EMERGENCY MANAGEMENT DATABASE

Carmen Areses Sánchez

Paula Blanco González

María del Carmen Cortés Navarro

María Gala González

ÍNDICE

WHY AN EMERGENCY DATA BASE

When talking about a hospital emergency department, an essential aspect of providing quality care and saving lives is the hospital's ability to manage patients and medical resources.

However, the reality for many hospitals when it comes to handling these situations is that when they reach a certain limit of patients, their system is not capable of supporting it challenging the smooth flow of patients and internal management. Patients not knowing where to go, treatment or triage room being occupied yet patients are still being called, doctors not being available in the treatment rooms or even boxes being empty when the hospital is at full capacity. In addition, many times, medical records can get lost or not updated in time causing delays and errors when diagnosing a patient.

In other areas, these problems may just lead to having difficulty managing the hospital but in the emergency department just a tiny mistake can and do lead to failures that can become crucial when it comes to saving lives.

To solve these issues, we have developed a database specifically design to manage the emergency department. This tool, not only organises patient flow based on their urgency, but it also optimizes the internal administration of the hospital guaranteeing that every emergency department works smoothly with one another.

Some interesting aspects of the database is the capability of tracking the patients from their arrival all the way up to the by registering the time of arrival and discharge from each phase, whether is entering triage or even being admitted into the hospital.

In addition, the database allows the hospital administrator to have great control over all emergency departments. The program has great flexibility when it comes to adding or eliminating rooms of any type and even associating doctors with said rooms. This overview helps optimize the entire hospital administration in a simple and fast way.

In summary, this database represents a significant advancement in the management of hospital emergency services, solving critical problems and improving the efficiency and quality of medical care.

HOW DOES OUR SYSTEM DATABASE WORK?

1. The patient enters the hospital and is attended by the receptionist. The receptionist can either search for the patient if it’s not their firs time in the emergency room or create a new patient. In both cases the patient will have:

- Name
- Surname
- Age
- Sex
- Urgency level [0-1]

The receptionist establishes a first urgency level between 0 and 1 based on the severity of the patient case. If the receptionist chooses an urgency of 0, the patient will follow the normal course of the emergency department. However, if they choose a 1, it means maximum urgency and the patient will be treated immediately.

2. The patient then goes to the waiting room until they are called to triage. In triage, the nurse will evaluate the patient more in depth. They will fill in the rest of the patient's information (height and weight), evaluate the patient’s symptoms and will again establish an emergency level on a scale that goes from 1 to 5, with 1 being the highest level and 5 being the lowest level. This selected emergency level will determine the urgency with which the patient will be treated.

Finally, the nurse will determine the specialty to which the patient will be assigned so that he or she can receive care from a specialized doctor.

The patient’s file after triage is performed would look like this:

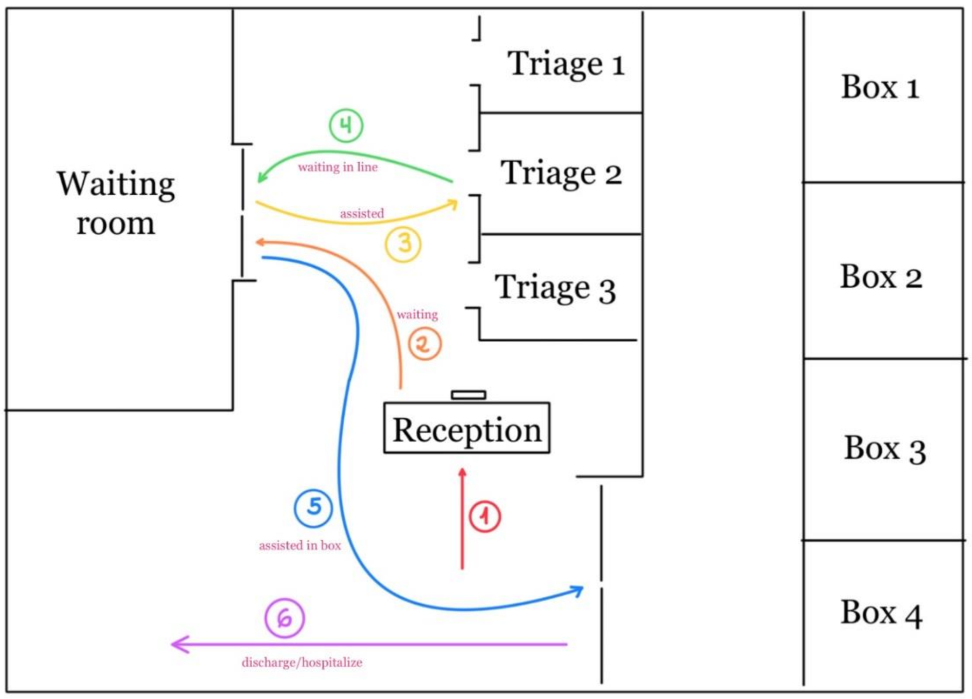
- Name
- Surname
- Age
- Height
- Weight
- Symptoms
- Speciality assigned
- Urgency level [1-5]

The patient then leaves triage and goes back to the waiting room.

3. The patient gets called into a box depending on the specialty that has been assigned in triage, since each box has a specific specialty and associated doctors.

Inside the box, the doctor receives the patient's file and with it he will treat the patient. If they see it convenience they can add some commentaries to the file.

Once the diagnosis is made, the doctor decides whether to discharge the patient or hospitalize them.



USER STORIES

As actors we have:

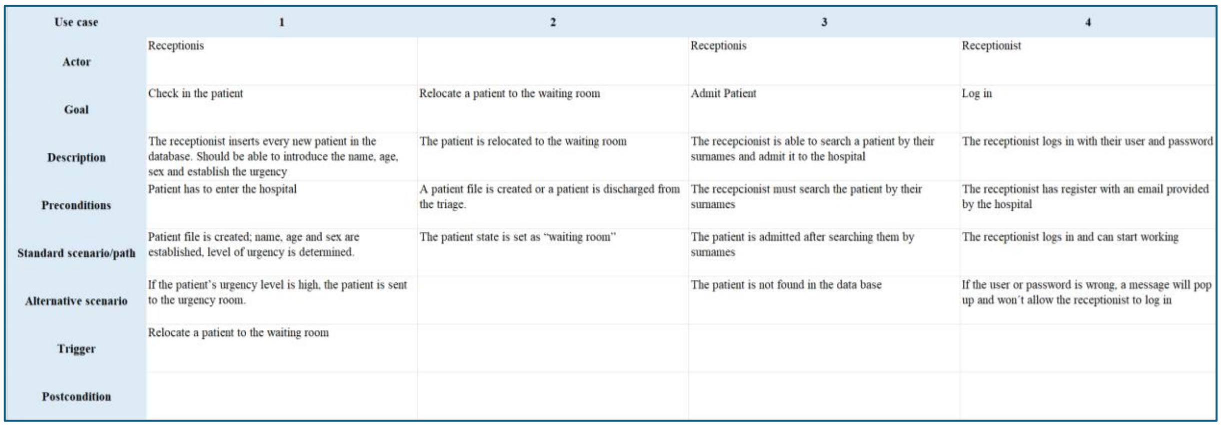
- Receptionist
- Nurse
- Doctor
- Hospital administrator

1. As a receptionist, I want to check in patients to know how many people are in the hospital.
2. As a receptionist, I want to check in patients to derive them to the emergency room if the patients are in high risk.
3. As a receptionist, I want to check in patients to send them to the waiting room until a triage is free.
4. As a receptionist, I want to log in with my username and password in order to start working.
5. As a nurse, I want to evaluate the patient to assign them to a doctor.
6. As a nurse, I want to evaluate the patient to assign the urgency and derive them to the urgency room or the waiting room.
7. As a nurse, I want to log in with my username and password in order to start working.
8. As a doctor, I want to assist the patients in the box, to reach further conclusions and discharge or hospitalize them.
9. As a doctor, I want to discharge patients after assisting them.
10. As a doctor, I want to be assigned patients only in my working hours and when I am not attending anyone else, to properly assist my patients.
11. As a doctor, I want to log in with my username and password in order to start working.
12. As a hospital administrator, I want to administrate which doctor is in the hospital, to assign them to a box.
13. As a hospital administrator, I want to control where each patient is, to…

14. As a hospital administrator, I want to be able to add, modify info and eliminate doctor`s information, to have a control of the doctors working in my department.
15. As a hospital manager, I want to log in with my username and password in order to start working.

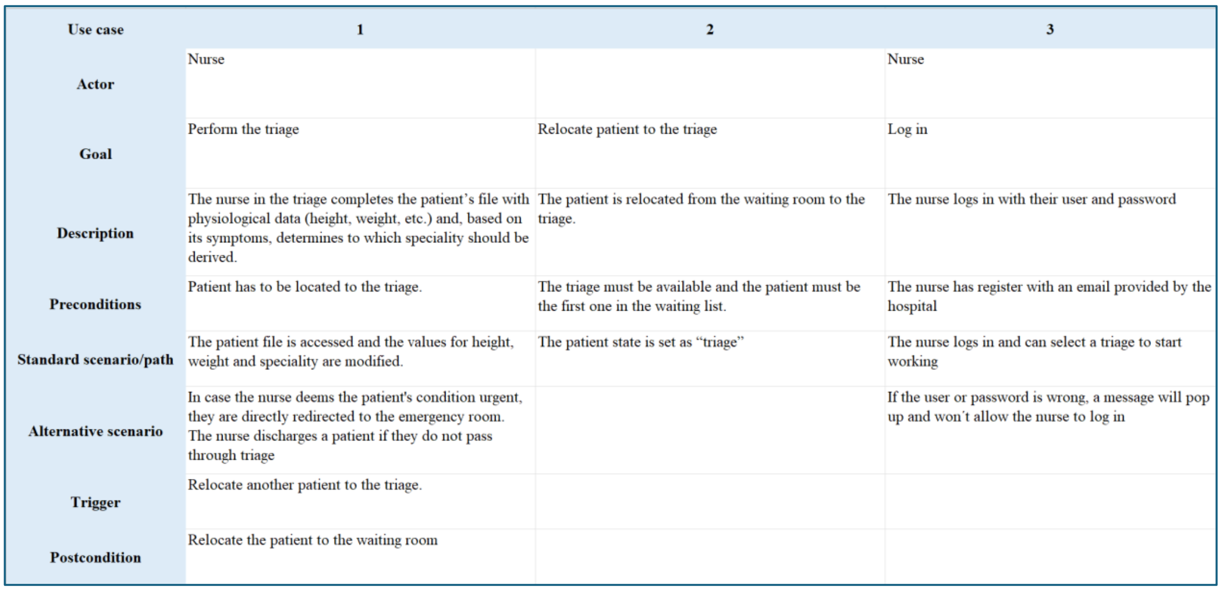
USE CASES

Receptionist:



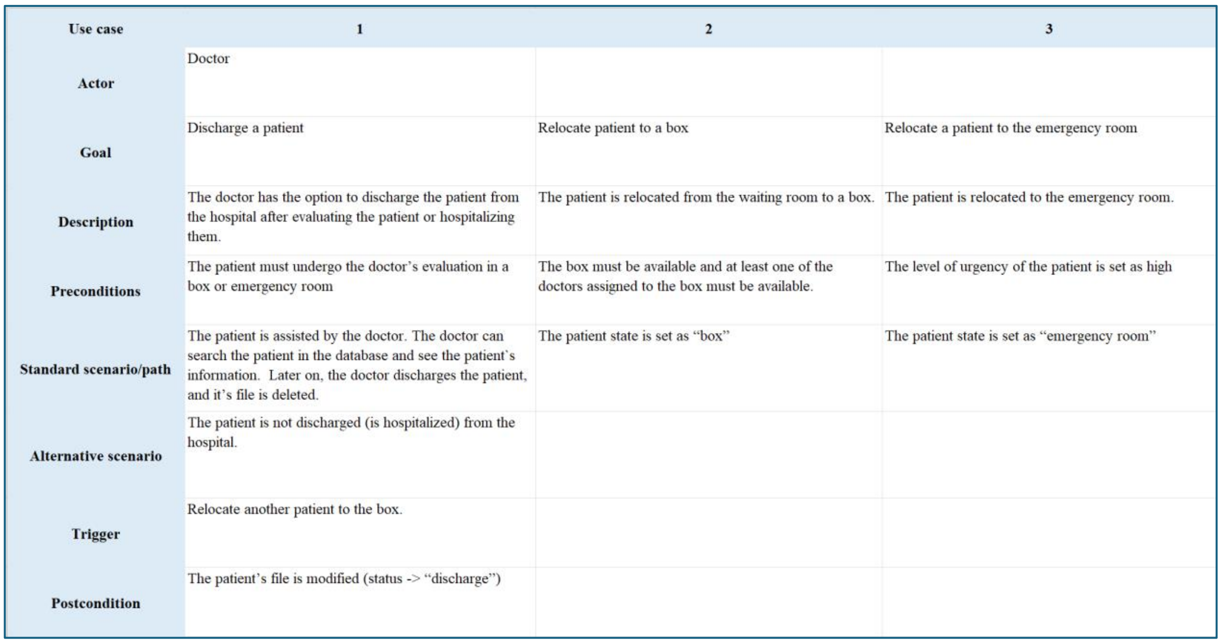
Use case	1	2	3	4
Actor	Receptionis		Receptionis	Receptionist
Goal	Check in the patient	Relocate a patient to the waiting room	Admit Patient	Log in
Description	The receptionist inserts every new patient in the database. Should be able to introduce the name, age, sex and establish the urgency	The patient is relocated to the waiting room	The recepcionist is able to search a patient by their surnames and admit it to the hospital	The receptionist logs in with their user and password
Preconditions	Patient has to enter the hospital	A patient file is created or a patient is discharged from the triage.	The recepcionist must search the patient by their surnames	The receptionist has register with an email provided by the hospital
Standard scenario/path	Patient file is created; name, age and sex are established, level of urgency is determined.	The patient state is set as "waiting room"	The patient is admitted after searching them by surnames	The receptionist logs in and can start working
Alternative scenario	If the patient's urgency level is high, the patient is sent to the urgency room.		The patient is not found in the data base	If the user or password is wrong, a message will pop up and won´t allow the receptionist to log in
Trigger	Relocate a patient to the waiting room			
Postcondition				

Nurse:

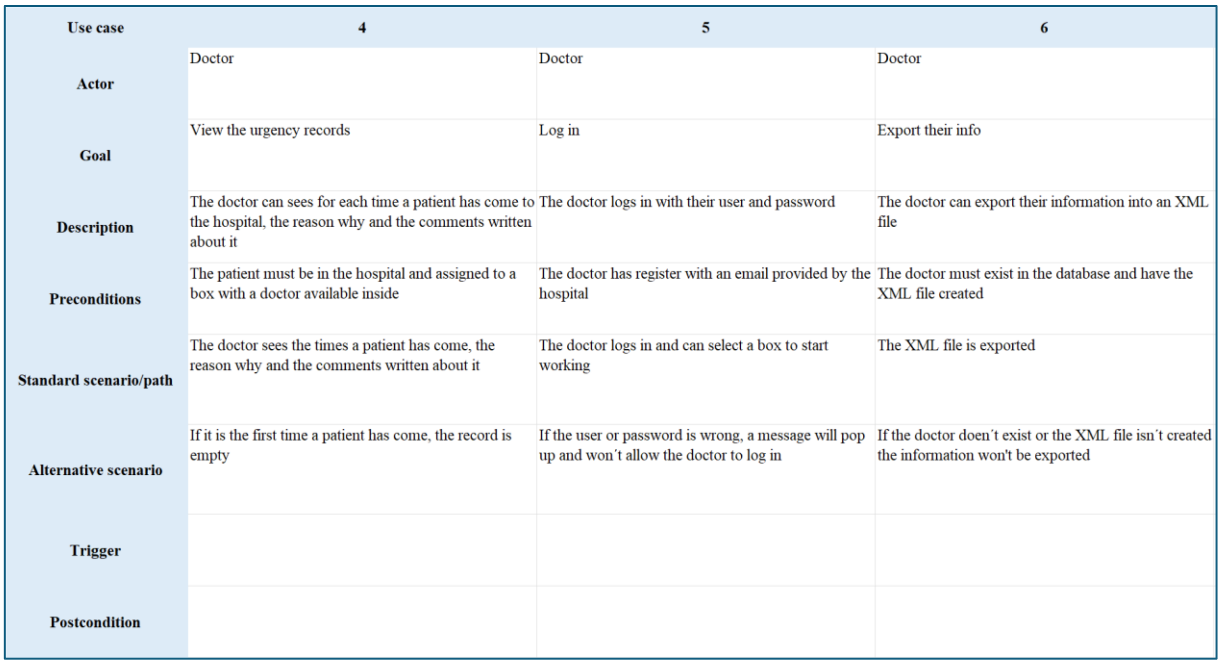


Use case	1	2	3
Actor	Nurse		Nurse
Goal	Perform the triage	Relocate patient to the triage	Log in
Description	The nurse in the triage completes the patient's file with physiological data (height, weight, etc.) and, based on its symptoms, determines to which speciality should be derived.	The patient is relocated from the waiting room to the triage.	The nurse logs in with their user and password
Preconditions	Patient has to be located to the triage.	The triage must be available and the patient must be the first one in the waiting list.	The nurse has register with an email provided by the hospital
Standard scenario/path	The patient file is accessed and the values for height, weight and speciality are modified.	The patient state is set as "triage"	The nurse logs in and can select a triage to start working
Alternative scenario	In case the nurse deems the patient's condition urgent, they are directly redirected to the emergency room. The nurse discharges a patient if they do not pass through triage		If the user or password is wrong, a message will pop up and won´t allow the nurse to log in
Trigger	Relocate another patient to the triage.		
Postcondition	Relocate the patient to the waiting room		

Doctor:

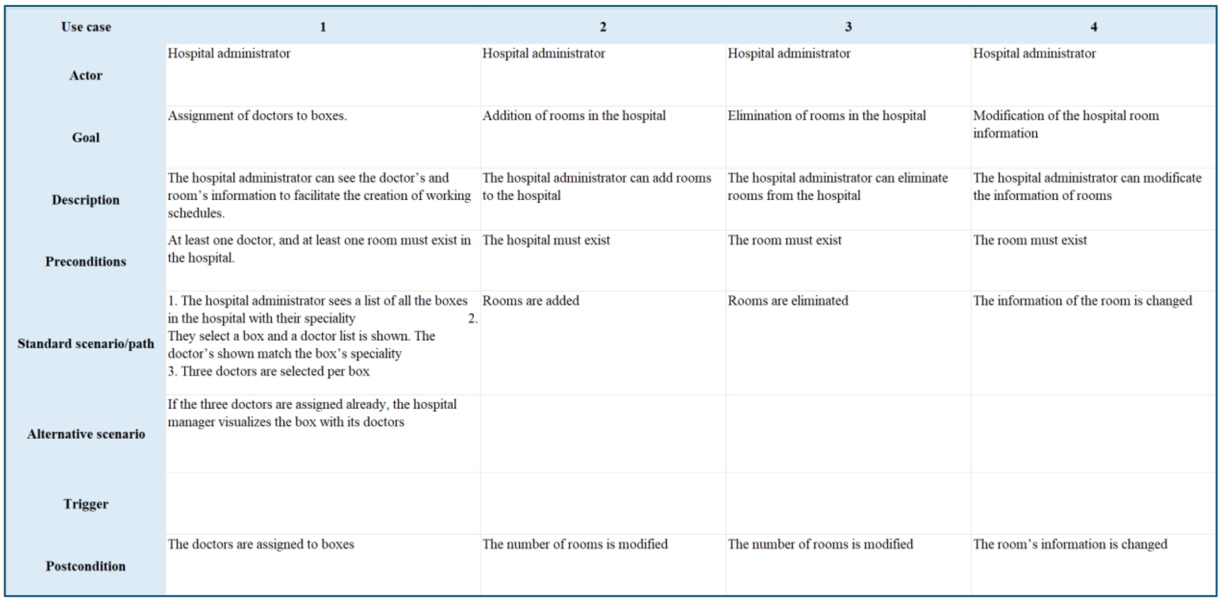


Use case	1	2	3
Actor	Doctor		
Goal	Discharge a patient	Relocate patient to a box	Relocate a patient to the emergency room
Description	The doctor has the option to discharge the patient from the hospital after evaluating the patient or hospitalizing them.	The patient is relocated from the waiting room to a box.	The patient is relocated to the emergency room.
Preconditions	The patient must undergo the doctor's evaluation in a box or emergency room	The box must be available and at least one of the doctors assigned to the box must be available.	The level of urgency of the patient is set as high
Standard scenario/path	The patient is assisted by the doctor. The doctor can search the patient in the database and see the patient`s information. Later on, the doctor discharges the patient, and it's file is deleted.	The patient state is set as "box"	The patient state is set as "emergency room"
Alternative scenario	The patient is not discharged (is hospitalized) from the hospital.		
Trigger	Relocate another patient to the box.		
Postcondition	The patient's file is modified (status -> "discharge")		

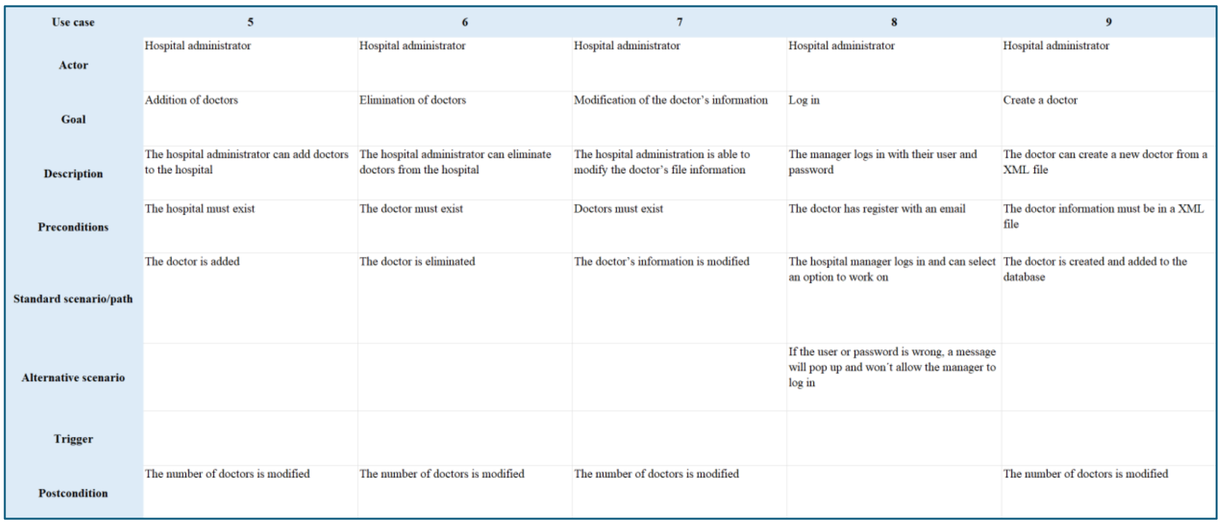


Use case	4	5	6
Actor	Doctor	Doctor	Doctor
Goal	View the urgency records	Log in	Export their info
Description	The doctor can sees for each time a patient has come to the hospital, the reason why and the comments written about it	The doctor logs in with their user and password	The doctor can export their information into an XML file
Preconditions	The patient must be in the hospital and assigned to a box with a doctor available inside	The doctor has register with an email provided by the hospital	The doctor must exist in the database and have the XML file created
Standard scenario/path	The doctor sees the times a patient has come, the reason why and the comments written about it	The doctor logs in and can select a box to start working	The XML file is exported
Alternative scenario	If it is the first time a patient has come, the record is empty	If the user or password is wrong, a message will pop up and won´t allow the doctor to log in	If the doctor doen´t exist or the XML file isn´t created the information won't be exported
Trigger			
Postcondition			

Hospital administrator:

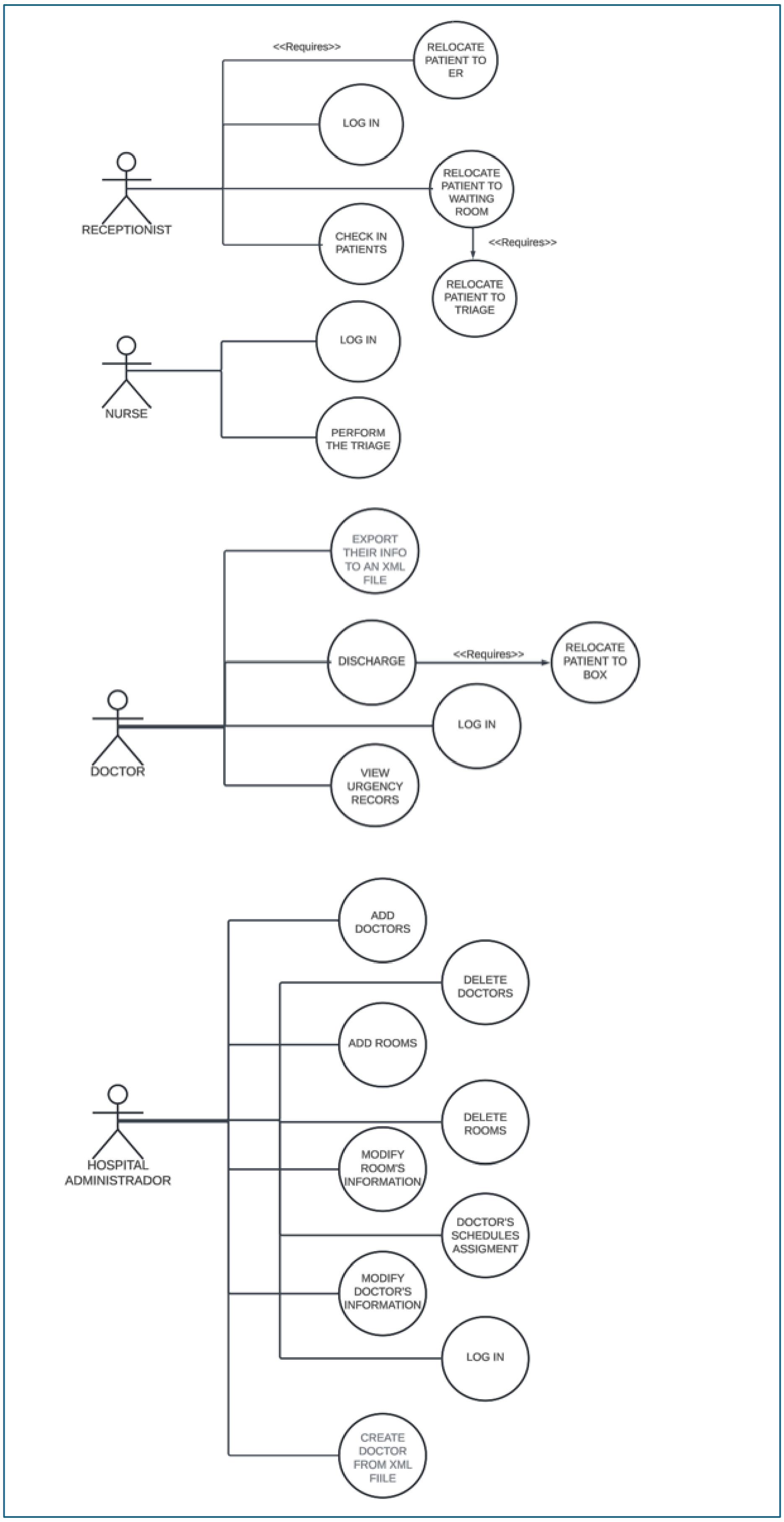


Use case	1	2	3	4
Actor	Hospital administrator	Hospital administrator	Hospital administrator	Hospital administrator
Goal	Assignment of doctors to boxes.	Addition of rooms in the hospital	Elimination of rooms in the hospital	Modification of the hospital room information
Description	The hospital administrator can see the doctor's and room's information to facilitate the creation of working schedules.	The hospital administrator can add rooms to the hospital	The hospital administrator can eliminate rooms from the hospital	The hospital administrator can modificate the information of rooms
Preconditions	At least one doctor, and at least one room must exist in the hospital.	The hospital must exist	The room must exist	The room must exist
Standard scenario/path	1. The hospital administrator sees a list of all the boxes in the hospital with their speciality 2. They select a box and a doctor list is shown. The doctor's shown match the box's speciality 3. Three doctors are selected per box	Rooms are added	Rooms are eliminated	The information of the room is changed
Alternative scenario	If the three doctors are assigned already, the hospital manager visualizes the box with its doctors			
Trigger				
Postcondition	The doctors are assigned to boxes	The number of rooms is modified	The number of rooms is modified	The room's information is changed

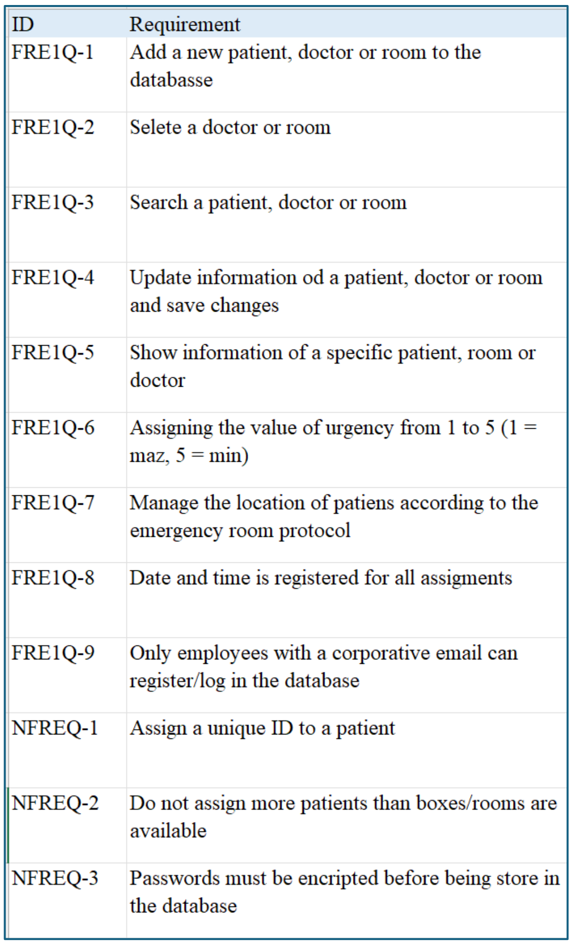


Use case	5	6	7	8	9
Actor	Hospital administrator	Hospital administrator	Hospital administrator	Hospital administrator	Hospital administrator
Goal	Addition of doctors	Elimination of doctors	Modification of the doctor's information	Log in	Create a doctor
Description	The hospital administrator can add doctors to the hospital	The hospital administrator can eliminate doctors from the hospital	The hospital administration is able to modify the doctor's file information	The manager logs in with their user and password	The doctor can create a new doctor from a XML file
Preconditions	The hospital must exist	The doctor must exist	Doctors must exist	The doctor has register with an email	The doctor information must be in a XML file
Standard scenario/path	The doctor is added	The doctor is eliminated	The doctor's information is modified	The hospital manager logs in and can select an option to work on	The doctor is created and added to the database
Alternative scenario				If the user or password is wrong, a message will pop up and won't allow the manager to log in	
Trigger					
Postcondition	The number of doctors is modified	The number of doctors is modified	The number of doctors is modified		The number of doctors is modified

UML USE CASE DIAGRAM

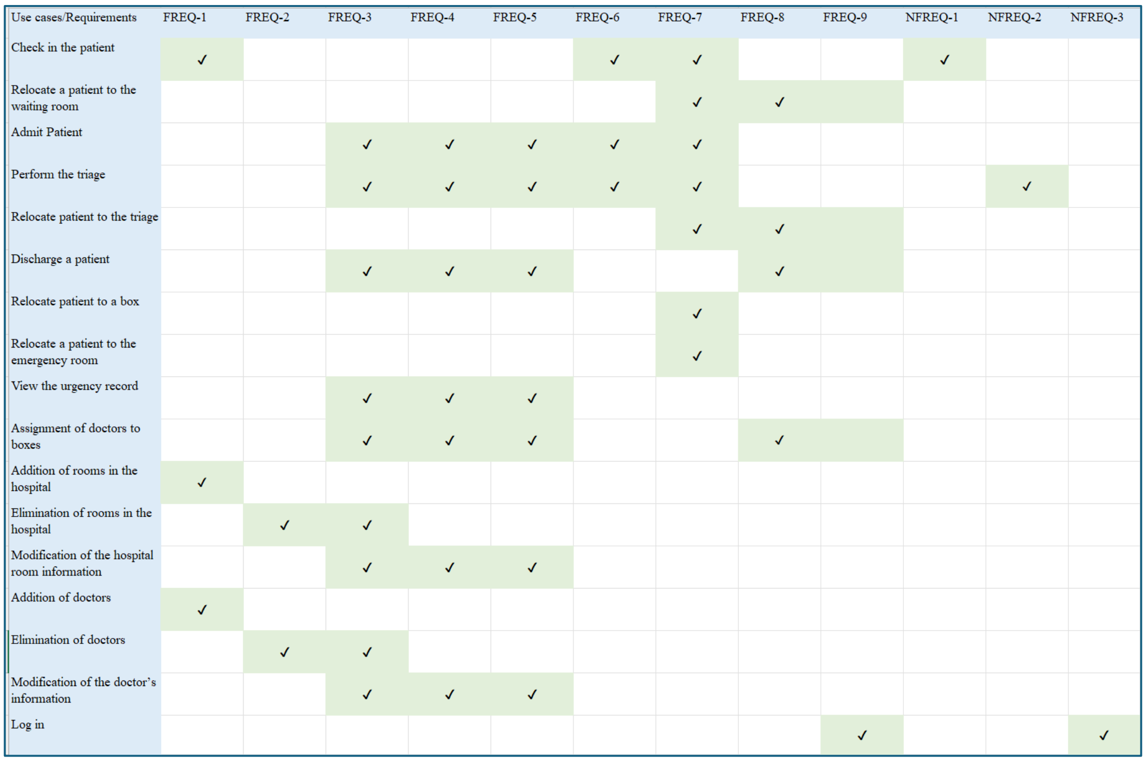


REQUIREMENT’S TABLE



ID	Requirement
FRE1Q-1	Add a new patient, doctor or room to the databasse
FRE1Q-2	Selete a doctor or room
FRE1Q-3	Search a patient, doctor or room
FRE1Q-4	Update information od a patient, doctor or room and save changes
FRE1Q-5	Show information of a specific patient, room or doctor
FRE1Q-6	Assigning the value of urgency from 1 to 5 (1 = maz, 5 = min)
FRE1Q-7	Manage the location of patiens according to the emergency room protocol
FRE1Q-8	Date and time is registered for all assigments
FRE1Q-9	Only employees with a corporative email can register/log in the database
NFREQ-1	Assign a unique ID to a patient
NFREQ-2	Do not assign more patients than boxes/rooms are available
NFREQ-3	Passwords must be encripted before being store in the database

TRACEABILITY MATRIX



Use cases/Requirements	FREQ-1	FREQ-2	FREQ-3	FREQ-4	FREQ-5	FREQ-6	FREQ-7	FREQ-8	FREQ-9	NFREQ-1	NFREQ-2	NFREQ-3
Check in the patient	✓					✓	✓			✓		
Relocate a patient to the waiting room							✓	✓				
Admit Patient			✓	✓	✓	✓	✓					
Perform the triage			✓	✓	✓	✓	✓				✓	
Relocate patient to the triage							✓	✓				
Discharge a patient			✓	✓	✓			✓				
Relocate patient to a box							✓					
Relocate a patient to the emergency room							✓					
View the urgency record			✓	✓	✓							
Assignment of doctors to boxes			✓	✓	✓			✓				
Addition of rooms in the hospital	✓											
Elimination of rooms in the hospital		✓	✓									
Modification of the hospital room information			✓	✓	✓							
Addition of doctors	✓											
Elimination of doctors		✓	✓									
Modification of the doctor's information			✓	✓	✓							
Log in									✓			✓

E-R DIAGRAM

We have designed an E-R diagram that has five entities:

- Patient
- Doctor
- Box
- Triage
- Speciality

Each entity has its own attributes and form different relationships with each other.

Patient-Speciality: because we store medical records, each patient may have more than one specialty assigned over time. On the other hand, each specialty can be assigned to multiple patients.

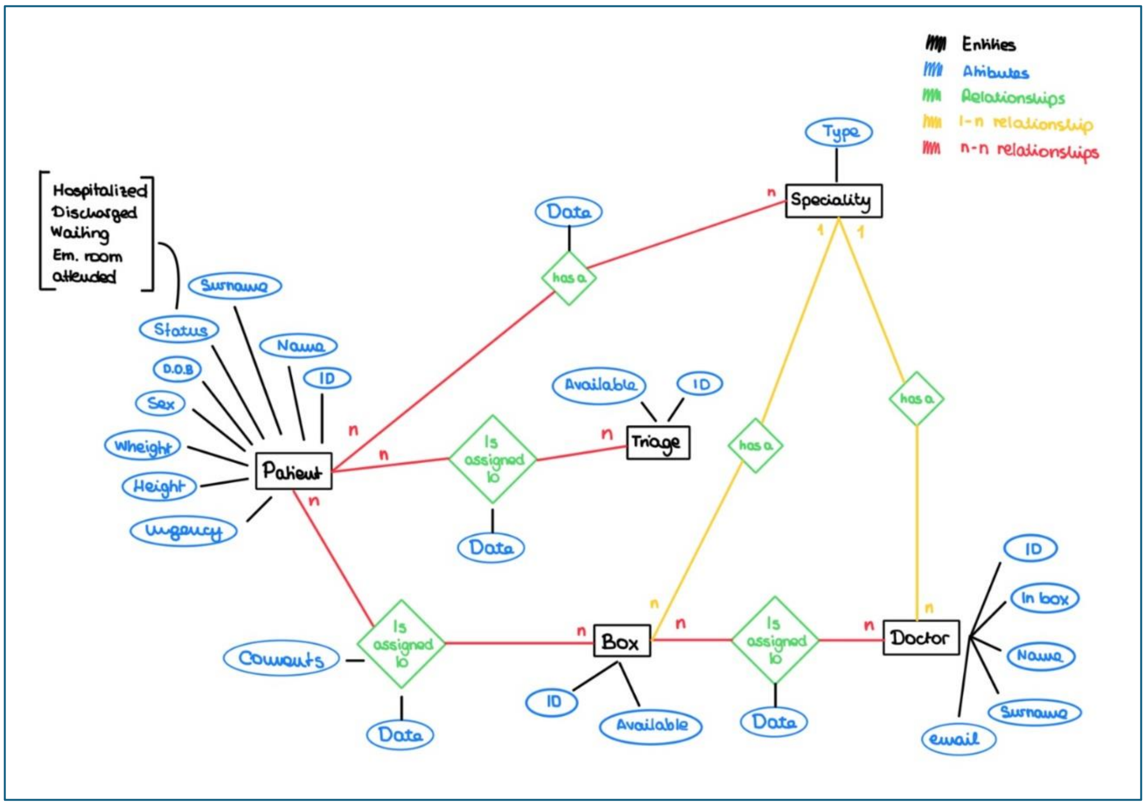
Patient-Triage: a patient may be called to a different triage each time they go to the emergency room and many patients go through one triage.

Patient-Box: a patient may be called to a different box each time they go to the emergency room and many patients go through one box.

Box-Speciality: each box has one speciality assigned, but there can be more than one box with that speciality.

Box-Doctor: each box has three doctors assigned to it. And because there can be more than one with the same speciality, a doctor can be assigned to more than one box over time.

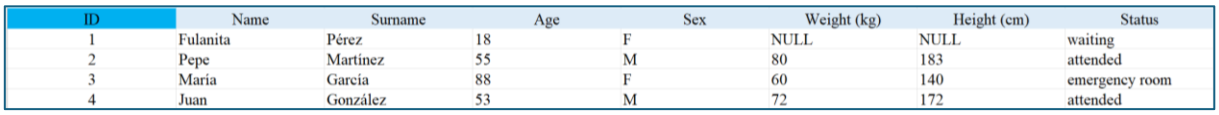
Doctor-Speciality: each doctor has one speciality, but there can be more than one doctor with that speciality.



RELATIONAL DATABASE'S TABLES

- Primary key
- Foreign key

Patient:



ID	Name	Surname	Age	Sex	Weight (kg)	Height (cm)	Status
1	Fulanita	Pérez	18	F	NULL	NULL	waiting
2	Pepe	Martínez	55	M	80	183	attended
3	María	García	88	F	60	140	emergency room
4	Juan	González	53	M	72	172	attended

ID: INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT

Name: TEXT NOT NULL

Age: TEXT NOT NULL

Sex: TEXT NOT NULL **CHECK**(sex = "M" OR sex = "F")

Weight: INTEGER

Height: INTEGER

Status: TEXT NOT NULL **CHECK** (status = "waiting" OR status = "attended" OR status = "emergency room" OR status = "discharged" OR status = "hospitalized")

CREATE TABLE Patients (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, surname TEXT NOT NULL, birthdate DATE NOT NULL, sex TEXT NOT NULL CHECK (sex = 'Man' OR sex ='Woman'), weight INTEGER, height INTEGER, status TEXT

NOT NULL CHECK (status = 'waiting' OR status = 'waitingInLine' OR status = 'assisted' OR status = 'assistedInBox' OR status = 'emergency room' OR status = 'discharged' OR status = 'hospitalized'), urgency INTEGER CHECK (urgency = 1 OR urgency = 2 OR urgency = 3 OR urgency = 4 OR urgency = 5))

Triage:

ID	Available
1	True
2	False

ID: INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT

Available: Boolean NOT NULL

CREATE TABLE Triages (id INTEGER PRIMARY KEY AUTOINCREMENT, available Boolean NOT NULL)

Box:

ID	Available	Speciality_type
1	Ture	Cardio
2	False	Trauma

ID: INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT

Available: Boolean NOT NULL

Speciality_type: TEXT REFERENCES Speciality(Type) ON DELETE RESTRICT

CREATE TABLE Boxes (id INTEGER PRIMARY KEY AUTOINCREMENT, available Boolean NOT NULL, speciality_type TEXT NOT NULL REFERENCES Specialities(type) ON DELETE RESTRICT)

Doctor:

ID	Name	Surname	Speciality_type
1	Jose	Paredes	Traumatology
2	Maria Paz	Cruz	Radiology
3	Damián	Ballesteros	General

ID: INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT

Name: TEXT NOT NULL

Speciality_type: TEXT NOT NULL REFERENCES Speciality (Type) ON DELETE RESTRICT

If the doctor does not own a speciality, we set this value into default value (ex. urgency) as this person works in a general box.

CREATE TABLE Doctors (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, surname TEXT NOT NULL, email TEXT NOT NULL, speciality_type TEXT NOT NULL REFERENCES Specialities(type) ON DELETE RESTRICT, in_box Boolean NOT NULL)

Speciality:

Type
Cardiology
Traumatology
Radiology
General

Type: TEXT NOT NULL PRIMARY KEY

CREATE TABLE Specialities (type TEXT PRIMARY KEY)

Patient-Speciality (n-n table):

Patient_id	Speciality_type	Date

Patient_ID: INTEGER NOT NULL REFERENCES Patient(ID) ON DELETE CASCADE

Speciality_type: TEXT NOT NULL REFERENCES Speciality(Type) ON DELETE RESTRICT

Date: DATE NOT NULL

PRIMARY KEY (Patient_ID, Speciality_Type)

CREATE TABLE PatientSpeciality (patient_id INTEGER REFERENCES Patients(id) ON DELETE CASCADE, speciality_type TEXT REFERENCES Specialities(type) ON DELETE RESTRICT, date DATETIME NOT NULL, PRIMARY KEY (patient_id, speciality_type, date))

Patient-Triage (n-n table):

Patient_id	Triage_id	Time of arrival	Time of discharge

Patient_ID: INTEGER NOT NULL REFERENCES Patient(ID) ON DELETE CASCADE

Triage_ID: INTEGER NOT NULL REFERENCES Triage(ID) ON DELETE SET NULL

Time of arrival: DATE NOT NULL

Time of discharge: DATE NOT NULL

PRIMARY KEY (Patient_ID, Triage_ID)

CREATE TABLE PatientTriage (patient_id INTEGER REFERENCES Patients(id) ON DELETE CASCADE, triage_id INTEGER REFERENCES Triages(id) ON DELETE SET NULL, date DATETIME NOT NULL, PRIMARY KEY (patient_id, triage_id, date))

Patient-Box (n-n table):

Patient_id	Box_id	Time of arrival	Time of discharge	Comments

Patient_ID: INTEGER NOT NULL REFERENCES Patient(ID) ON DELETE CASCADE

Box_ID: TEXT NOT NULL REFERENCES Box(ID) ON DELETE SET NULL

Time of arrival: DATE NOT NULL

Time of discharge: DATE NOT NULL

Comments: TEXT NOT NULL

PRIMARY KEY (Patient_ID, Box_ID)

CREATE TABLE PatientBox (patient_id INTEGER REFERENCES Patients(id) ON DELETE CASCADE, box_id INTEGER REFERENCES Boxes(id) ON DELETE SET NULL, date DATETIME NOT NULL, comments TEXT, PRIMARY KEY (patient_id, box_id, date))

Box-Doctor (n-n table):

Box_id	Doctor_id	Date

Box_ID: INTEGER NULL REFERENCES Box(ID) ON DELETE CASCADE

Doctor_ID: INTEGER NOT NULL REFERENCES Doctor(ID) ON DELETE SET NULL

Date: DATE NOT NULL

PRIMARY KEY: (Box_ID, Doctor_ID)

CREATE TABLE BoxDoctor (box_id INTEGER REFERENCES Boxes(id) ON DELETE CASCADE, doctor_id INTEGER REFERENCES Doctors(id) ON DELETE SET NULL, date DATETIME NOT NULL, PRIMARY KEY (box_id, doctor_id, date))

Apart from the tables created from the E-R diagram, we have created three extra tables to use in other methods:

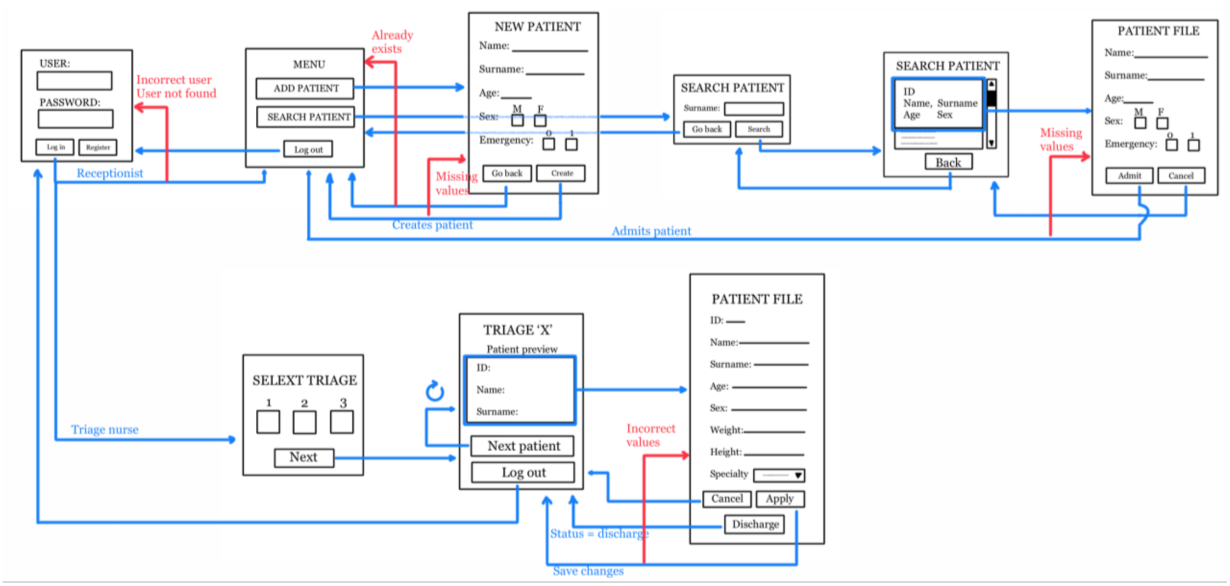
Roles

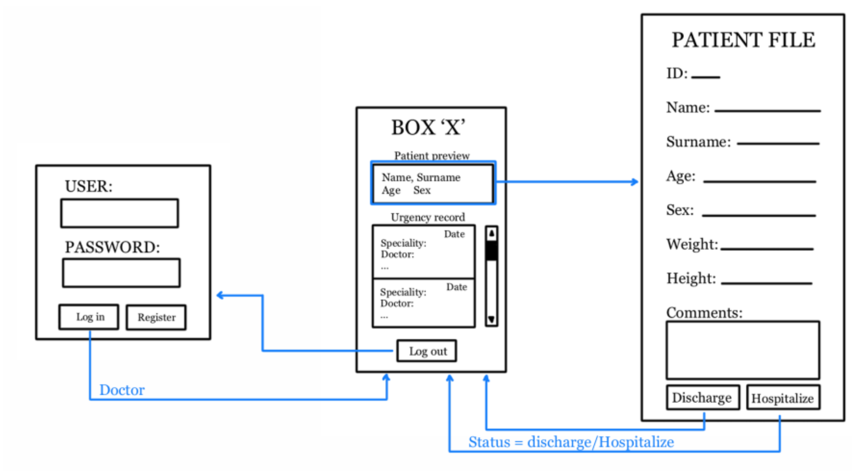
CREATE TABLE roles (ID NUMBER(10) NOT NULL, roleName VARCHAR, PRIMARY KEY (ID))

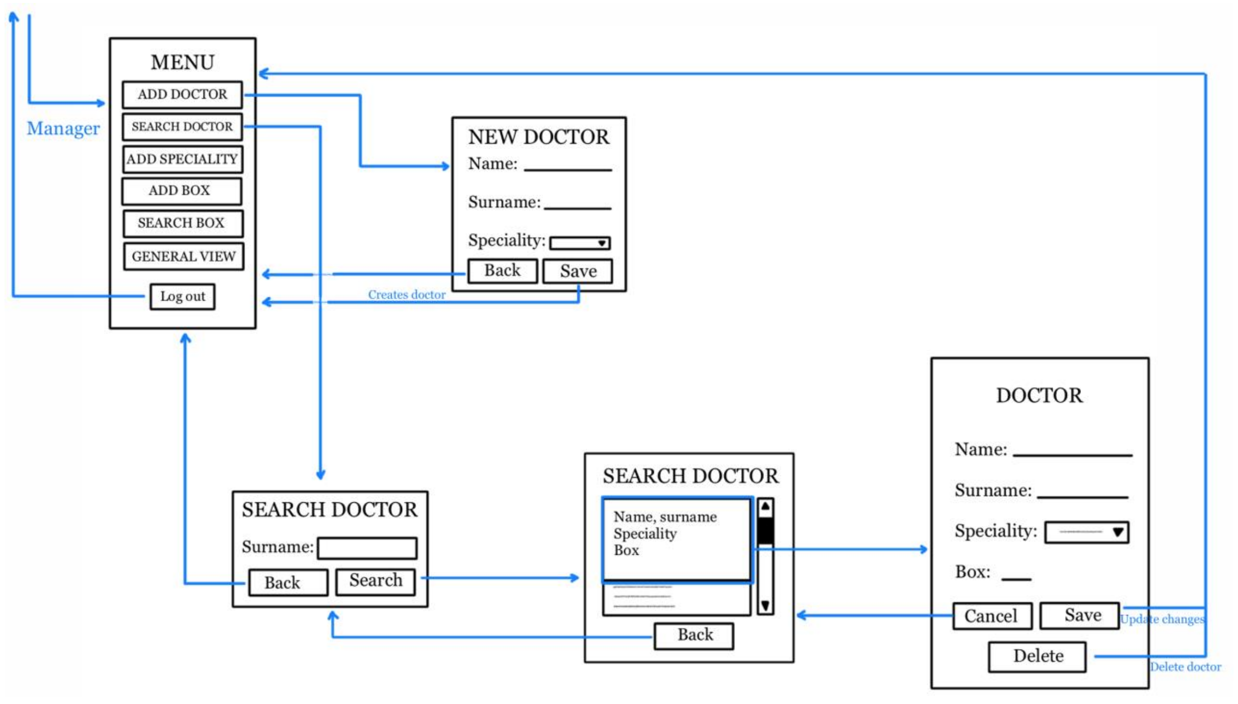
Users

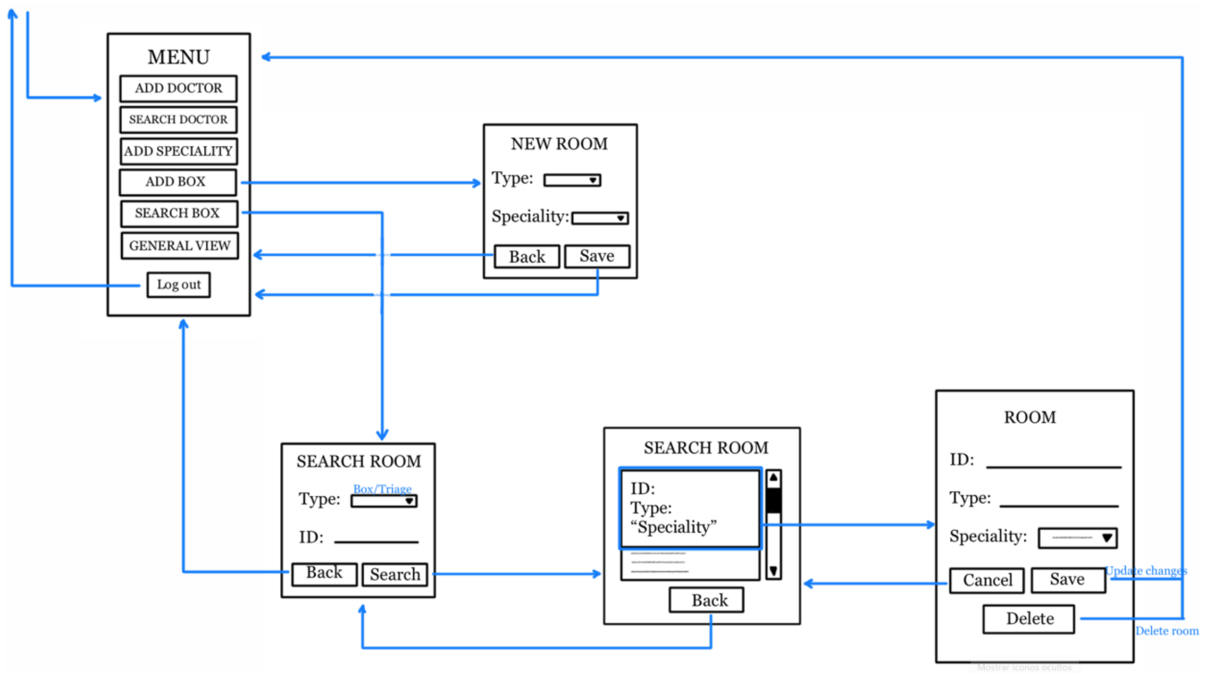
CREATE TABLE users (ID NUMBER(10) NOT NULL, email VARCHAR NOT NULL UNIQUE, password VARCHAR, ROLE_ID NUMBER(10), PRIMARY KEY (ID))

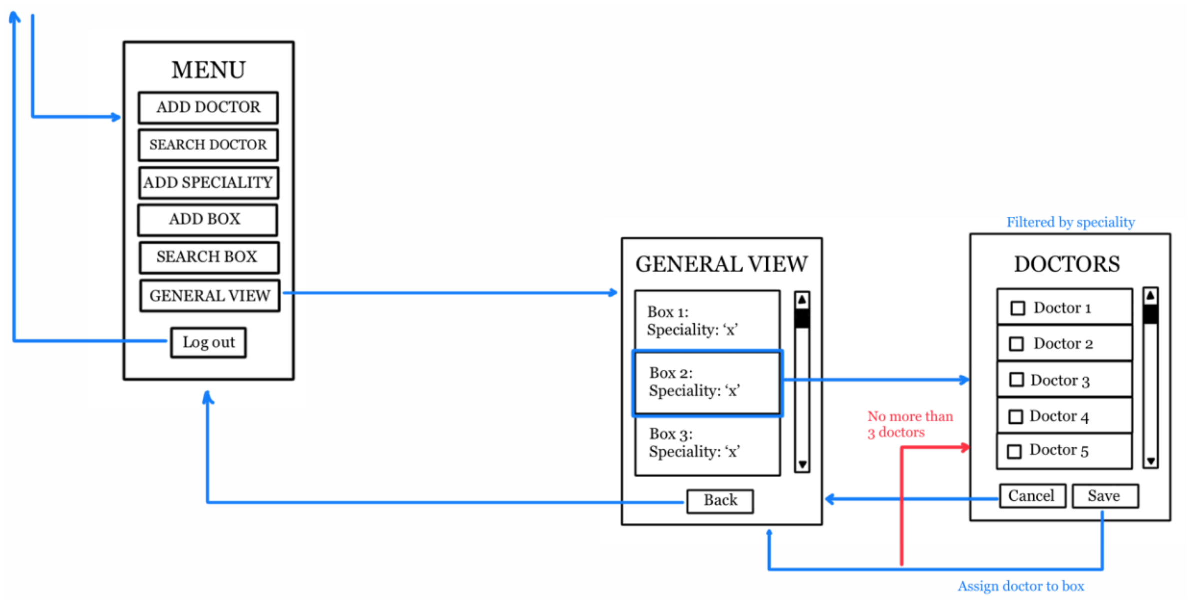
MOCK-UP


USER:
PASSWORD:
Log in
Register
Incorrect user
User not found
Receptionist
MENU
ADD PATIENT
SEARCH PATIENT
Log out
Already exists
NEW PATIENT
Name:
Surname:
Age:
Sex: M F
Emergency: 0 1
Go back
Create
Missing values
Creates patient
SEARCH PATIENT
Surname:
Go back
Search
SEARCH PATIENT
ID
Name, Surname
Age Sex
Back
PATIENT FILE
Name:
Surname:
Age:
Sex: M F
Emergency: 0 1
Admit
Cancel
Missing values
Admits patient
Triage nurse
SELEXT TRIAGE
1 2 3
Next
TRIAGE 'X'
Patient preview
ID:
Name:
Surname:
Next patient
Log out
PATIENT FILE
ID:
Name:
Surname:
Age:
Sex:
Weight:
Height:
Specialty
Cancel
Apply
Discharge
Incorrect values
Status = discharge
Save changes


USER:
PASSWORD:
Log in
Register
Doctor
BOX 'X'
Patient preview
Name, Surname
Age Sex
Urgency record
Speciality:
Doctor:
Date
Log out
PATIENT FILE
ID:
Name:
Surname:
Age:
Sex:
Weight:
Height:
Comments:
Discharge
Hospitalize
Status = discharge/Hospitalize


Manager
MENU
ADD DOCTOR
SEARCH DOCTOR
ADD SPECIALITY
ADD BOX
SEARCH BOX
GENERAL VIEW
Log out
NEW DOCTOR
Name:
Surname:
Speciality:
Back
Save
Creates doctor
SEARCH DOCTOR
Surname:
Back
Search
SEARCH DOCTOR
Name, surname
Speciality
Box
Back
DOCTOR
Name:
Surname:
Speciality:
Box:
Cancel
Save
Update changes
Delete
Delete doctor

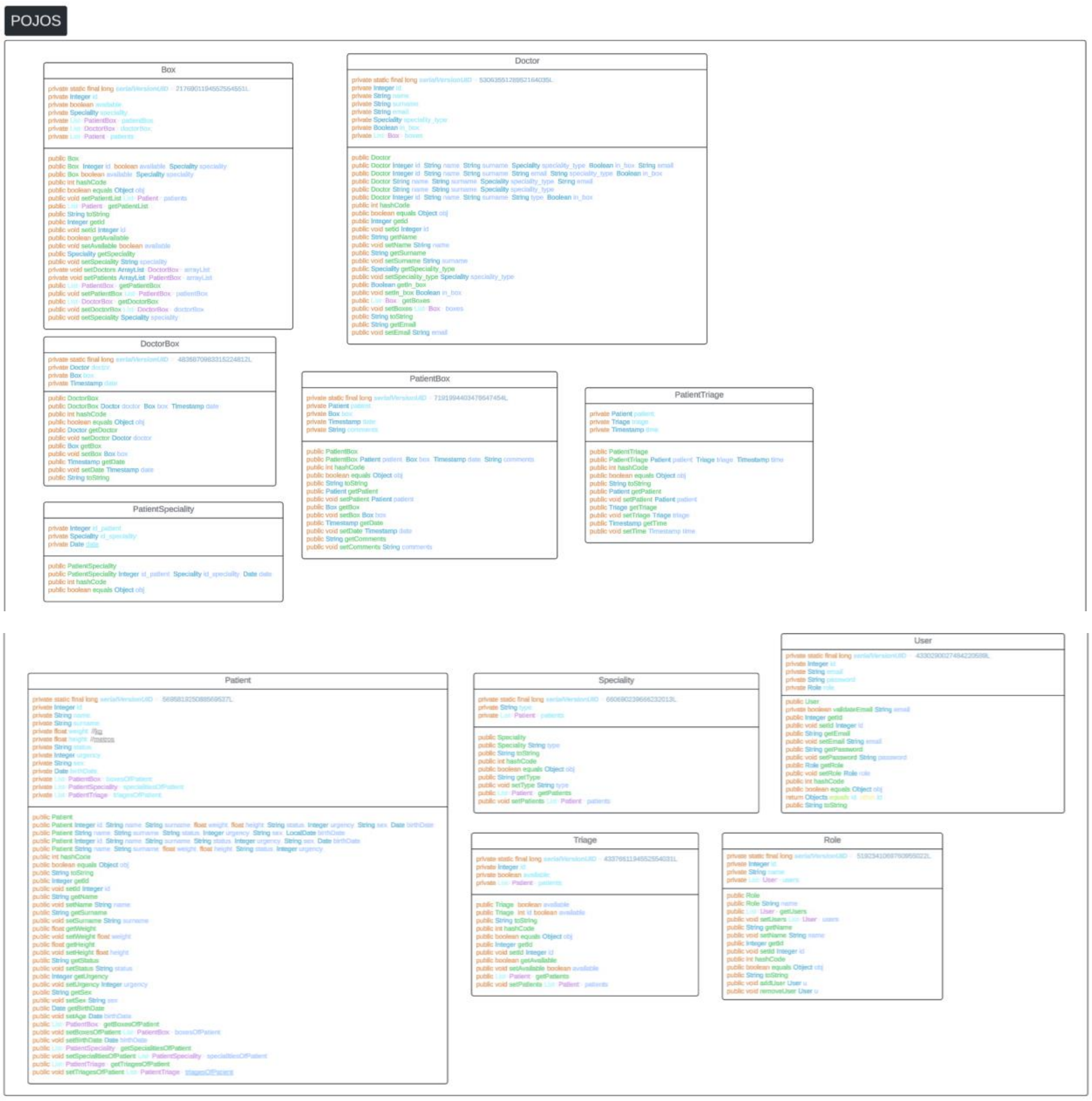

MENU
ADD DOCTOR
SEARCH DOCTOR
ADD SPECIALITY
ADD BOX
SEARCH BOX
GENERAL VIEW
Log out
NEW ROOM
Type:
Speciality:
Back
Save
SEARCH ROOM
Box/Triage
Type:
ID:
Back
Search
SEARCH ROOM
ID:
Type:
“Speciality”
Back
ROOM
ID:
Type:
Speciality:
Cancel
Save
Update changes
Delete
Delete room



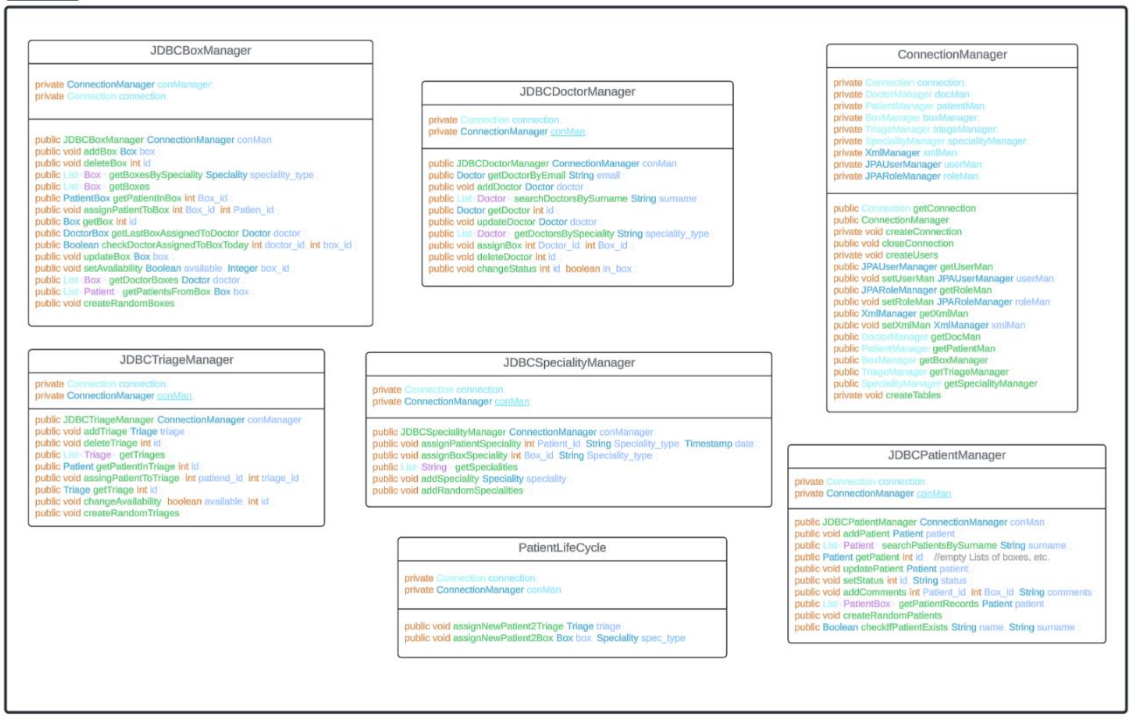
XML DML

```
<!--The type of the attribute "id" of Employee needs
to be CDATA instead of ID, since DTD doesn't allow
IDs that start with a number-->

<!ELEMENT Patient (id, name, surname, weight, height, status, urgency, sex, birthDate)>
<!ATTLIST Patient
id CDATA #REQUIRED
weight CDATA #IMPLIED
height CDATA #IMPLIED
urgency CDATA #IMPLIED
>
<!ELEMENT Doctor (id, name, surname, email, speciality, in_box, box)>
<!ATTLIST Doctor
id CDATA #REQUIRED
in_box CDATA >

<!ELEMENT Box (id, available, speciality, patients)>
<!ATTLIST Box
id CDATA #REQUIRED
available CDATA>

<!ELEMENT name (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT birthDate (#PCDATA)>

<!ELEMENT speciality (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT patients (Patient+)>
<!ELEMENT box (Box+)>
]>
```

UML CLASS DIAGRAM



JDBC



```
JDBCBoxManager
private ConnectionManager conManager
private Connection connection

public JDBCBoxManager ConnectionManager conMan
public void addBox Box box
public void deleteBox int id
public List<Box> getBoxesBySpeciality Speciality speciality_type
public List<Box> getBoxes
public PatientBox getPatientInBox int Box_id
public void assignPatientToBox int Box_id int Patien_id
public Box getBox int id
public DoctorBox getLastBoxAssignedToDoctor Doctor doctor
public Boolean checkDoctorAssignedToBoxToday int doctor_id int box_id
public void updateBox Box box
public void setAvailability Boolean available Integer box_id
public List<Box> getDoctorBoxes Doctor doctor
public List<Patient> getPatientsFromBox Box box
public void createRandomBoxes
```

```
JDBCDoctorManager
private Connection connection
private ConnectionManager conMan

public JDBCDoctorManager ConnectionManager conMan
public Doctor getDoctorByEmail String email
public void addDoctor Doctor doctor
public List<Doctor> searchDoctorsBySurname String surname
public Doctor getDoctor int id
public void updateDoctor Doctor doctor
public List<Doctor> getDoctorsBySpeciality String speciality_type
public void assignBox int Doctor_id int Box_id
public void deleteDoctor int id
public void changeStatus int id boolean in_box
```

```
ConnectionManager
private Connection connection;
private DoctorManager docMan;
private PatientManager patientMan;
private BoxManager boxManager;
private TriageManager triageManager;
private SpecialityManager specialityManager;
private XmlManager xmlMan;
private JPAUserManager userMan;
private JPARoleManager roleMan;

public Connection getConnection
public ConnectionManager
private void createConnection
public void closeConnection
private void createUsers
public JPAUserManager getUserMan
public void setUserMan JPAUserManager userMan
public JPARoleManager getRoleMan
public void setRoleMan JPARoleManager roleMan
public XmlManager getXmlMan
public void setXmlMan XmlManager xmlMan
public DoctorManager getDocMan
public PatientManager getPatientMan
public BoxManager getBoxManager
public TriageManager getTriageManager
public SpecialityManager getSpecialityManager
private void createTables
```

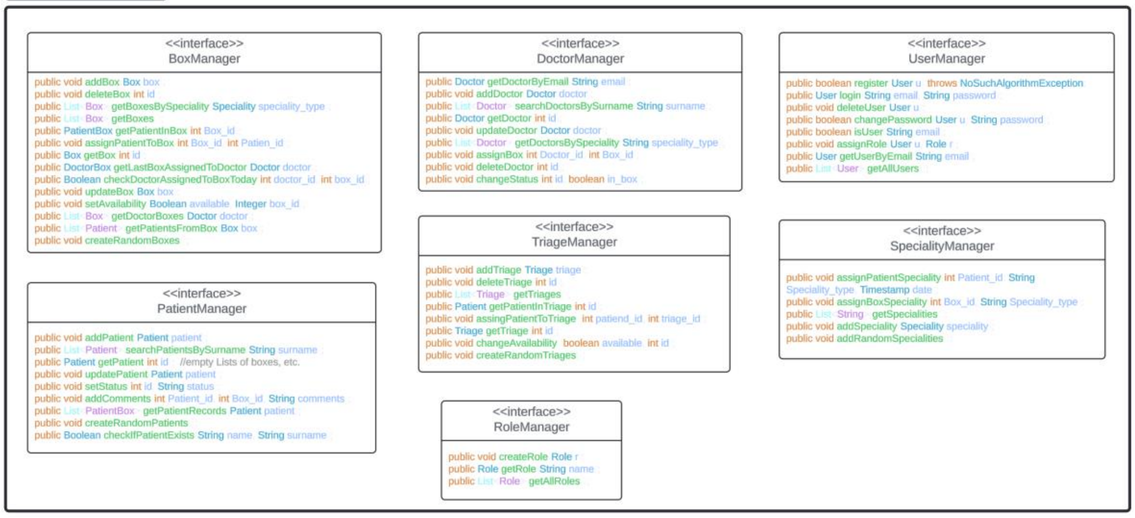
```
JDBCTriageManager
private Connection connection
private ConnectionManager conMan

public JDBCTriageManager ConnectionManager conManager
public void addTriage Triage triage
public void deleteTriage int id
public List<Triage> getTriages
public Patient getPatientInTriage int id
public void assingPatientToTriage int patiend_id int triage_id
public Triage getTriage int id
public void changeAvailability boolean available int id
public void createRandomTriages
```

```
JDBCSpecialityManager
private Connection connection
private ConnectionManager conMan

public JDBCSpecialityManager ConnectionManager conManager
public void assignPatientSpeciality int Patient_id String Speciality_type Timestamp date
public void assignBoxSpeciality int Box_id String Speciality_type
public List<String> getSpecialities
public void addSpeciality Speciality speciality
public void addRandomSpecialities
```

```
PatientLifeCycle
private Connection connection;
private ConnectionManager conMan

public void assignNewPatient2Triage Triage triage
public void assignNewPatient2Box Box box Speciality spec_type
```

```
JDBCPatientManager
private Connection connection
private ConnectionManager conMan

public JDBCPatientManager ConnectionManager conMan
public void addPatient Patient patient
public List<Patient> searchPatientsBySurname String surname
public Patient getPatient int id    //empty Lists of boxes, etc.
public void updatePatient Patient patient
public void setStatus int id String status
public void addComments int Patient_id int Box_id String comments
public List<PatientBox> getPatientRecords Patient patient
public void createRandomPatients
public Boolean checkIfPatientExists String name String surname
```

INTERFACES



```
<<interface>>
BoxManager
public void addBox Box box
public void deleteBox int id
public List<Box> getBoxesBySpeciality Speciality speciality_type
public List<Box> getBoxes
public PatientBox getPatientInBox int Box_id
public void assignPatientToBox int Box_id int Patien_id
public Box getBox int id
public DoctorBox getLastBoxAssignedToDoctor Doctor doctor
public Boolean checkDoctorAssignedToBoxToday int doctor_id int box_id
public void updateBox Box box
public void setAvailability Boolean available Integer box_id
public List<Box> getDoctorBoxes Doctor doctor
public List<Patient> getPatientsFromBox Box box
public void createRandomBoxes
```

```
<<interface>>
DoctorManager
public Doctor getDoctorByEmail String email
public void addDoctor Doctor doctor
public List<Doctor> searchDoctorsBySurname String surname
public Doctor getDoctor int id
public void updateDoctor Doctor doctor
public List<Doctor> getDoctorsBySpeciality String speciality_type
public void assignBox int Doctor_id int Box_id
public void deleteDoctor int id
public void changeStatus int id boolean in_box
```

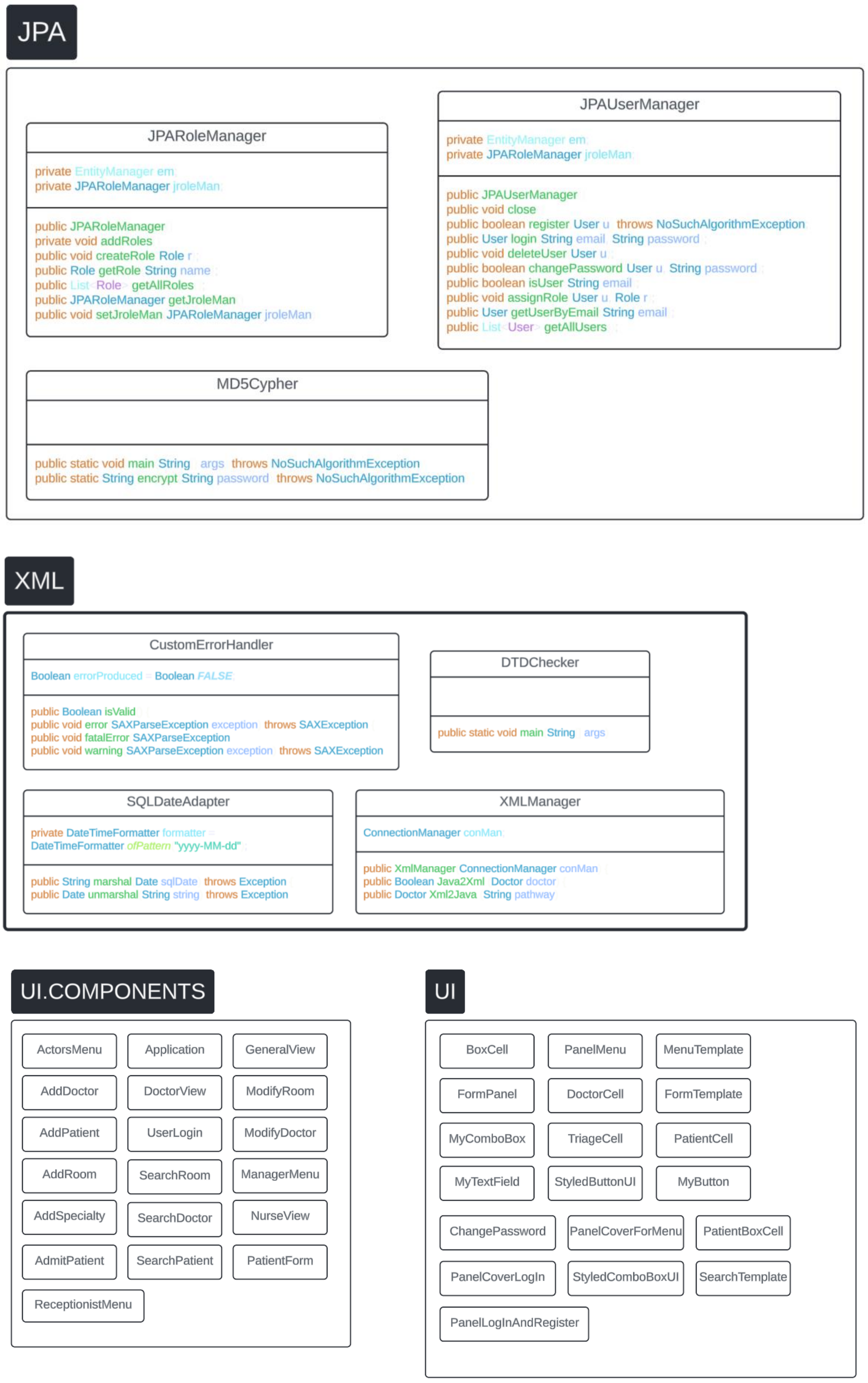
```
<<interface>>
UserManager
public boolean register User u throws NoSuchAlgorithmException
public User login String email String password
public void deleteUser User u
public boolean changePassword User u String password
public boolean isUser String email
public void assignRole User u Role r
public User getUserByEmail String email
public List<User> getAllUsers
```

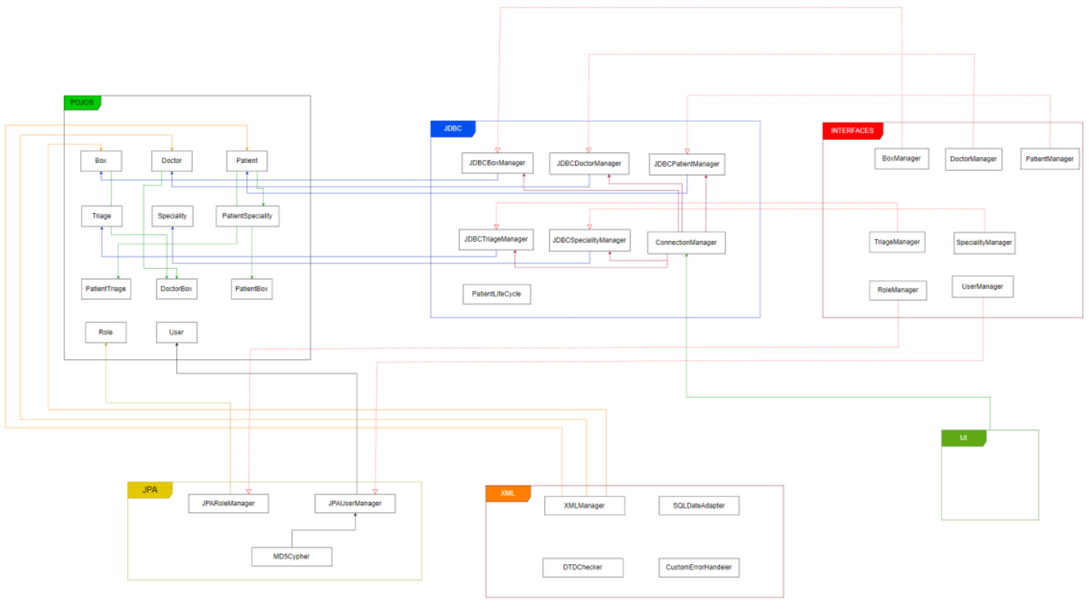
```
<<interface>>
TriageManager
public void addTriage Triage triage
public void deleteTriage int id
public List<Triage> getTriages
public Patient getPatientInTriage int id
public void assingPatientToTriage int patiend_id int triage_id
public Triage getTriage int id
public void changeAvailability boolean available int id
public void createRandomTriages
```

```
<<interface>>
SpecialityManager
public void assignPatientSpeciality int Patient_id String
Speciality_type Timestamp date
public void assignBoxSpeciality int Box_id String Speciality_type
public List<String> getSpecialities
public void addSpeciality Speciality speciality
public void addRandomSpecialities
```

```
<<interface>>
PatientManager
public void addPatient Patient patient
public List<Patient> searchPatientsBySurname String surname
public Patient getPatient int id    //empty Lists of boxes, etc.
public void updatePatient Patient patient
public void setStatus int id String status
public void addComments int Patient_id int Box_id String comments
public List<PatientBox> getPatientRecords Patient patient
public void createRandomPatients
public Boolean checkIfPatientExists String name String surname
```

```
<<interface>>
RoleManager
public void createRole Role r
public Role getRole String name
public List<Role> getAllRoles
```


JPA
JPARoleManager
private EntityManager em;
private JPARoleManager jroleMan;
public JPARoleManager()
private void addRoles()
public void createRole(Role r);
public Role getRole(String name);
public List<Role> getAllRoles();
public JPARoleManager getJroleMan()
public void setJroleMan(JPARoleManager jroleMan)
JPAUserManager
private EntityManager em;
private JPARoleManager jroleMan;
public JPAUserManager()
public void close()
public boolean register(User u) throws NoSuchAlgorithmException
public User login(String email, String password);
public void deleteUser(User u);
public boolean changePassword(User u, String password);
public boolean isUser(String email);
public void assignRole(User u, Role r);
public User getUserByEmail(String email);
public List<User> getAllUsers();
MD5Cypher
public static void main(String[] args) throws NoSuchAlgorithmException
public static String encrypt(String password) throws NoSuchAlgorithmException
XML
CustomErrorHandler
Boolean errorProduced = Boolean.FALSE;
public Boolean isValid() {
public void error(SAXParseException exception) throws SAXException {
public void fatalError(SAXParseException
public void warning(SAXParseException exception) throws SAXException
DTDChecker
public static void main(String[] args)
SQLDateAdapter
private DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd");
public String marshal(Date sqlDate) throws Exception {
public Date unmarshal(String string) throws Exception
XMLManager
ConnectionManager conMan;
public XmlManager(ConnectionManager conMan) {
public Boolean Java2Xml (Doctor doctor) {
public Doctor Xml2Java (String pathway)
UI.COMPONENTS
ActorsMenu
Application
GeneralView
AddDoctor
DoctorView
ModifyRoom
AddPatient
UserLogin
ModifyDoctor
AddRoom
SearchRoom
ManagerMenu
AddSpecialty
SearchDoctor
NurseView
AdmitPatient
SearchPatient
PatientForm
ReceptionistMenu
UI
BoxCell
PanelMenu
MenuTemplate
FormPanel
DoctorCell
FormTemplate
MyComboBox
TriageCell
PatientCell
MyTextField
StyledButtonUI
MyButton
ChangePassword
PanelCoverForMenu
PatientBoxCell
PanelCoverLogIn
StyledComboBoxUI
SearchTemplate
PanelLogInAndRegister


POJOS
Box
Doctor
Patient
Triage
Speciality
PatientSpeciality
PatientTriage
DoctorBox
PatientBox
Role
User
JDBC
JDBCBoxManager
JDBCDoctorManager
JDBCPatientManager
JDBCTriageManager
JDBCSpecialityManager
ConnectionManager
PatientLifeCycle
INTERFACES
BoxManager
DoctorManager
PatientManager
TriageManager
SpecialityManager
RoleManager
UserManager
UI
JPA
JPARoleManager
JPAUserManager
MD5Cypher
XML
XMLManager
SQLDateAdapter
DTDChecker
CustomErrorHandeler