ALLERGY DETECTION DATABASE

Index:

- **The group composition and the project's topic.**

This is the project of the group 07 that is composed by:

Carlo Bottini Morcillo, Carmen Caballero Herreros, Noelia Aubá Arribas and Pablo Hita Lara.

The topic of this project is to create a detection allergy database.

This project seeks to improve the communication between patients and doctors with a program that controls, manages and diagnoses allergies. To be able to do this, the doctor will be able to add patients, delete and modify patient's information from the database so the doctor can treat easily the patient. Also, the doctor can control about allergies, symptoms and treatments in the database and assign them to a specific patient. Then, the patient can see this updated information in his account.

- **The project´s feature, the requirement´s table, the use case diagram and the traceability matrix.**

This project seeks to improve the communication between patients and doctors with a program that controls, manages and diagnoses allergies. To be able to do this, the doctor will be able to add patients, delete and modify patient's information from the database so the doctor can treat easily the patient. Also, the doctor can control about allergies, symptoms and treatments in the database and assign them to a specific patient. Then, the patient can see this updated information in his account.

USES CASES LIST:

- 1) As a doctor, I want to add a new patient to save it in the data base.
- 2) As a doctor, I want to delete a patient to delete unnecessary information.
- 3) As a doctor, I want to modify patient´s information to get it updated.
- 4) As a doctor, I want to add new patient's symptoms.
- 5) As a doctor, I want to add new patient's allergies.
- 6) As a doctor, I want to diagnose the patient's treatment
- 7) As a doctor, I want to create a prescription to treat the patient.
- 8) As a doctor, I want to see the medical score of a patient and modify it.
- 9) As a doctor, I want to upload an XML.
- 10) As a doctor, I want to download an XML.
- 11) As a patient, I want to check my medical score

- 12) As a patient, I want to see the prescription for my treatment

USES CASES:

(1)

Name: Add a new patient

Actor: Doctor

Goal: Doctor adds a new patient on the database.

Preconditions:

- There is a doctor
- There is a new patient that want to be registered
- The new patient is not registered.

Standard path: The Doctor add the new patient's personal information on the Data Base.

Alternative path: The patient is already registered.

The name and surname of the patient are equals to another.

Trigger: The patient wants to be registered in the data base

Postconditions: The patient information is registered in the Data Base.

(2)

Name: Delete a patient

Actor: Doctor

Goal: Doctor can erase the information of a patient

Preconditions:

- The patient is on the Data Base.
- The patient is not anymore available. (dead or changed company)

Standard path: The Doctor deletes the patient information on the Data Base.

Alternative path: The patient doesn't exist.

Trigger: The patient is dead.

Postconditions: The patient information is not anymore in the Data Base.

(3)

Name: Modify patient information

Actor: Doctor

Goal: Doctor can change the information of a patient in the Data Base

Preconditions:

- The patient is already registered on the Data Base.

Standard path: The Doctor modify the patient information on the Data Base.

Alternative path: There is no information about the patient.

Trigger: A modification must be done.

Postconditions: The patient information has been changed.

(4)

Name: To add the patient's symptoms.

Actor: Doctor.

Goal: to add a new symptom to the patient.

Preconditions:

- The patient exists.
- There is a doctor.
- The patient has symptoms that already exist in the data base.

Standard path: The Doctor add the different symptoms in the data base.

Alternative path:

- The patient doesn't have symptoms.
- The symptom is repeated.

Trigger: The patient has symptoms .

Postconditions: The symptoms are added in the data base.

(5)

Name: To add the patient's allergies

Actor: Doctor

Goal: to add a new allergy to the patient.

Preconditions:

- The patient exists.
- There is a doctor that already exist in the data base.

Standard path: The Doctor add the different allergy in the data base.

Alternative path:

- The allergy is repeated.

Trigger: The patient has an allergy.

Postconditions: The allergies are added in the data base.

(6)

Name: Diagnose patient's treatment

Actor: Doctor

Goal: The doctor can give a treatment to the patient

Preconditions:

- The patient exists
- There is a doctor
- The patient has a symptom
- The patient has an allergy
- The patient needs treatment

Standard path: The Doctor gives the prescription to the patient

Alternative path:

- The patient doesn't have symptoms
- The patient does not need a treatment

Trigger: The patient needs treatment

Postconditions: The patient gets its prescription for the treatment.

(7)

Name: Create prescription

Actor: Doctor

Goal: create a prescription for the patient

Preconditions:

- The patient exists
- There is a doctor
- The patient has a symptom
- The patient has an allergy
- The patient needs treatment

Standard path: The Doctor gives the prescription to the patient

Alternative path:

- The patient doesn't have symptoms
- The patient does not need a treatment

Trigger: The patient needs treatment

Postconditions: The patient gets its prescription.

(8)

Name: Update medical score

Actor: Doctor

Goal: Doctor sees the medical score and if necessary, he modifies it.

Preconditions:

- There is a doctor
- There is a patient registered

Standard path: The Doctor sees the medical score of a patient and can delete symptoms or allergies related.

Alternative path:

- The patient is not registered.

Trigger: The doctor wants to check the medical score of a patient

Postconditions: The doctor has updated the medical score of the patient

(9)

Name: Upload XML

Actor: Doctor

Goal: Doctor uploads an XML.

Preconditions:

- There is a doctor or there is a patient registered

Standard path: The Doctor select between a patient or himself and upload the information

Alternative path:

- The patient or doctor is not registered.

Trigger: The doctor wants to upload information

Postconditions: The information is uploaded.

(10)

Name: Download XML

Actor: Doctor

Goal: Doctor downloads an XML

Preconditions:

- There is a database created with information

Standard path: The Doctor selects the option to download the information

Alternative path:

- There is no information to download.

Trigger: The doctor wants to download an XML

Postconditions: The information of the database is downloaded as XML.

(11)

Name: See the patient's Medical Score.

Actor: Patient.

Goal: The patient can visualize him/her medical score.

Preconditions:

- The patient exists.
- There is a doctor.
- There is a medical score.

Standard path: The patient can see him/her medical score.

Alternative path:

- Symptoms are not saved.
- Allergies are not detected.

Trigger: The patient needs to see the medical score.

Postconditions: The patient sees him/her medical score.

(12)

Name: Show the patient's prescription

Actor: Patient

Goal: The Patient can see the different prescriptions that has

Preconditions:

- The patient had already a prescription
- The patient had already got a treatment.

Standard path: The Patient can see all the prescriptions, the used and not used too.

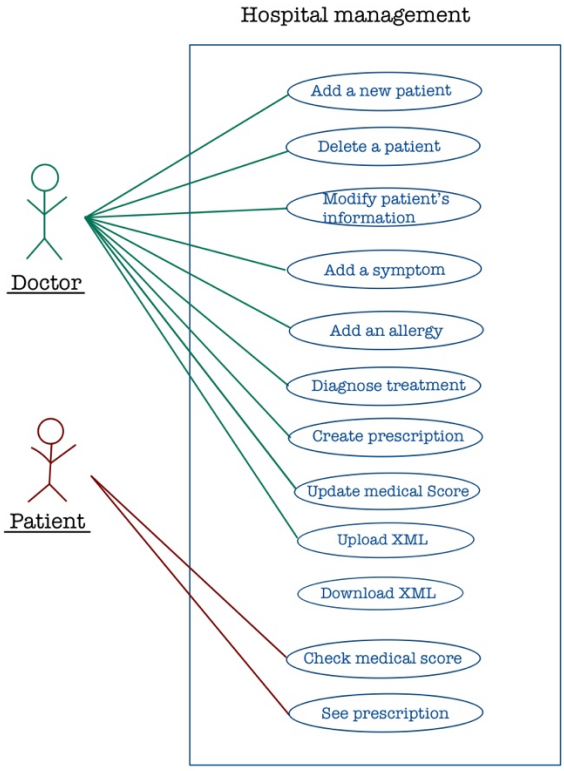
Alternative path:

- There is no prescription
- There is no treatment available.

Trigger: The patients want to see the prescription

Postconditions: The patient sees his prescription with all the data

USE CASE DIAGRAM:



TRACEABILITY MATRIX :

USES CASES:

1.- Delete a patient:

2.- Modify patient:

3.-Add a patient:

4.- Add new allergy:

5.-Diagnose patient's treatment:

6.- Add a new symptom:

7.- Check medical score:

8.-See the prescription:

9.-Create the prescription:

10.-Download XML:

11.-Upload XML:

12.-See medical score:

REQUIREMENT´S LIST:

Functional:

→ The doctor have access to the system. (**F1**)

→The patient have access to the system. (**F2**)

→ The doctor can create a prescription. (**F3**)

→ The patient can see the prescription. (**F4**)

→ The doctor can delete a patient. (**F5**)

→ The doctor can add a patient. (**F6**)

→ The doctor can modify a patient. (**F7**)

→ The doctor can search a patient. (**F8**)

→ The doctor add the symptoms. (F9)

→ The doctor assigns a treatment to allergies. (**F10**)

Non-Functional:

→ The system will complete the commands in less than 2 seconds. (**NF1**)

→ The system is programmed in Java. (**NF2**)

→ The same allergy cannot be registered twice. (**NF3**)

→ There are no symptoms. (**NF4**)

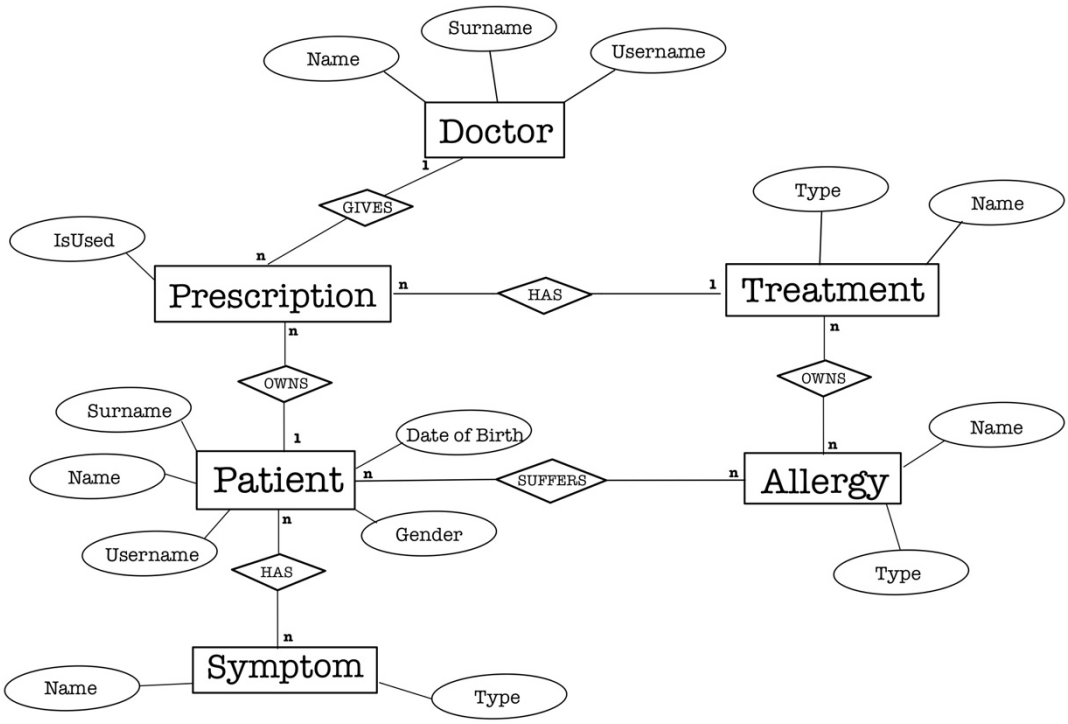
→ The same patient cannot be registered twice. (**NF5**)

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	NF1	NF2	NF3	NF4	NF5
Delete a patient	x				x			x			x				x
Modify a patient	x						x	x			x				x
Add a Patient	x					x					x				x
Add new allergy	x							x			x		x		
Diagnose patient's treatmen	x									x	x				
Add a new symptom	x							x	x		x			x	
Check medical score		x									x		x		
See the prescription		x		x							x				
Create the prescription	x		x								x				
Download XML:	x														
Upload XML:	x							x							
Update medical score:	x						x	x			x			x	

(The pinks are functional requirements).

(The blues are non-functional requirements).

- **E-R diagram**



Doctor:

- Id (primary key)
- Name
- Surname
- Username

Patient:

- Date of Birth
- Username =id (primary key)
- Name
- Surname
- Password
- Gender

Symptom:

- Id(primary key)
- Type

Allergy:

- Id(primary key)
- Type

Prescription:

- Id(primary key)
- IsUsed
- GivenTo(Patient)
- GivenBy(Doctor)
- Treatment

Treatment:

- Id(primary key)
- Name
- Treatment(type)

- **Relational tables from E-R diagram:**

patient

PK			
id	name	dateOfBirth	gender
1	Noelia Auba	4/8/03	FEMALE
2	Carmen Caballero	25/4/03	FEMALE
3	Pablo Hita	6/3/99	MALE
4	Carlo Bottini	30/10/04	MALE
5	Fernando Guzman	26/10/02	MALE
6	Natalia Garcia	7/8/05	FEMALE
7	Jaime Martin	5/7/07	MALE
8	Maria Luna	19/2/03	FEMALE

doctor

PK	
id	name
1	Ruben Martinez
2	Catalina Sanchez

prescription

PK	boolean	FK--> patient (id)	FK--> doctor (id)	FK-->TREATMENT(id)
id	isUsed	given_to	given_by	treatment_id
1	YES	4	1	2
2	NO	2	2	5
3	NO	3	1	3
4	NO	1	1	1

symptom

PK		
id	name	type
1	Sneeze	Respiratory
2	Nasal Congestion	Respiratory
3	Cough	Respiratory
4	Runny Nose	Respiratory
5	Asthma Attack	Respiratory
6	Hives	Cutaneous
7	Itch	Cutaneous
8	Burning Rush	Cutaneous
9	Abdominal Pain	Digestive
10	Nausea	Digestive
11	Swelling Mouth	Digestive
12	Tingling Tongue	Digestive
13	Chest Tightness	Others
14	Watery Eyes	Others
15	Fever	Others
16	Headache	Others

allergy

PK		
id	name	type
1	Dairy Allergy	Alimentary
2	Pollen Allergy	Stationary
3	Nut Allergy	Alimentary
4	Dust Mite Allergy	Stationary
5	Insect Allergy	Insect
6	Drug Allergy	Drugs
7	Sun Allergy	Cutaneous
8	Gluten Allergy	Alimentary

treatment

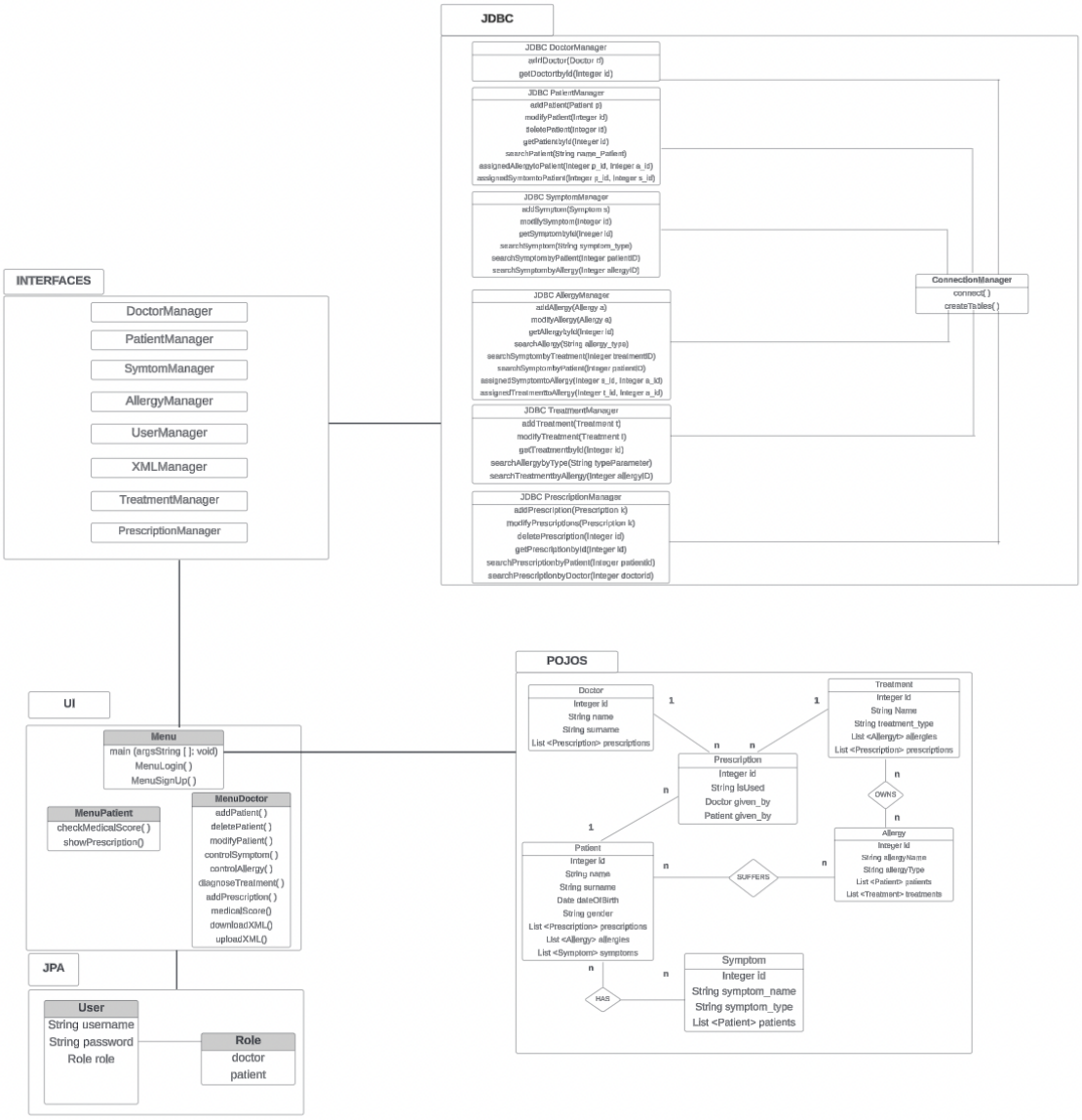
PK		
id	name	type
1	Desloratadina	Antihistamines
2	Dexametasona	Corticosteroids
3	Clorfenamina	Decongestants
4	Albuterol	Bronchodilators
5	Epinefrina	Vaccine
6	Prednisona	Corticosteroids
7	Oximetazolina	Decongestants
8	Metaproterenol	Bronchodilators
9	Adrenalina	Vaccine

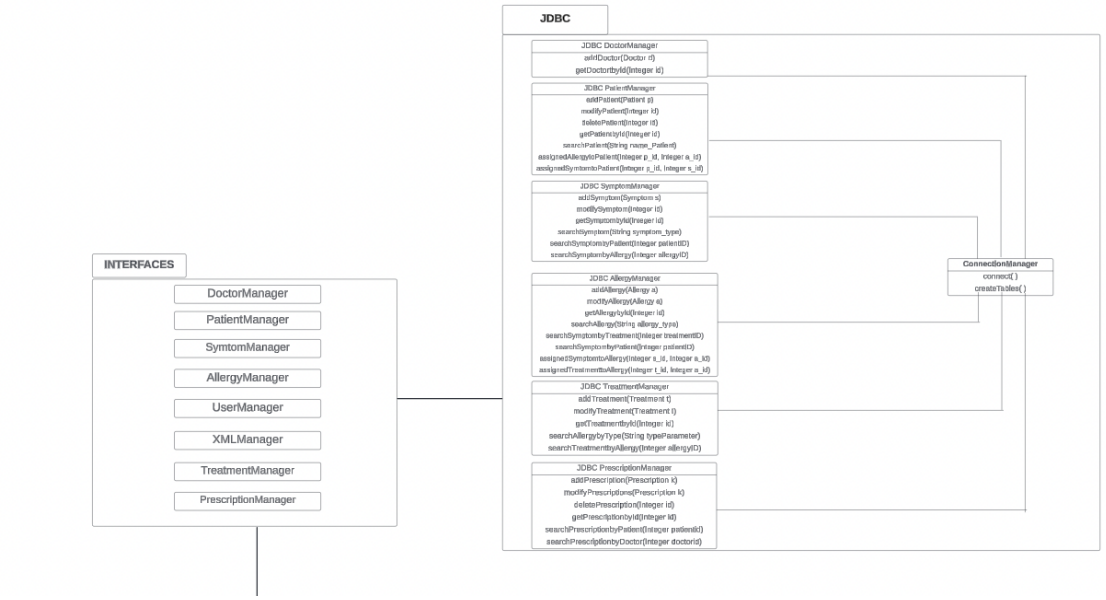
N-N TABLES:

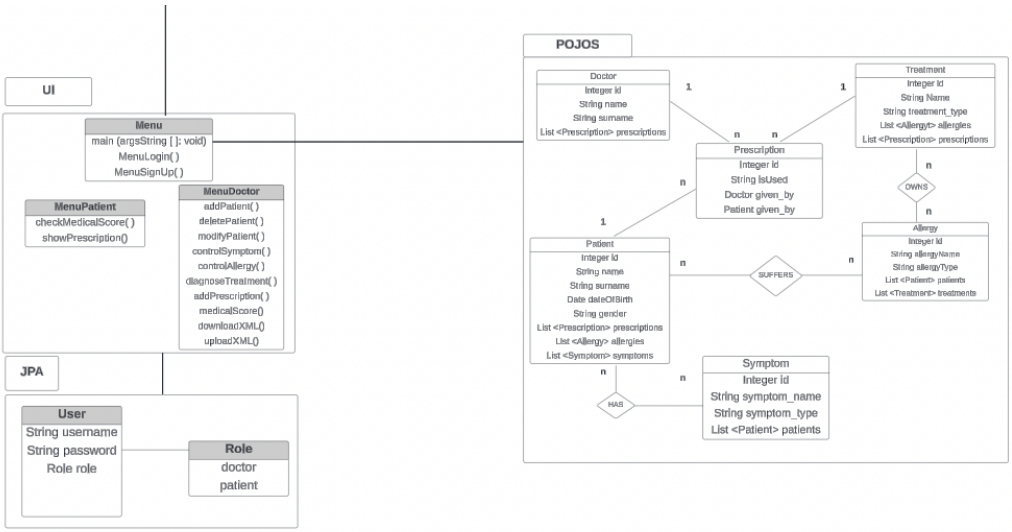
HAS			**OWNS**	
FK -> patient(id)	FK -> symptom(id)		FK -> allergy(	FK -> treatment(id)
patient_id	symptom_id		allergy_id	treatment_id
1	3		4	1
4	11		2	3
2	2		7	2
5	14		1	5
			3	1
			6	4
SUFFERS			8	9
FK -> patient(id)	FK -> allergy (id)			
patient_id	allergy_id			
1	1			
2	4			
3	3			
4	2			
5	6			
6	7			
7	5			
8	1			

- **<u>UML:</u>**

It is difficult to show everything together. So first, there is the entire UML. Then, it is divided for a better visual, but the zoom is needed as a help.

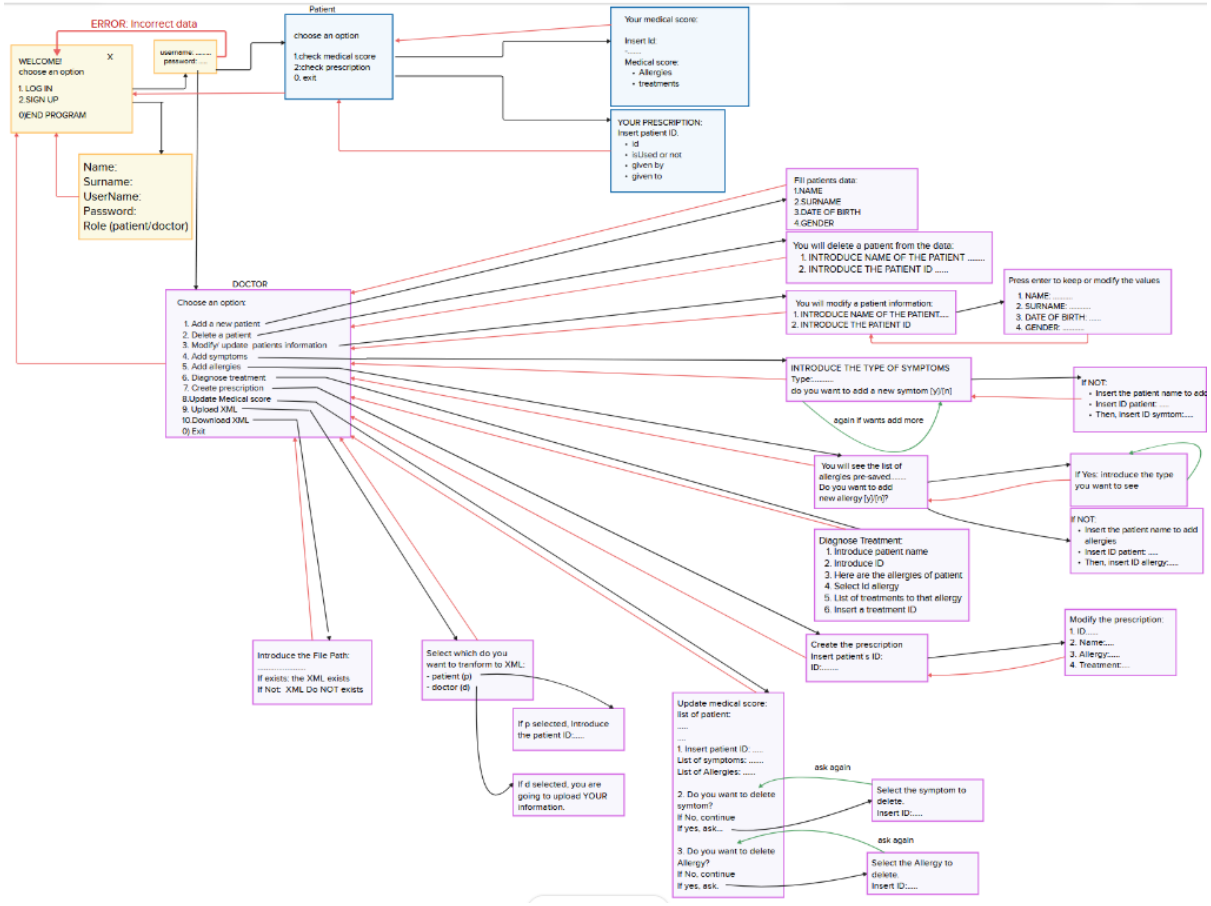


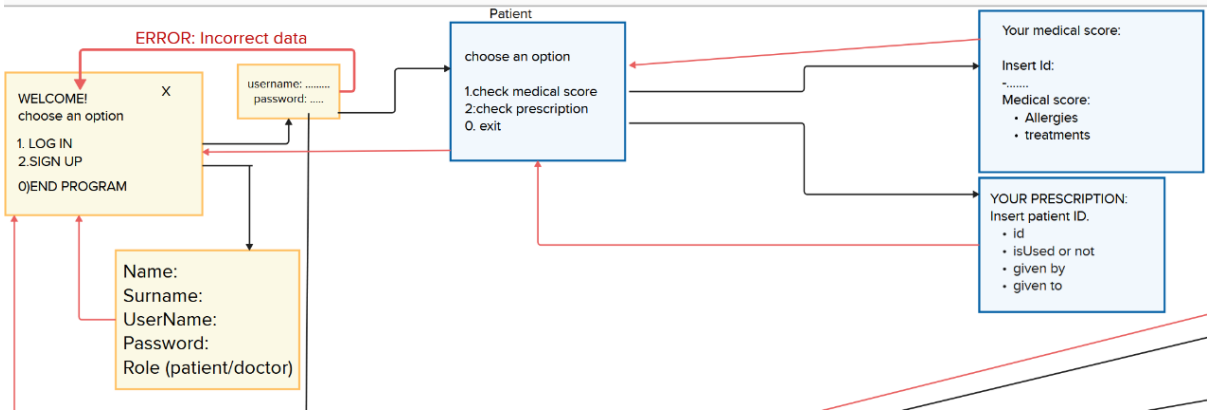

JDBC
JDBC DoctorManager
addDoctor(Doctor d)
getDoctorbyId(Integer id)
JDBC PatientManager
addPatient(Patient p)
modifyPatient(Integer id)
deletePatient(Integer id)
getPatientbyId(Integer id)
searchPatient(String name_Patient)
assignedAllergytoPatient(Integer p_id, Integer a_id)
assignedSymtomtoPatient(Integer s_id, Integer s_id)
JDBC SymptomManager
addSymptom(Symptom s)
modifySymptom(Integer id)
getSymptombyId(Integer id)
searchSymptom(String symptom_type)
searchSymptombyPatient(Integer patientID)
searchSymptombyAllergy(Integer allergyID)
ConnectionManager
connect( )
createTables( )
JDBC AllergyManager
addAllergy(Allergy a)
modifyAllergy(Allergy a)
getAllergybyId(Integer id)
searchAllergy(String allergy_type)
searchSymptombyTreatment(Integer treatmentID)
searchSymptombyPatient(Integer patientID)
assignedSymptomtoAllergy(Integer s_id, Integer a_id)
assignedTreatmenttoAllergy(Integer t_id, Integer a_id)
JDBC TreatmentManager
addTreatment(Treatment t)
modifyTreatment(Treatment t)
getTreatmentbyId(Integer id)
searchAllergybyType(String typeParameter)
searchTreatmentbyAllergy(Integer allergyID)
JDBC PrescriptionManager
addPrescription(Prescription k)
modifyPrescriptions(Prescription k)
deletePrescription(Integer id)
getPrescriptionbyId(Integer id)
searchPrescriptionbyPatient(Integer patientId)
searchPrescriptionbyDoctor(Integer doctorId)
INTERFACES
DoctorManager
PatientManager
SymtomManager
AllergyManager
UserManager
XMLManager
TreatmentManager
PrescriptionManager

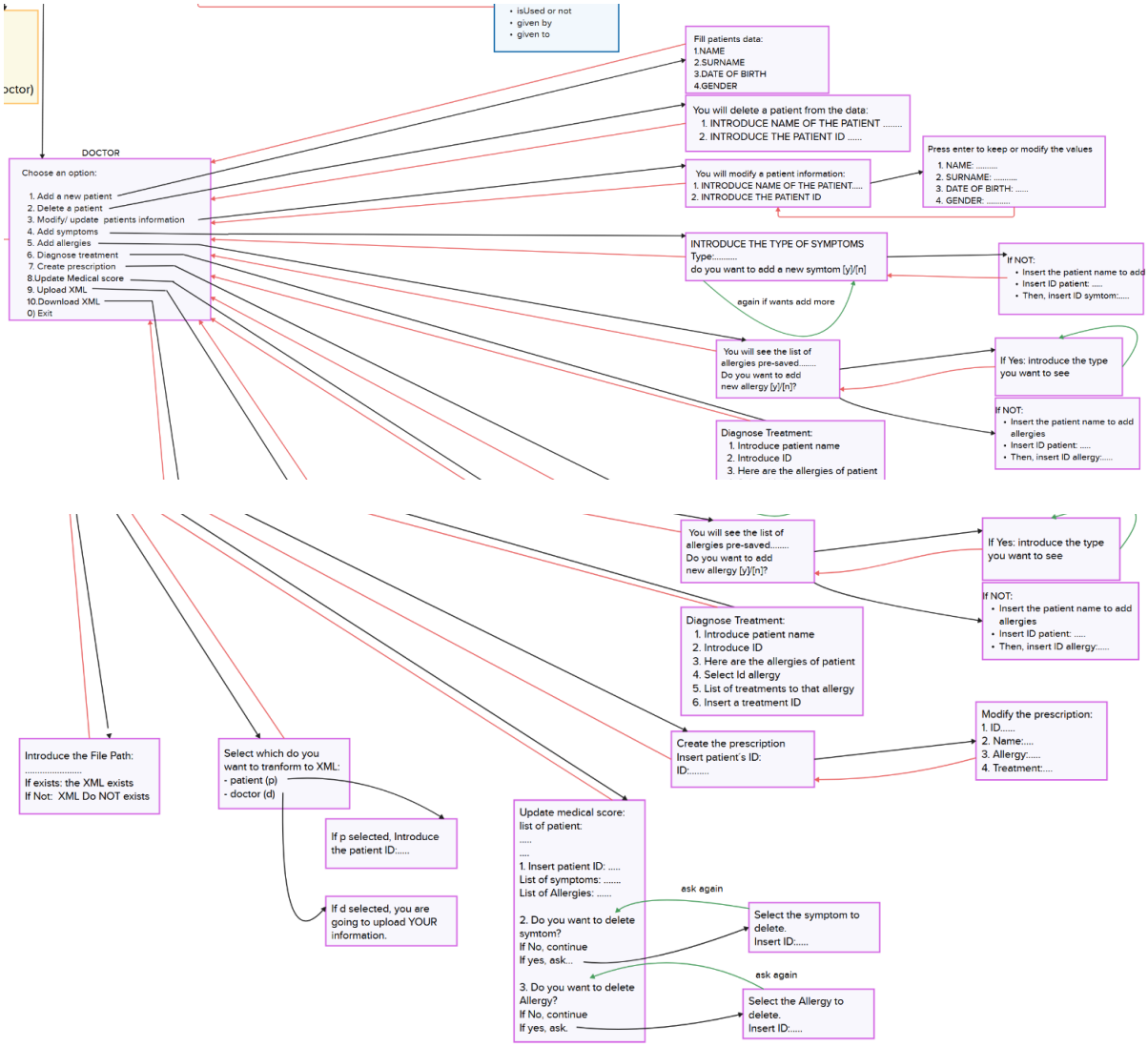

UI
Menu
main (argsString [ ]: void)
MenuLogin( )
MenuSignUp( )
MenuPatient
checkMedicalScore( )
showPrescription()
MenuDoctor
addPatient( )
deletePatient( )
modifyPatient( )
controlSymptom( )
controlAllergy( )
diagnoseTreatment( )
addPrescription( )
medicalScore()
downloadXML()
uploadXML()
JPA
User
String username
String password
Role role
Role
doctor
patient
POJOS
Doctor
Integer id
String name
String surname
List <Prescription> prescriptions
Treatment
Integer id
String Name
String treatment_type
List <Allergy> allergies
List <Prescription> prescriptions
Prescription
Integer id
String IsUsed
Doctor given_by
Patient given_by
OWNS
Allergy
Integer id
String allergyName
String allergyType
List <Patient> patients
List <Treatment> treatments
Patient
Integer id
String name
String surname
Date dateOfBirth
String gender
List <Prescription> prescriptions
List <Allergy> allergies
List <Symptom> symptoms
SUFFERS
HAS
Symptom
Integer id
String symptom_name
String symptom_type
List <Patient> patients

- **Mock-up skeleton:**

Link to the original in the web:
https://app.mural.co/t/mockupbatabaseproyect9546/m/mockupbatabaseproyect9546/1710780784922/805a8ac5800fd8b751891e80d0aa2d8b4657a41f?sender=u83d35c18692736f8de918685


ERROR: Incorrect data
WELCOME!
choose an option
1. LOG IN
2.SIGN UP
0)END PROGRAM
username: .....
password: .....
Patient
choose an option
1.check medical score
2:check prescription
0. exit
Your medical score:
Insert Id:
Medical score:
Allergies
treatments
YOUR PRESCRIPTION:
Insert patient ID.
id
isUsed or not
given by
given to
Name:
Surname:
UserName:
Password:
Role (patient/doctor)
DOCTOR
Choose an option:
1. Add a new patient
2. Delete a patient
3. Modify/ update patients information
4. Add symptoms
5. Add allergies
6. Diagnose treatment
7. Create prescription
8.Update Medical score
9. Upload XML
10.Download XML
0) Exit
Fill patients data:
1.NAME
2.SURNAME
3.DATE OF BIRTH
4.GENDER
You will delete a patient from the data:
1. INTRODUCE NAME OF THE PATIENT .....
2. INTRODUCE THE PATIENT ID .....
You will modify a patient information:
1. INTRODUCE NAME OF THE PATIENT....
2. INTRODUCE THE PATIENT ID
Press enter to keep or modify the values
1. NAME: .....
2. SURNAME: .....
3. DATE OF BIRTH: .....
4. GENDER: .....
INTRODUCE THE TYPE OF SYMPTOMS
Type:.....
do you want to add a new symtom [y][n]
again if wants add more
If NOT:
Insert the patient name to add
Insert ID patient: ....
Then, insert ID symtom:....
You will see the list of allergies pre-saved.....
Do you want to add new allergy [y][n]?
If Yes: introduce the type you want to see
If NOT:
Insert the patient name to add allergies
Insert ID patient: ....
Then, insert ID allergy:....
Diagnose Treatment:
1. Introduce patient name
2. Introduce ID
3. Here are the allergies of patient
4. Select Id allergy
5. List of treatments to that allergy
6. Insert a treatment ID
Create the prescription
Insert patient s ID:
ID:.....
Modify the prescription:
1. ID:....
2. Name:....
3. Allergy:.....
4. Treatment:...
Introduce the File Path:
.....
If exists: the XML exists
If Not: XML Do NOT exists
Select which do you want to trenform to XML:
- patient (p)
- doctor (d)
If p selected, Introduce the patient ID:.....
If d selected, you are going to upload YOUR information.
Update medical score:
list of patient:
.....
1. Insert patient ID: .....
List of symptoms: .....
List of Allergies: .....
2. Do you want to delete symtom?
If No, continue
If yes, ask...
ask again
Select the symptom to delete.
Insert ID:.....
3. Do you want to delete Allergy?
If No, continue
If yes, ask.
ask again
Select the Allergy to delete.
Insert ID:.....


ERROR: Incorrect data
WELCOME!
choose an option
1. LOG IN
2.SIGN UP
0)END PROGRAM
username: .....
password: .....
Patient
choose an option
1.check medical score
2:check prescription
0. exit
Your medical score:
Insert Id:
Medical score:
Allergies
treatments
YOUR PRESCRIPTION:
Insert patient ID.
id
isUsed or not
given by
given to
Name:
Surname:
UserName:
Password:
Role (patient/doctor)



- **XML:**

In our project, we used XML to facilitate the structured storage and easy exchange of data between the system components, particularly for representing patient and doctor information. We used JAXB (Java Architecture XML Binding) to convert Java objects into XML files, using marshaller, and XML files back to Java objects, using unmarshaller.

For marshalling, the methods patient2XML and doctor2XML convert Patient and Doctor objects into XML files. This process involves creating a JAXBContext for the respective class, then using a Marshaller to output the object data in a well-structured XML format. The generated XML files are saved in the ./xmls directory, and the XML content is also printed to the console for verification.

- **XSLT:**

Additionally, to improve the presentation of our project, we employed XSLT (Extensible Stylesheet Language Transformation) to transform these XML files into HTML for web presentation. This approach ensured data consistency, portability, and an organized separation between data storage and presentation layers.

The methods patient2Html and doctor2Html handle the transformation of XML files into HTML using XSLT (Extensible Stylesheet Language Transformations). This involves first converting the object to XML, then using a TransformerFactory to create a Transformer with a specified XSLT stylesheet. The transformer then processes the XML file to produce an HTML file, saved in the ./xmls directory.

- **DTD:**

Finally, we have implemented the DTD for our two XML files, patient and doctor, to ensure data integrity and consistency by defining the structure and rules for these documents. This facilitates validation of the XML data, ensuring it meets the required standards, and improves interoperability between different systems by providing a clear and consistent data format.