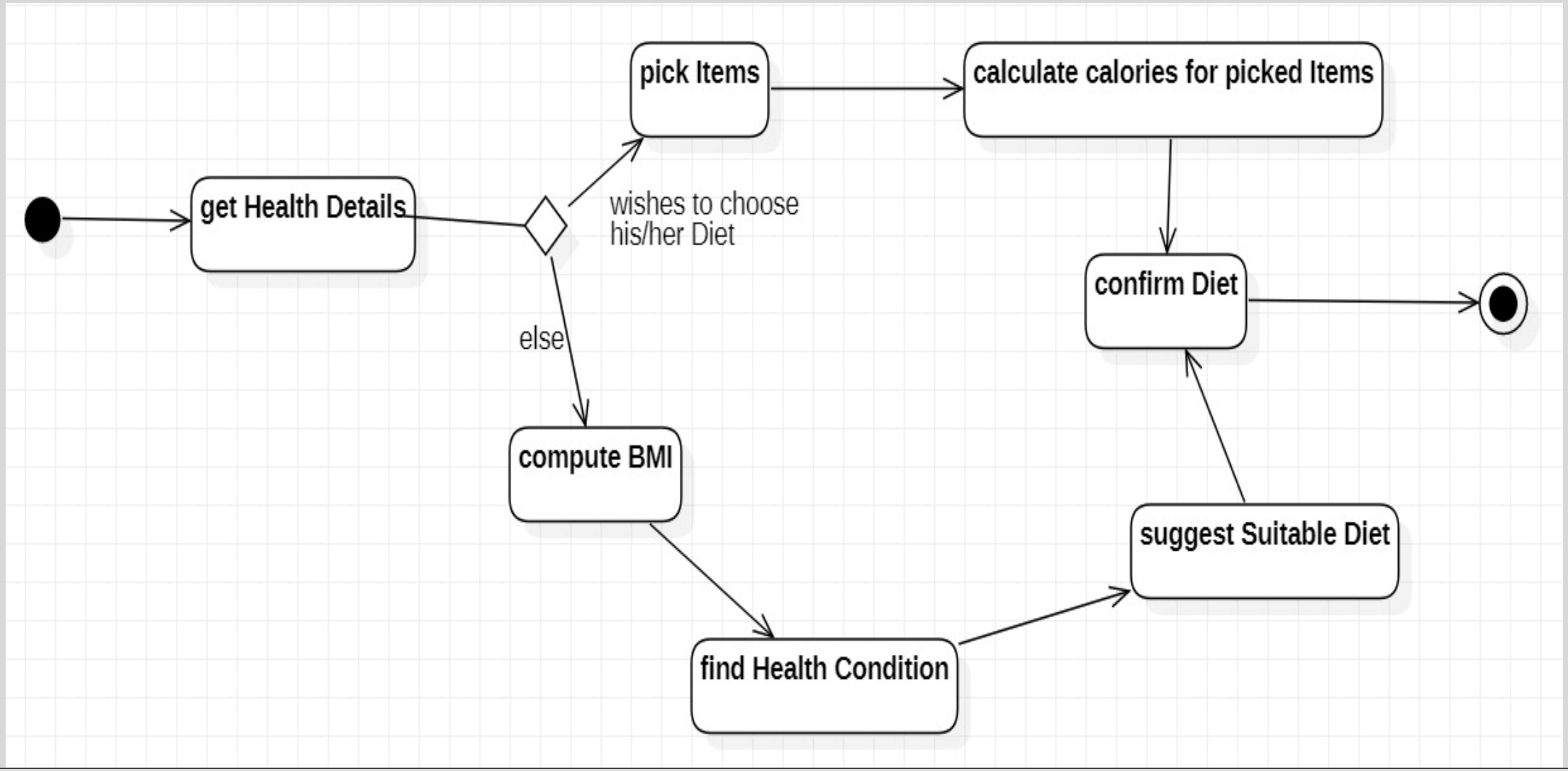
- The Pharmacy staff initiates the activity by getting the list of available medicine
- Pharmacy staff wans to add new medicine which results in Fork operation
- We have a Fork operation where the insert to catalogue and add description
- Fork operation:
- -> Insert to catalogue
- -> Add description
- Insert to catalogue and Add description results in another two states Update catalogue and update description respectively
- These 2 activities will be joined together and result in another state that is Update medicine list

4.Uploading prescription



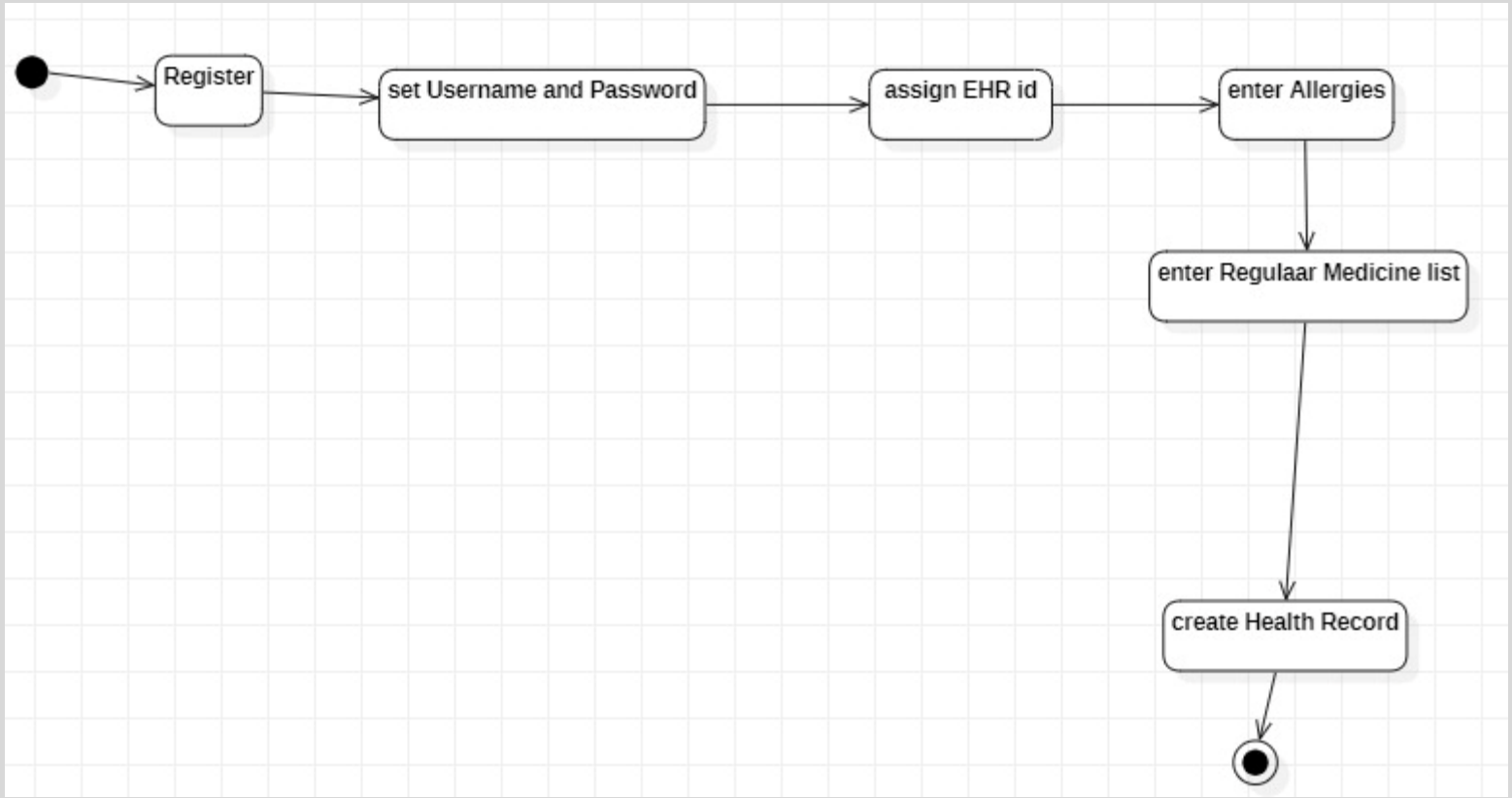
- The user initiates the activity by connecting camera
- Once the camera connected successfully, the prescription will be captured
- then the captured one will be uploaded through the system

5.Diet suggestion



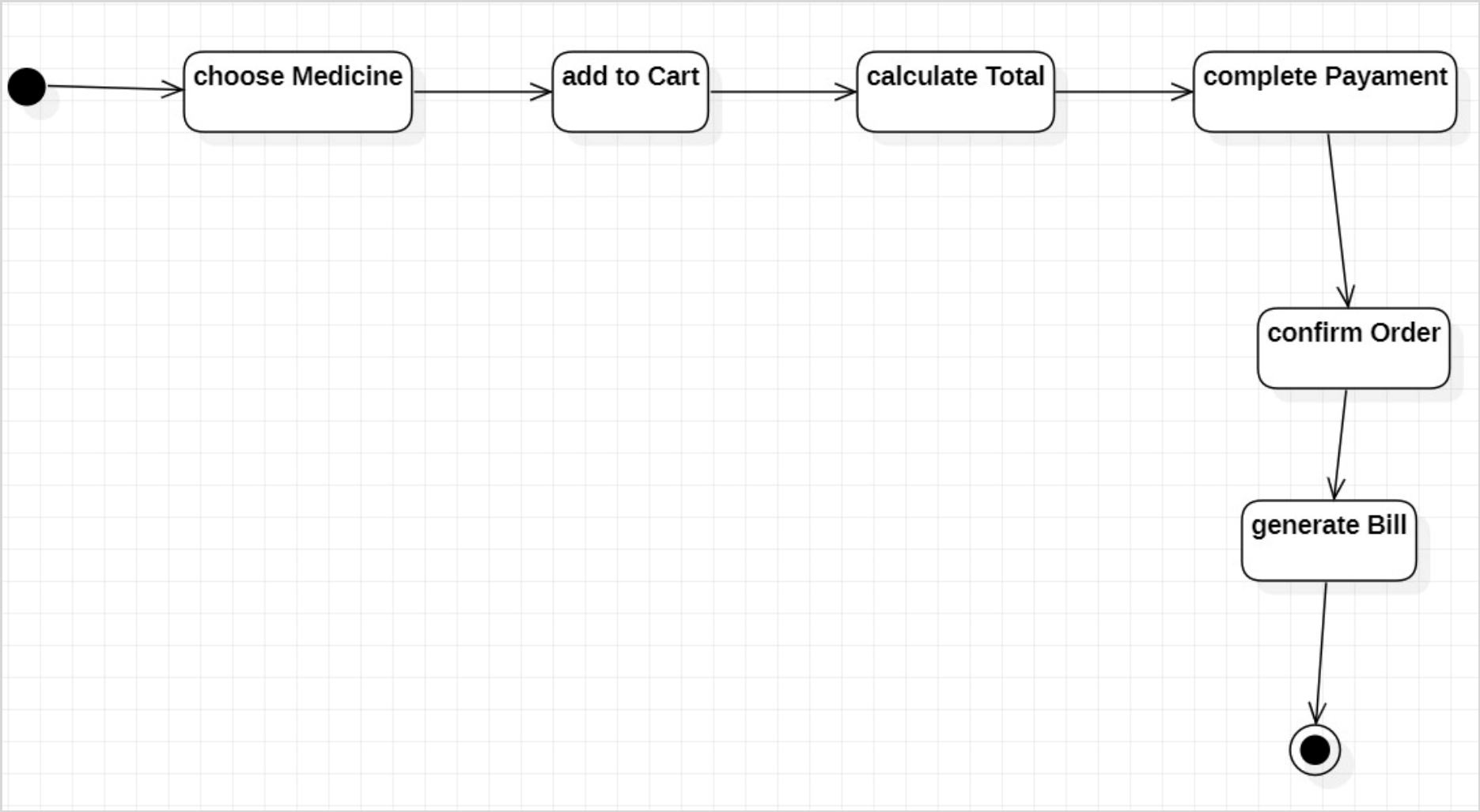
- User initiates the activity by entering health details
- We have choice of compute BMI and wishes to choose their own diet
- If the decision is made on compute BMI then health condition will be analysed
- Based on this the Suitable diet will be suggested
- And the user have to confirm the suggested diet plan and then activity will be terminated
- If the decision is made on Wishes to choose diet then the user have to pick the desired food items
- Based on the picked items the calorie will be calculated
- And the diet was prepared and then activity will be terminated

6.Creating health record



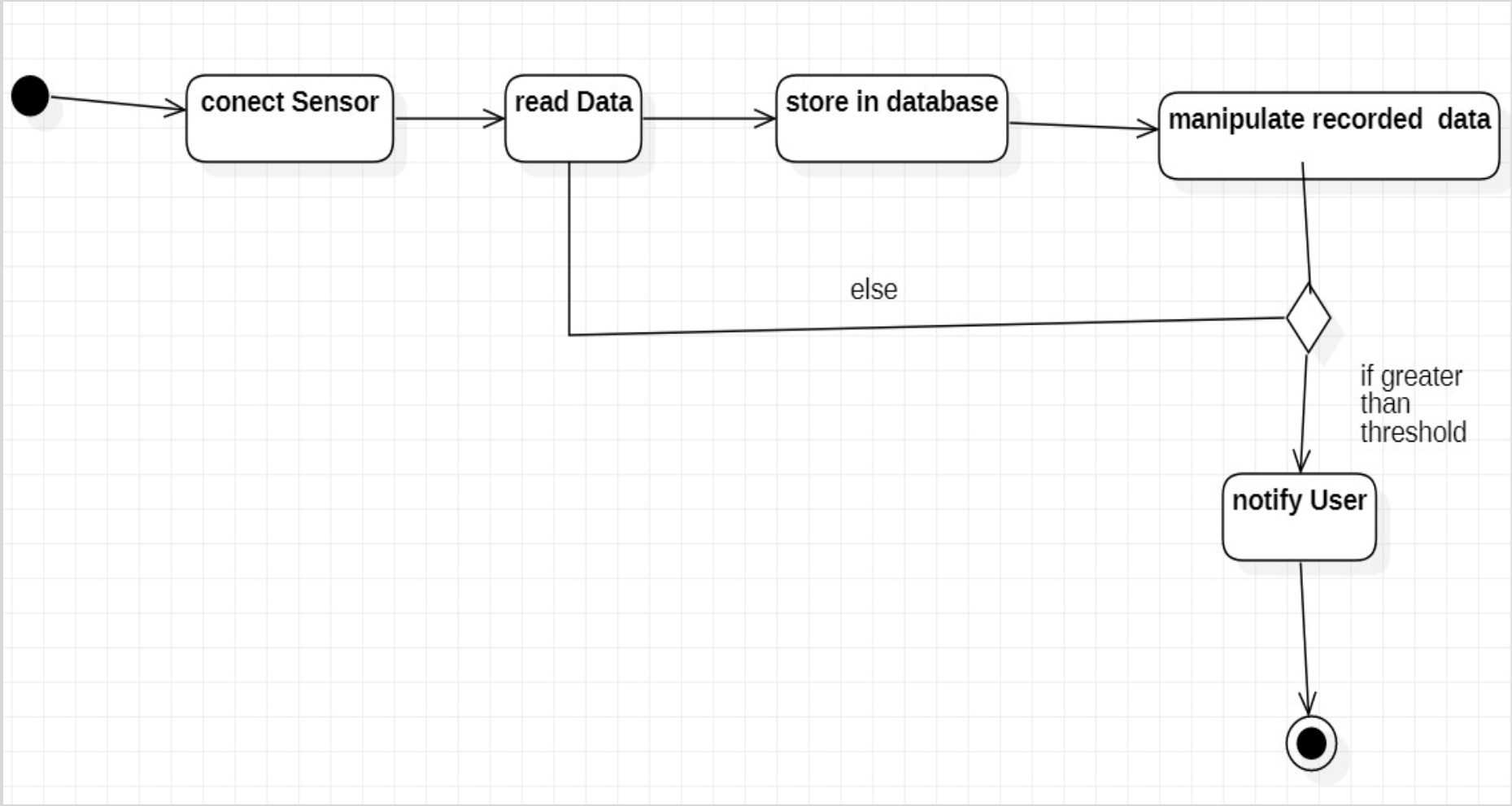
- The user initiate the activity by doing register into the system
- User have to set username and password for their Account
- Once the user has completed the account creation, the system will assign the unique E-Health record ID
- By using their Id the user have to enter the allergies details they have got
- User have to enter Regular Medicines list
- In next state unique Health record will be created and then activity ill be terminated

7. Bill generation for an order



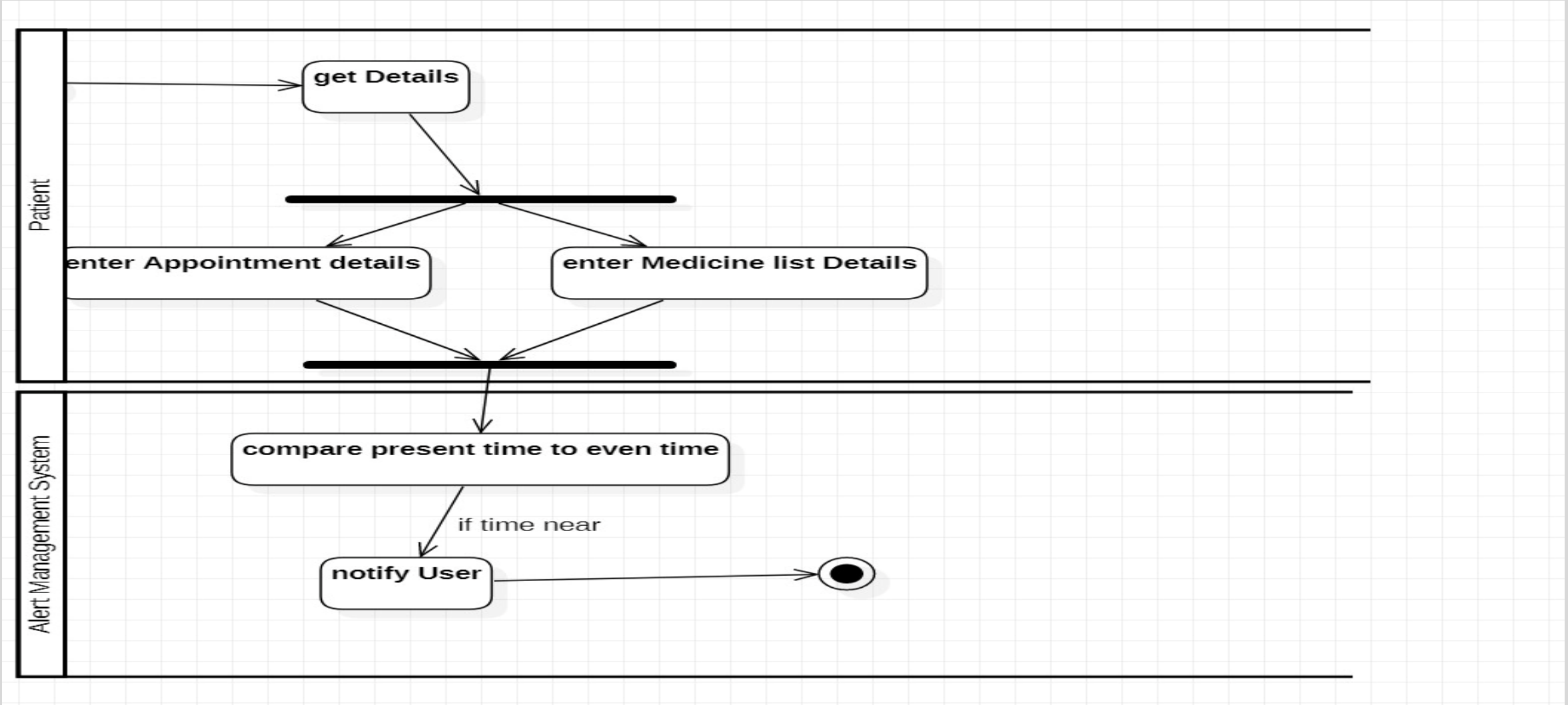
- The user initiates the activity by choosing desired medicine
- In next state the added medicines will be added to cart
- Then the total amount will be calculated for the cart items
- And the user do payment for the those medicine
- After the completion of payment, order will be confirmed
- In last state, the Bill will be generated for the order

8.Alert creation process



- The activity is initiated by connecting sensor
- After the connection will be established , the sensor starts to read data
- The recorded data will be stored in database
- And the manipulation of data have been done
- we have choices that red data is greater than threshold or not
- If the decision is captured data is greater than threshold range then notification will be send to user

9.Notification of appointment time and date



- From the initial state the details will be got
- We have Fork operation where Enter appointment details and Enter medicine list
- Fork operation:
- -> Enter appointments details
- -> Enter medicine list
- These 2 states will be joined and result in new state called compare present time to new time
- In this state the present time will be compared to the time of regular medicine
- If the current time is near the notification will be send to user and then activity will be terminated

CODE GENERATION

1.Allergies.java

```
import java.util.*;
/**
 *
 */
public class ALLERGIES {
    /**
     * Default constructor
     */
    public ALLERGIES() {
    }
    /**
     *
     */
    private String symptoms;
    /**
     *
     */
    private String cause;
    /**
     *
     */
    private String comment;
    /**
```

2.Ambulace.java

```
import java.util.*;
/**
 *
 */
public class AMBULANCE extends USER {
    /**
     * Default constructor
     */
    public AMBULANCE() {
    }
    /**
     *
     */
    private String location;
    /**
     *
     */
    private String driver_name;
    /**
     *
     */
    private Int contact_no;
    /**
     *
     */
```

```
    public void confirm booking() {
            // TODO implement here
        }
        /**
         *
         */
        public void check location() {
            // TODO implement here
        }
    }
```

3.Appointment.java

```
import java.util.*;
/**
 *
 */
public class APPOINMENTS {
    /**
     * Default constructor
     */
    public APPOINMENTS() {
    }
    /**
     *
     */
    private Date date;

    /**
     *
     */
    private Time time;

    /**
     *
     */
```

```
private String doctorname;


    /**
     *
     */
    private Boolean isdatenear;
    /**
     *
     */
    public void checkavailability() {
        // TODO implement here
    }


    /**
     *
     */
    public void sendconfirmation() {
        // TODO implement here
    }


    /**
     *
     */
    public void update() {
        // TODO implement here
    }
```

4.BMI.java

```
import java.util.*;
/**
 *
 */
public class BMI {
    /**
     * Default constructor
     */
    public BMI() {
    }
    /**
     *
     */
    private Int height;
    /**
     *
     */
    private Int weight;
    /**
     *
     */
```

```
private Int range;
    /**
     *
     */
    public void calculateBMI() {
        // TODO implement here
    }
    /**
     *
     */
    public void fitnesslevel() {
        // TODO implement here
    }

}
```

5.Camera.java

```
import java.util.*;
/**
 */
public class CAMERA {


    /**
     * Default constructor
     */
    public CAMERA() {
    }
    /**
     */
    private String camid;
    /**
     */
    private String status;
    /**
     */
    public void connect() {
        // TODO implement here
    }


}
```

6.Cart.java

```
import java.util.*;
/**
 */
public class CART {
    /**
     * Default constructor
     */
    public CART() {
    }
    /**
     */
    private String cartid;
    /**
     *
     */
    private Int noofitems;
    /**
     */
    public void add cart() {
        // TODO implement here
    }
    /**
     */
```

```
public void view cart() {
        // TODO implement here
    }
    /**
     */
    public void update() {
        // TODO implement here
    }
    /**
     */
    public void checkout() {
        // TODO implement here
    }
    /**
     */
    public void price() {
        // TODO implement here
    }

}
```

7.Dietplan.java

```
import java.util.*;
/**
 */
public class DIET PLAN {
    /**
     * Default constructor
     */
    public DIET PLAN() {
    }
    /**
     */
    private String type;
    /**
     */
    private Int calories;
    /**
     */
    private String preference;
    /**
     */
    public void calculate calorie() {
        // TODO implement here
    }
    /**
     */
```

8.Diseasedescription.java

```
import java.util.*;
/**
 */
public class DISEASE DESCRIPTION {
    /**
     * Default constructor
     */
    public DISEASE DESCRIPTION() {
    }
    /**
     */
    private String disease_name;
    /**
     */
    private String medicine;
    /**
     */
    private Int med_period;
    /**
     */
```

```
 private Boolean isgenetic;
    /**
     */
    public void adddisease() {
        // TODO implement here
    }
    /**
     */
    public void removedisease() {
        // TODO implement here
    }
}
```

9.Doctor.java

```
import java.util.*;
/**
 */
public class DOCTOR extends USER {
    /**
     * Default constructor
     */
    public DOCTOR() {
    }
    /**
     */
    private String field;
    /**
     */
    private String designation;
    /**
     */
    private String name;
    /**
     */
    private String emailid;
    /**
     */
```

```
    public void prescribemed() {
        // TODO implement here
    }
    /**
     */
    public void acceptapptmts() {
        // TODO implement here
    }
    /**
     */
    public void booksenior() {
        // TODO implement here
    }

}
```

10.Ehr.java

```
import java.util.*;
/**
 */
public class EHR {
    /**
     * Default constructor
     */
    public EHR() {
    }
    /**
     */
    private String Hid;
    /**
     */
    public void updaterecord() {
        // TODO implement here
    }
}
```

11.Emergencyinfo.java

```
import java.util.*;
/**
 */
public class EMERGENCY INFO {
    /**
     * Default constructor
     */
    public EMERGENCY INFO() {
    }
    /**
     */
    private String location;
    /**
     */
    private Int mobileno;
    /**
     */
    public void currentlocation() {
        // TODO implement here
    }

}
```

12.Item.java

```
import java.util.*;
/**
 */
public class ITEM {
    /**
     * Default constructor
     */
    public ITEM() {
    }
    /**
     */
    private String itemid;
    /**
     */
    private Int quantity;
    /**
     */
    private Int unitprice;
    /**
     */
```

```
    public void search() {
        // TODO implement here
    }
    /**
     */
    public void add() {
        // TODO implement here
    }
    /**
     */
    public void remove() {
        // TODO implement here
    }
}
```

13.Measurements.java

```
import java.util.*;
/**
 */
public class MEASUREMENTS {
    /**
     * Default constructor
     */
    public MEASUREMENTS() {
    }
    /**
     */
    private Int o2level;
    /**
     */
    private Int co2level;
    /**
     */
    private Int bodytemp;
    /**
     */
    private Int pulse;
    /**
     */
```

```
public void setbenchmarks() {
        // TODO implement here
    }
    /**
     */
    public void getreading() {
        // TODO implement here
    }
    /**
     */
    public void notifyuser() {
        // TODO implement here
    }
}
```

14.Medicinespecification.java

```
public void setbenchmarks() {
        // TODO implement here
    }
    /**
     */
    public void getreading() {
        // TODO implement here
    }
    /**
     */
    public void notifyuser() {
        // TODO implement here
    }
}
```

15.Medicinebill.java

```
import java.util.*;
/**
 */
public class MEDICINE BILL {
    /**
     * Default constructor
     */
    public MEDICINE BILL() {
    }
    /**
     */
    private String billno;
    /**
     */
    private Int amt;
    /**
     */
    public void generatebill() {
        // TODO implement here
    }
}
```

16.Medicinecatalogue.java

```
import java.util.*;
/**
 */
public class MEDICINE CATALOGUE {
    /**
     * Default constructor
     */
    public MEDICINE CATALOGUE() {
    }
    /**
     */
    private String serialnum;
    /**
     */
    public void display() {
        // TODO implement here
    }
    /**
     */
```

```
public void getspecification() {
        // TODO implement here
    }
    /**
     */
    public void updatecatalogue() {
        // TODO implement here
    }
}
```

17.Order.java

```
import java.util.*;
/**
 *
 */
public class ORDER {
    /**
     * Default constructor
     */
    public ORDER() {
    }
    /**
     *
     */
    private String orderid;
    /**
     *
     */
    private Date orderdate;
```

```
/**
 *
 */
 private String status;

 /**
 *
 */
 private String billing address;
```

18.Patient Profile.java

```
import java.util.*;

/**
 *
 */
public class PATIENT PROFILE {

    /**
     * Default constructor
     */
    public PATIENT PROFILE() {
    }


    private String profileid
    /**
     *
     */
    private String name;
```

```
    private String name;

    /**
     *
     */
    private Int age;

    /**
     *
     */
    private String gender;

    /**
     *
     */
    private String address;
     */
    public void updateprofile() {
        // TODO implement here
    }

}
```

19.Patient.java

```
import java.util.*;
/**
 *
 */
public class PATIENT extends USER {
    /**
     * Default constructor
     */
    public PATIENT() {
    }
    /**
     *
     */
    public void bookapptmts() {
        // TODO implement here
    }
    /**
     *
```

```
 */
    public void uploadpres() {
        // TODO implement here
    }

    /**
     *
     */
    public void paybill() {
        // TODO implement here
    }

}
```

20.Payment.java

```
import java.util.*;
/**
 *
 */
public class PAYMENT {
    /**
     * Default constructor
     */
    public PAYMENT() {
    }
    /**
     *
     */
    private String mode;

    /**
     *
     */
    private Date date;
```

```
  private Date date;


    /**
     *
     */
    private Time time;



    /**
     *
     */
    public void status() {
        // TODO implement here
    }

}
```

21.Pharmacy.java

```
import java.util.*;
/**
 *
 */
public class PHARMACY extends USER {
    /**
     * Default constructor
     */
    public PHARMACY() {
    }
    /**
     *
     */
    private String address;
    /**
     *
     */
    private String name;
```

```
 /**
     *
     */
    public void updateinfo() {
        // TODO implement here
    }
    /**
     *
     */
    public void register() {
        // TODO implement here
    }
    /**
     *
     */
    public void addpharmacy() {
        // TODO implement here
    }
}
```

22.Pollution.java

```
import java.util.*;
/**
 *
 */
public class POLLUTION extends SENSOR {
     * Default constructor
     */
    public POLLUTION() {
    }
    private Boolean isdetected;

    /**
     *
     */
    public void pollutionlevel() {
        // TODO implement here
    }

}
```

23.Prescription.java

```
import java.util.*;
/**
 *
 */
public class PRESCRIPTION {
    /**
     * Default constructor
     */
    public PRESCRIPTION() {
    }
    /**
     *
     */
    private String presid;

    /**
     *
     */
    private String doctorname;
```

```
 /**
     *
     */
    public void send prescription() {
        // TODO implement here
    }

}
```

24.Regular Medicine List.java

```
import java.util.*;
/**
 *
 */
public class REGULAR MEDICINE LIST {
    /**
     * Default constructor
     */
    public REGULAR MEDICINE LIST() {
    }
    /**
     *
     */
    private Date commence;

    /**
     *
     */
    private Date end;
```

```
/**
 *
 */
private Time medicinetime;
/**
 *
 */
private Boolean istimenear;
/**
 *
 */
public void sendnotification() {
    // TODO implement here
}
/**
```

```
 */
    public void update() {
        // TODO implement here
    }

    /**
     *
     */
    public void checkmedtime() {
        // TODO implement here
    }

}
```

25.Sensor.java

```
import java.util.*;
/**
 *
 */
public class SENSOR {

    /**
     * Default constructor
     */
    public SENSOR() {
    }

    /**
     *
     */
    private String sensorid;

}
```

26.User.java

```
import java.util.*;
/**
 *
 */
public class USER {
    /**
     * Default constructor
     */
    public USER() {
    }
    /**
     *
     */
    private String userid;
    /**
     *
     */
    private String psswd;
```

```
 /**
     *
     */
    public void verify login() {
        // TODO implement here
    }

    /**
     *
     */
    public void update profile() {
        // TODO implement here
    }

}
```

27.Vehicle.java

```
import java.util.*;
/**
 *
 */
public class VEHICLE extends SENSOR {
    /**
     * Default constructor
     */
    public VEHICLE() {
    }
    private Boolean ison;

    public void findvehiclestatus() {
        // TODO implement here
    }

}
```

28.Vital.java

```
import java.util.*;

/**
 *
 */
public class VITAL extends SENSOR {
     * Default constructor
     */
    public VITAL() {
    }
    private Boolean isabnormal;

    /**
     *
     */
    public void checkabnormal() {
        // TODO implement here
    }
}
```