Concordia University
Engineering and Computer Science

COEN-6312-2184-UU

Model Driven Software Engineering

Professor: Wahab Hamou-Lhadj, Ph.D., ing.

Deliverable 4

Submitted By:

NAME	ID	EMAIL
Ashish Sharma	40050452	ashish.sharma5293@gmail.com
Harmanpreet Singh	40059358	harmansandhu63@gmail.com
Navjot Kaur Bhamrah	40050459	navjotkaurbhamrah@gmail.com
Amandeep	40046716	amandpsingh03@gmail.com
Raghav Sharda	40053703	raghavsharma2926@outlook.com
Shivya Pant	40068007	shivyapant@gmail.com

1.) State Diagrams

State Diagrams describes the behavior of the object of a class. The behavior of the system can also be depicted with the state diagrams. At any point in time, an object will be at a particular state which then moves to another state on receiving a signal or event. The transition of a state to another happens in response to an event or a signal.

The three classes on which we prepared our state diagrams are:
1.1 Patient class
1.2 Doctor class
1.3 Admin class

Fig 1.1 Patient State Diagram

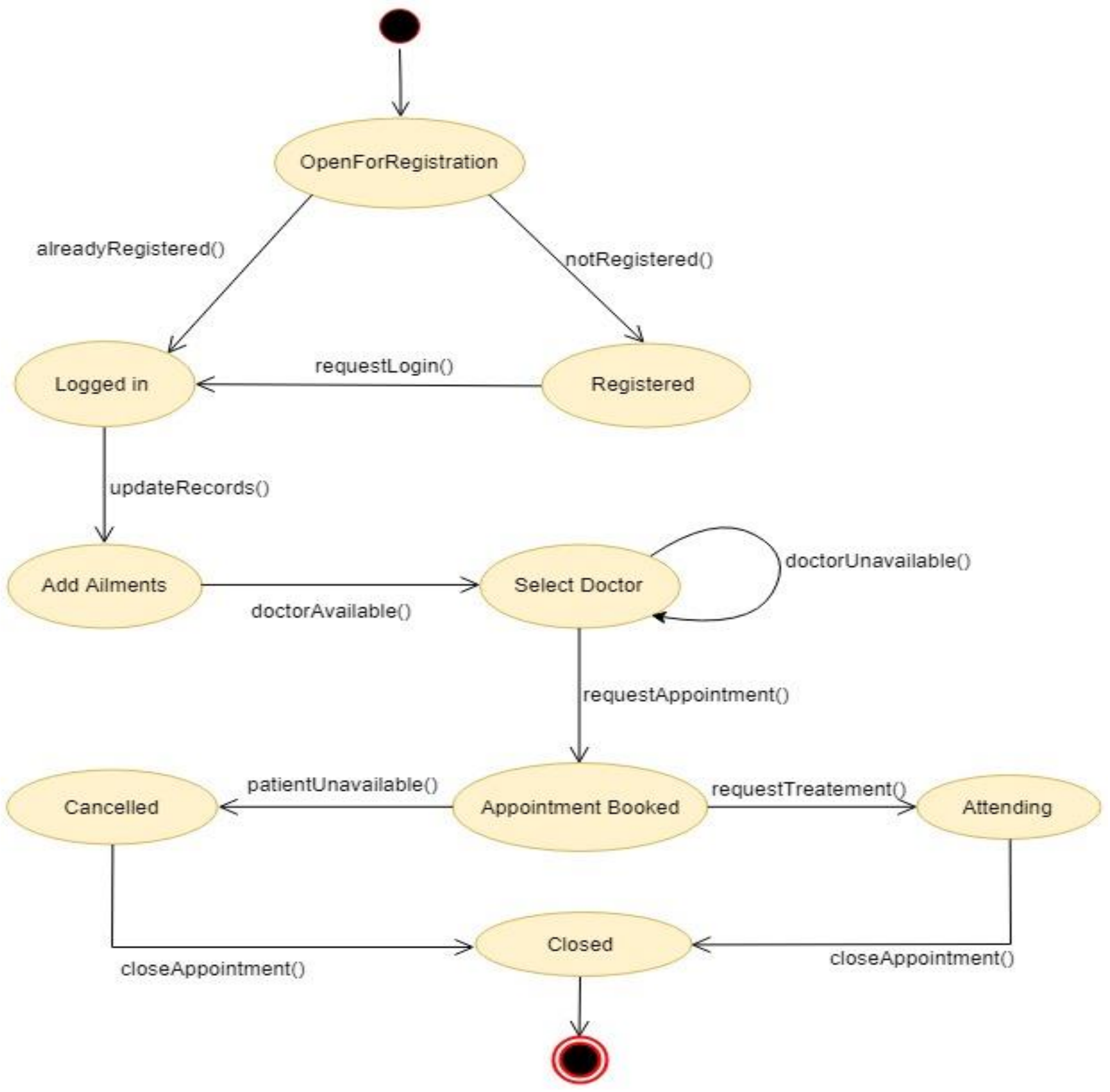

Fig 1.1.1 Patient State Transition Table

Event↓ State →	Open for Registration	Registered	Logged In	Add Ailments	Select Doctor	Appointment Booked	Canceled	Attending	Closed
alreadyRegistration()	**Logged In**								
notRegistered()	**Registered**								
requestLogin()		**Logged In**							
updateRecords()			**Add Ailments**						
doctorAvailable()				**Select Doctor**					
doctorUnavailable()					**Select Doctor**				
requestAppointment()					**Appointment Booked**				
patientUnavailable()						**Canceled**			
requestTreatment()						**Attending**			
closeAppointment()							**Closed**	**Closed**	

Fig 1.2 Doctor State Diagram

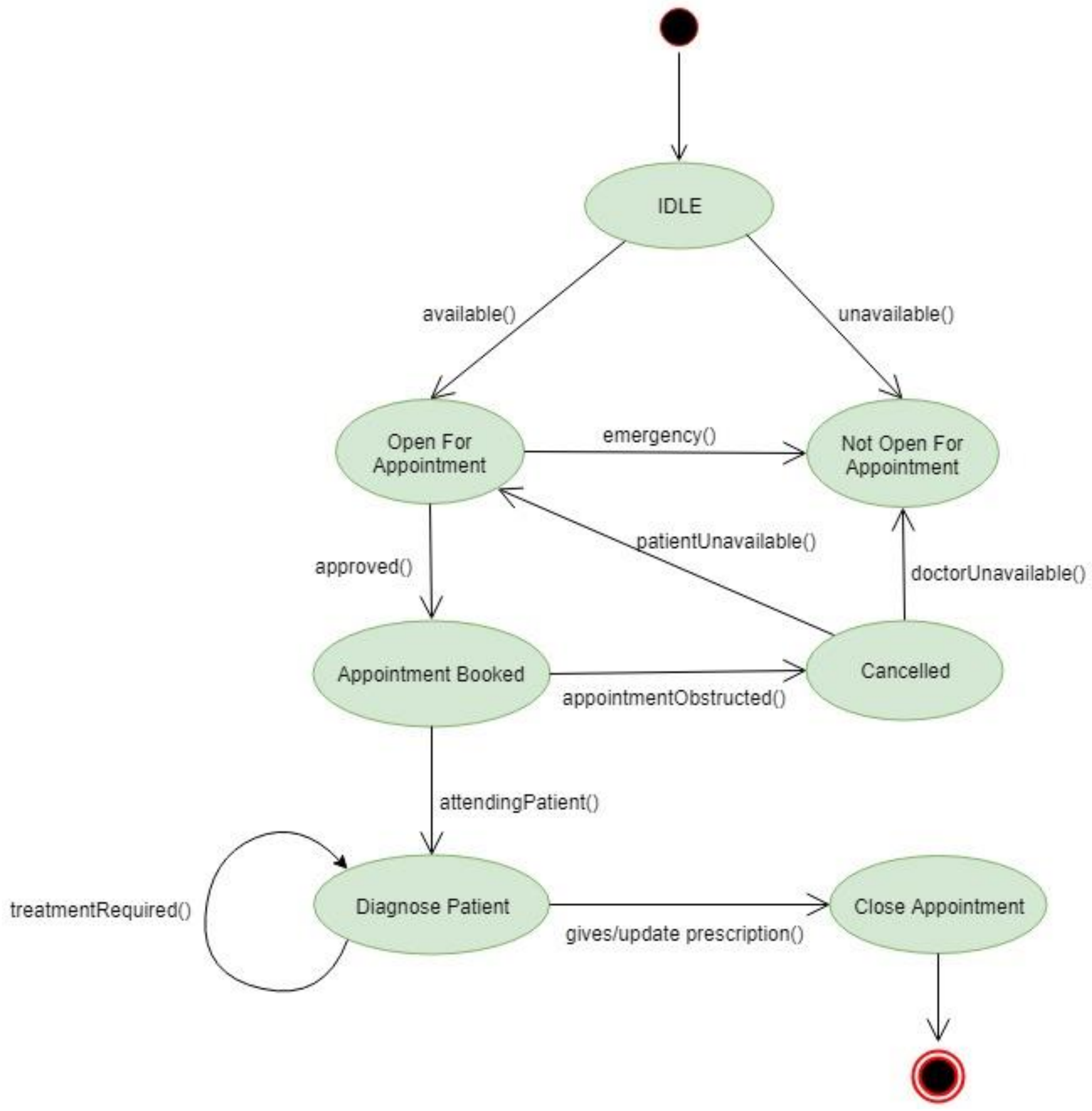

Fig 1.2.1 Doctor State Transition Table

Event↓ State →	Idle	Open For Appointment	Not Open For Appointment	Appointment Booked	Canceled	Diagnose Patient	Close Appointment
available()	**Open For Appointment**						
unavailable()	**Not Open for Appointment**						
emergency()		**Not Open For Appointment**					
approved()		**Appointment Booked**					
patientUnavailable()					**Open For Appointment**		
doctorUnavailable()					**Not Open For Appointment**		
appointmentObstructed()				**Canceled**			
attendingPatient()				**Diagnose Patient**			
treatmentRequired()						**Diagnose Patient**	
gives/update prescription ()						**Close Appointment**	

Fig 1.3 Admin State Diagram

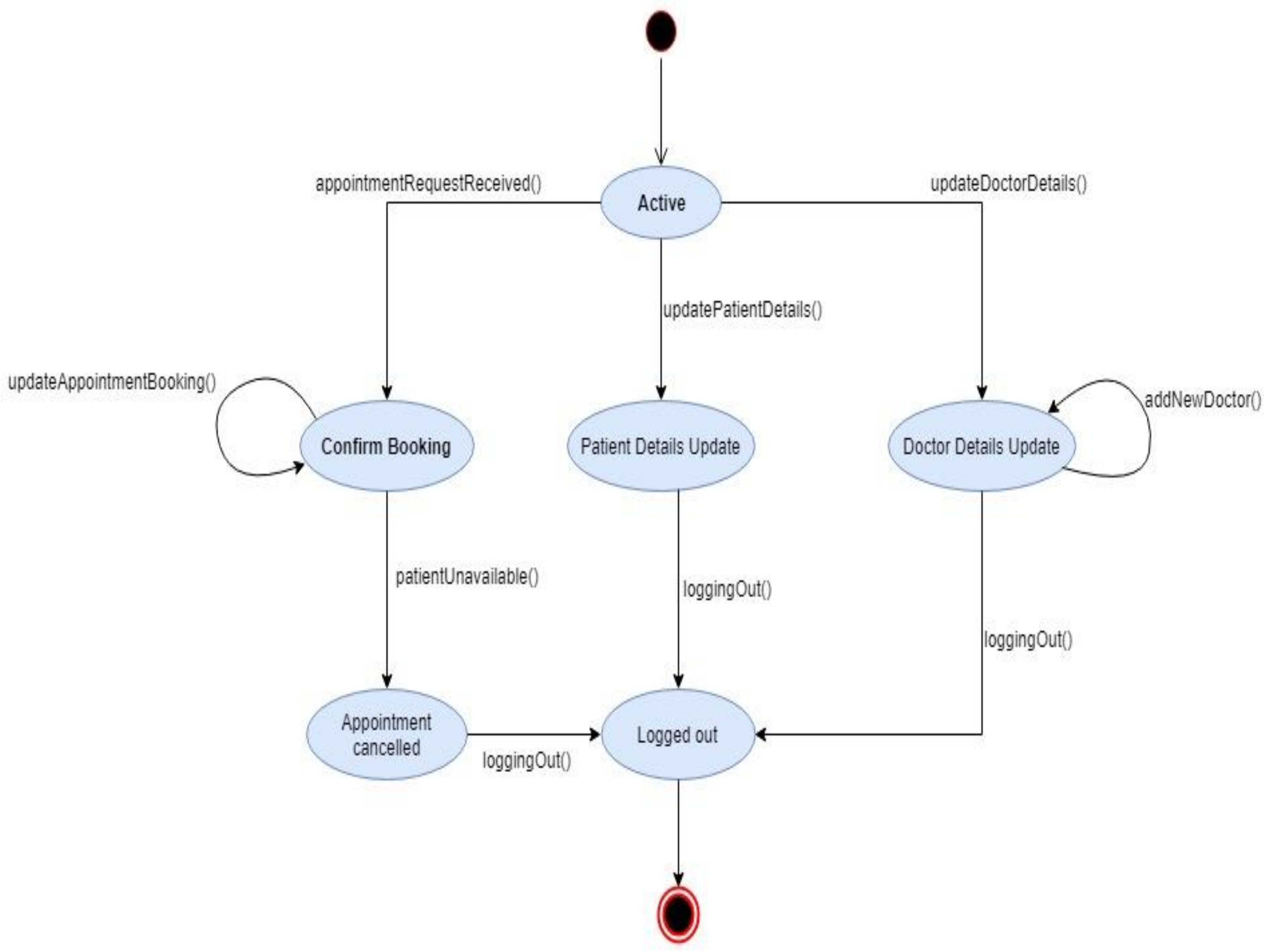

Fig 1.3.1 Admin State Transition Table

Event↓ State →	**Active/Logged In**	**Confirm Booking**	**Doctor Details Updated**	**Patient Details Updated**	**Appoinment Canceled**	**Logged Out**
appointmentRequestReceived()	**Confirm Booking**					
updatePatientProfile()	**Patient Details Updated**					
updateDoctorDetails()	**Doctor Details Updated**					
updateAppointmentBooking()		**Confirm Booking**				
addNewDoctor()			**Doctor Details Updated**			
patientUnavailable()		**Appointment Canceled**				
loggingOut()			**Logged Out**	**Logged Out**	**Logged Out**	

2. Action Specifications:

Action specification refers to specifying the units of executable functionality. We have written the pseudo code for **2 classes(Patient and Doctor)** covering in total **6 operations**.

2.1 In Patient class:
- the patient can add an ailment,
- modify ailment,
- selects a slot for an appointment accordingly,
- books an appointment.

2.2 In Doctor class:
- doctor can set the schedule,
- doctor can update his/her existing schedule.

2.1 Patient Class: -

```
class Patient
     String ailment;
     Appointment appointment;
     Doctor doctor;

     //operation 1
          addAilmentDetails(ailment): String
               if (!ailmentList.contains(ailment)){
                    then ailmentList.add(ailment)
                    return "successful"
               }
               else {
                    return "ailment already exists"
               }


     //operation 2
          addAppointment(appointment, dateTime) : String
               if(appointment.status == "False" &&
                         !dateTime==appointment.Time){

                    then add appointment details

                    return "Successful"
                    }

                else{

                    return “Unsuccessful”
                }

      //operation 3
           modifyAilmentDetails(ailment): Boolean
                if(ailmentList.contains(ailment)){
                    then modify ailment
                    update ailmentList
                    return “True”
                }
                else{
                    addAilmentDetails(ailment)
                    return “True”
                }



      //operation 4
           selectSlotByDoctorId(String doctorId) : String
                HashMap<String, ArrayList<DateTime> schedule> slots;
                if(slots.containsKey(doctorId)){
                    slots.get(doctorId)
                    next available slot booked
                    return DateTime slot;
                }
```

2.2 Doctor Class: -

```
class Doctor
      String doctorId, license, qualification, specialisation;
      Date dateOfJoining;
      DateTime dateT;
      String days;
      int yearsOfExperience;
      Schedule schedule;
      Prescription prescription;
      HashMap<String, ArrayList<Time> time> docSchedule;
      time.add(schedule.timeSlots);
```

```
//operation 5
     setSchedule(doctorId):Boolean

          HashMap<String, Time> timeSlot = docSchedule.get(doctorId)

          timeSlot.put(schedule.availableDays, schedule.timeSlots)

          docSchedule.put(doctorId, slots)

          return “True”




//operation 6

     updateSchedule(String doctorId,String newDate, Time newTime):void

          if(setSchedule()==“True”
               &&  docSchedule.contains(doctorId)){

               HashMap<String, Time> slot = docSchedule.get(doctorId)
               slot.put(newDate, newTime)
               docSchedule.put(doctorId, slot)

          }
```

References

[1]. https://www.tutorialspoint.com/uml/uml_statechart_diagram.htm
[2]. Research on transformation from UML statechart to classical state diagram
[3]. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/