

ALLERGY DETECTION DATABASE

Index:

- The group composition and the project's topic.	2
- The project's feature, the requirement's table, the use case diagram and the traceability matrix	2-9
- E-R diagram	9-10
- Relational table	11
- UML	12-13
- Mock-up skeleton	14-15
- XML	15
- HTML	15

- **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 patient, I want to check my medical score
- 9) As a patient, I want to see the prescription for my treatment

USES CASES:

(1)

Name: Add a new patient

Actor: Doctor

Goal: Doctor add a new patient on the data base.

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 exist.
- 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 don'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 exist
- 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 don't have symptoms
- The patient do 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 exist
- 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 don't have symptoms
- The patient do not need a treatment

Trigger: The patient needs treatment

Postconditions: The patient gets its prescription.

(8)

Name: See the patient's Medical Score.

Actor: Patient.

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

Preconditions:

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

Standard path: The patient is able to 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.

(9)

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.

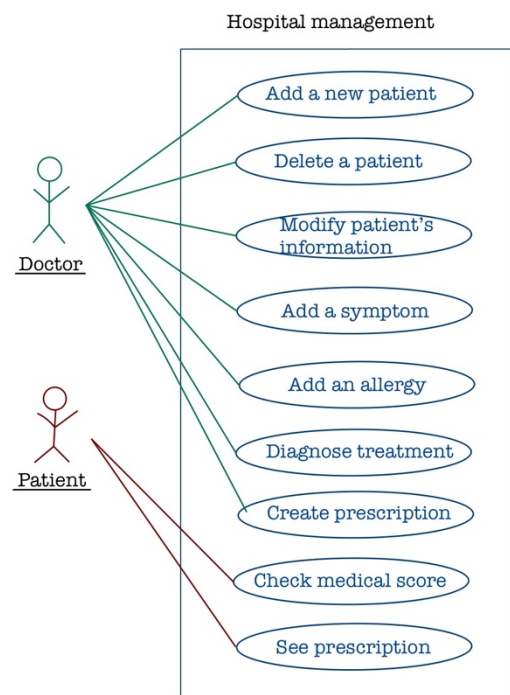
Alternative path:

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

Trigger: The patients want to see the prescription

Postconditions: The patient see 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

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:

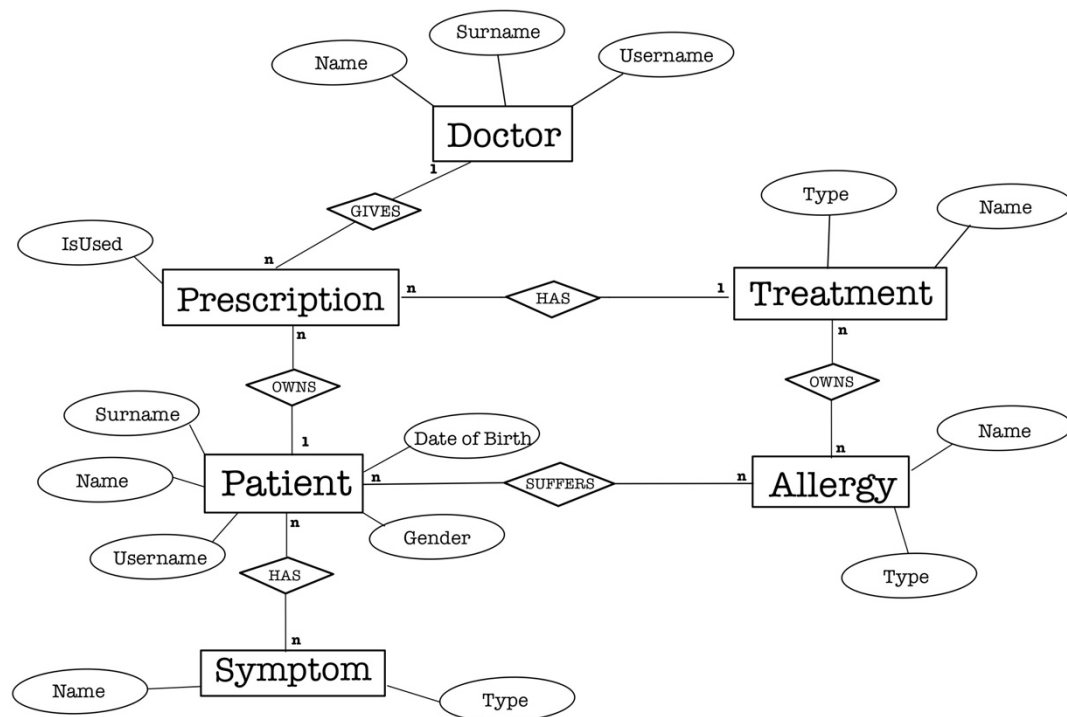
- The system will complete the commands in less than 2 seconds. **(NF1)**
- The system is programmed in Java. **(NF2)**
- The same allergy cannot be registered twice. **(NF3)**
- There are no symptoms. **(NF4)**
- The same patient cannot be registered twice. **(NF5)**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	NF1	NF2	NF3	NF4	NF5
2	Delete a patient	x				x			x			x				x
3	Modify a patient	x						x	x			x				x
4	Add a Patient	x					x					x				x
5	Add new allergy	x							x			x		x		
6	Diagnose patient's treatment	x									x	x				
7	Add a new symptom	x							x	x		x			x	
8	Check medical score		x									x		x		
9	See the prescription		x		x							x				
10	Create the prescription	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:**

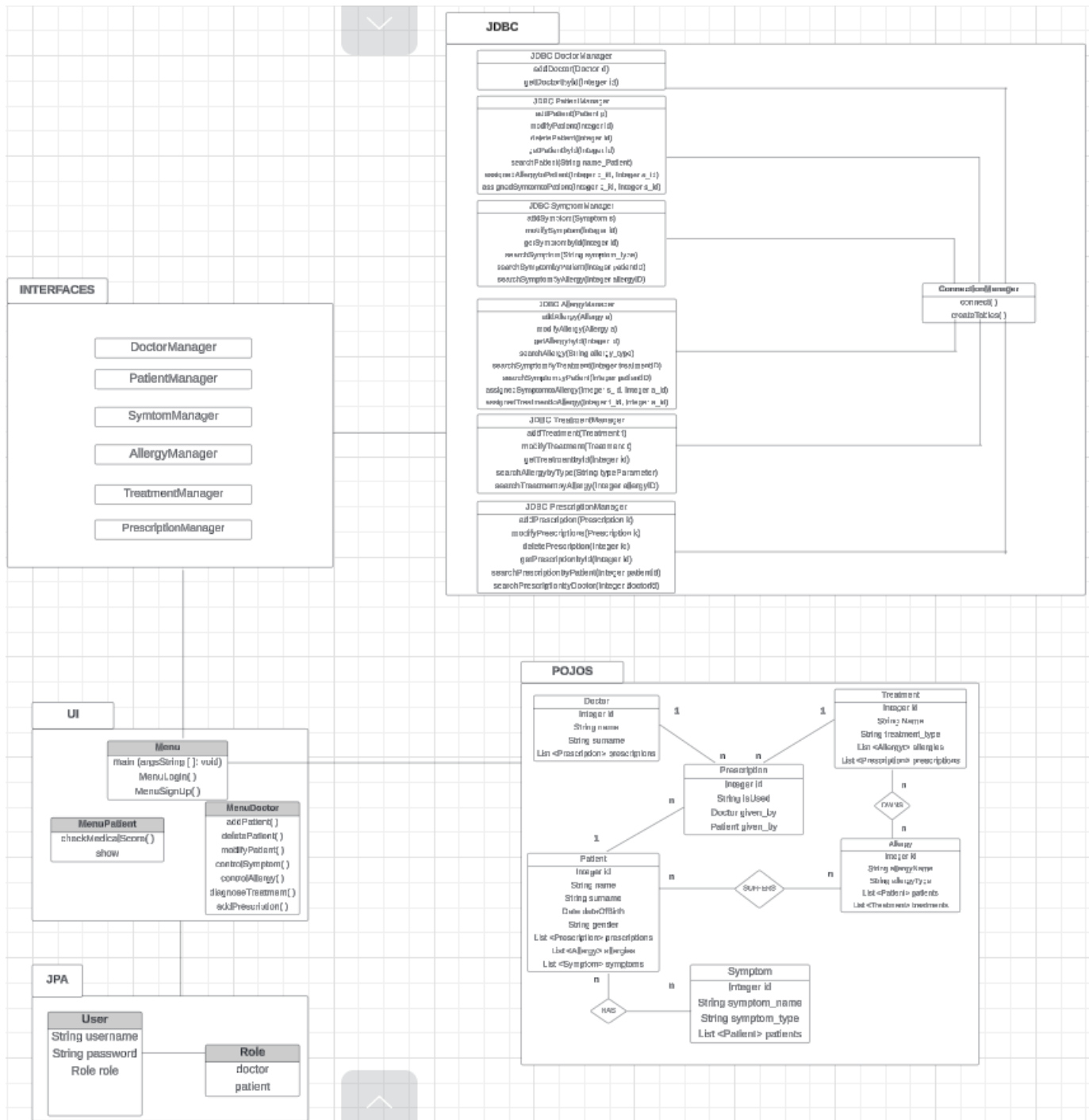
[illegible]

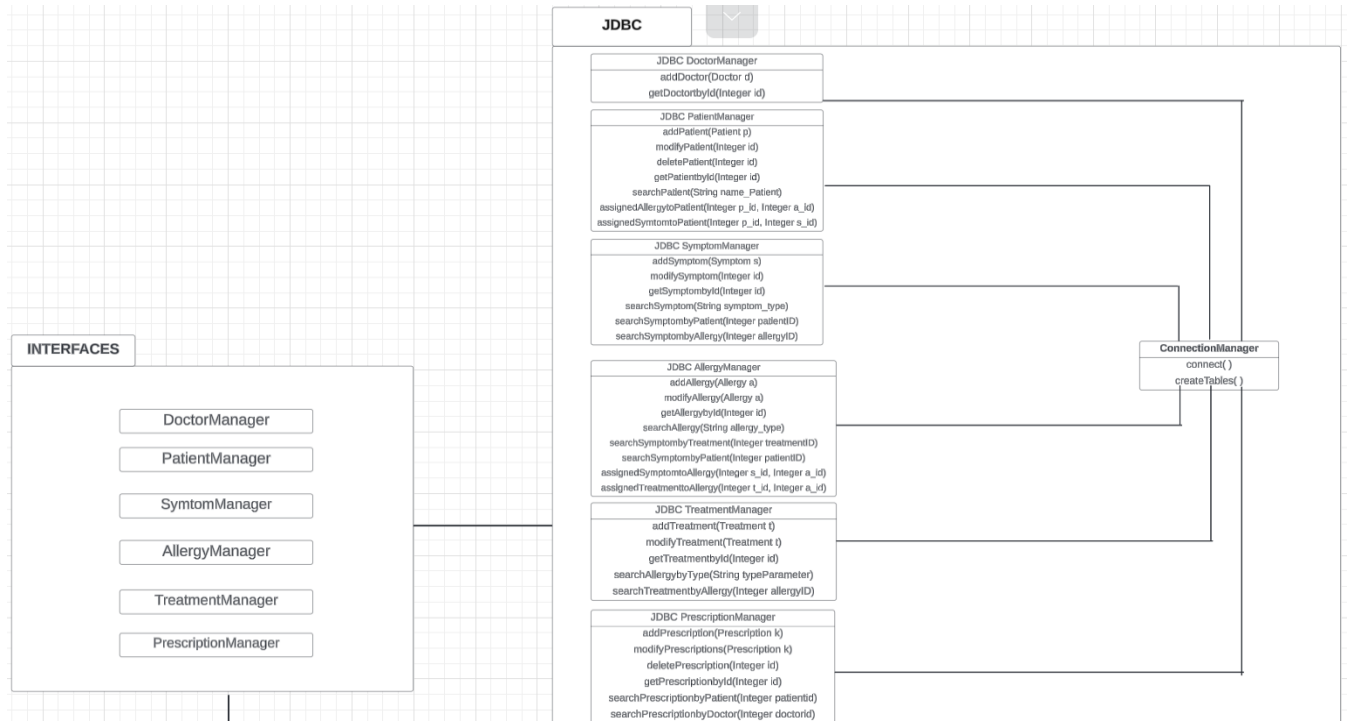
N-N TABLES:

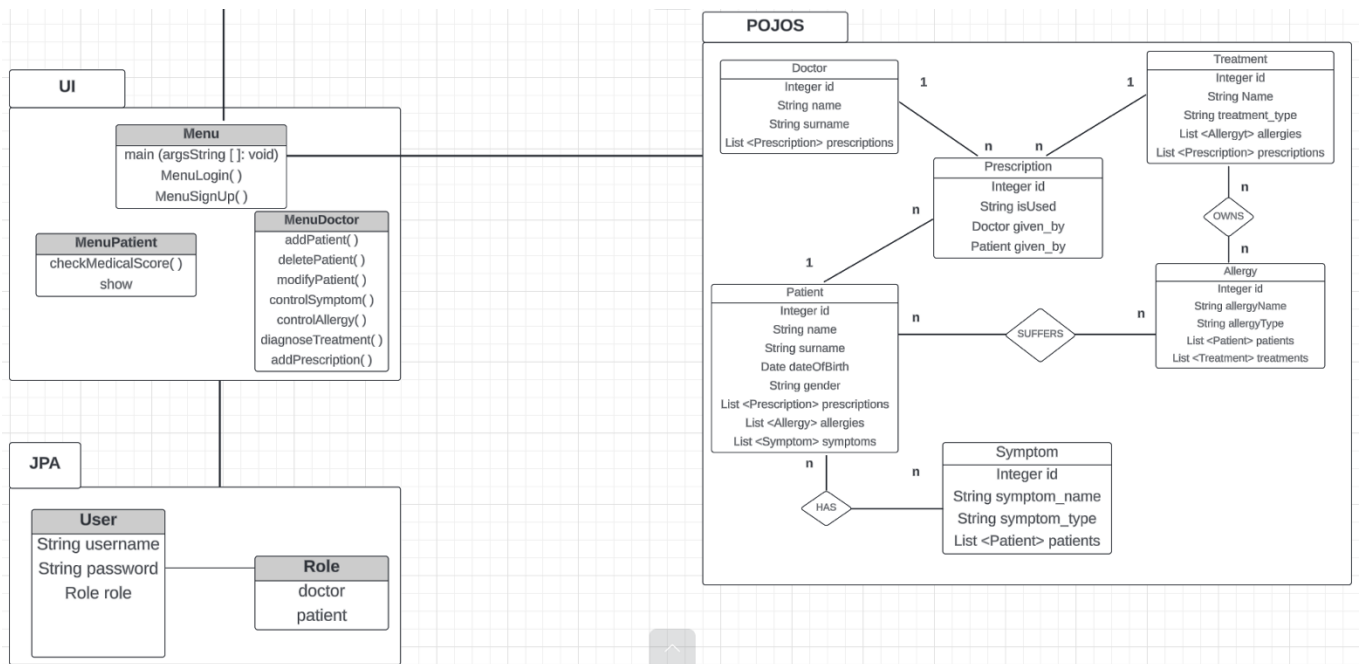
HAS		OWNS	
FK -> patient(id)	FK -> symptom(id)	FK -> allergy(id)	FK -> treatment(id)
patient_id	symptom_id	allergy_id	treatment_id
1	3	4	1
4	11	2	3
2	2	7	2
5	14	1	5
		3	1
		6	4
		8	9
SUFFERS			
FK -> patient(id)	FK -> allergy (id)		
patient_id	allergy_id		
1	1		
2	4		
3	3		
4	2		
5	6		
6	7		
7	5		
8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

- **UML:**

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.



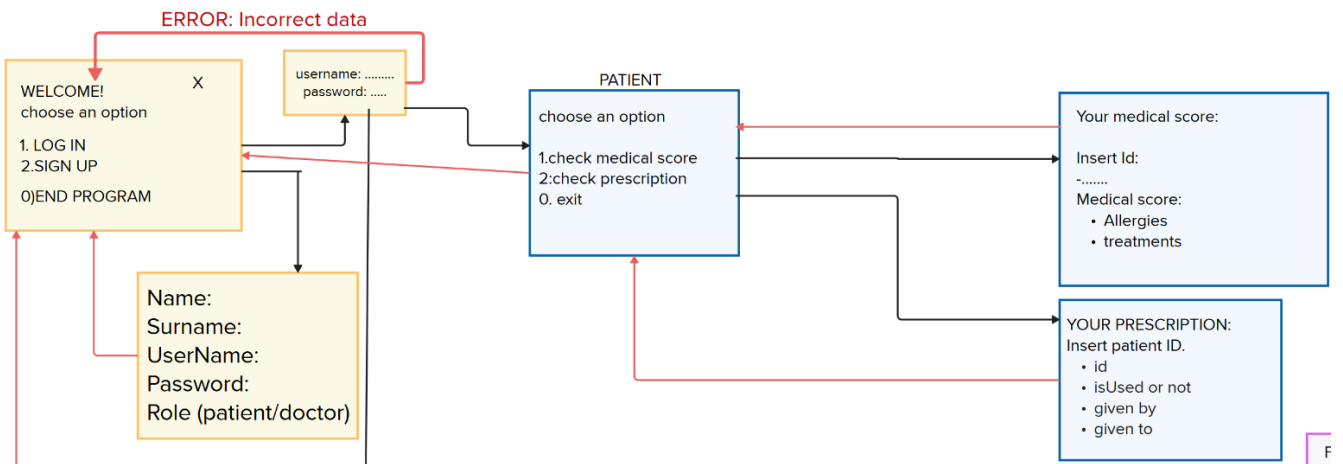
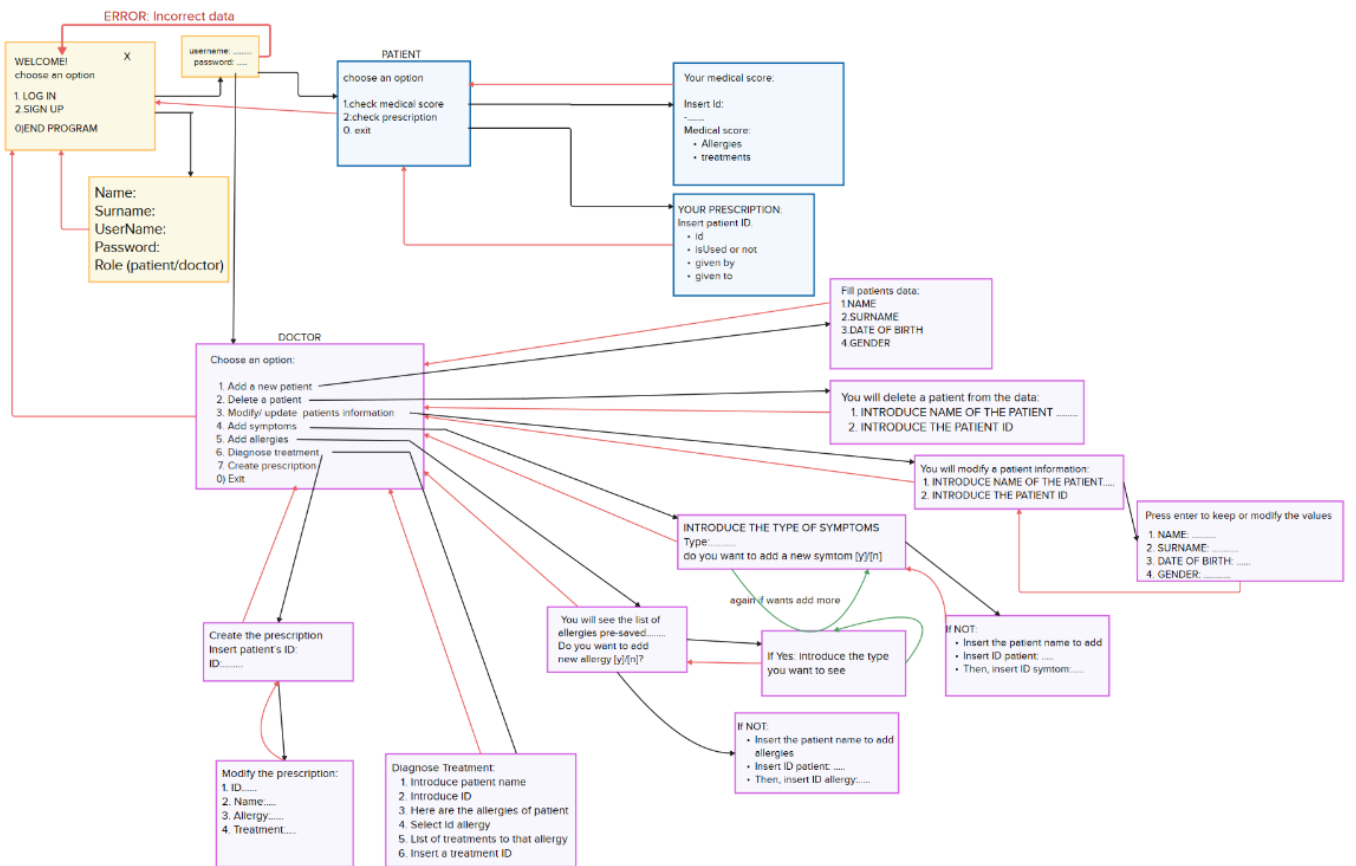


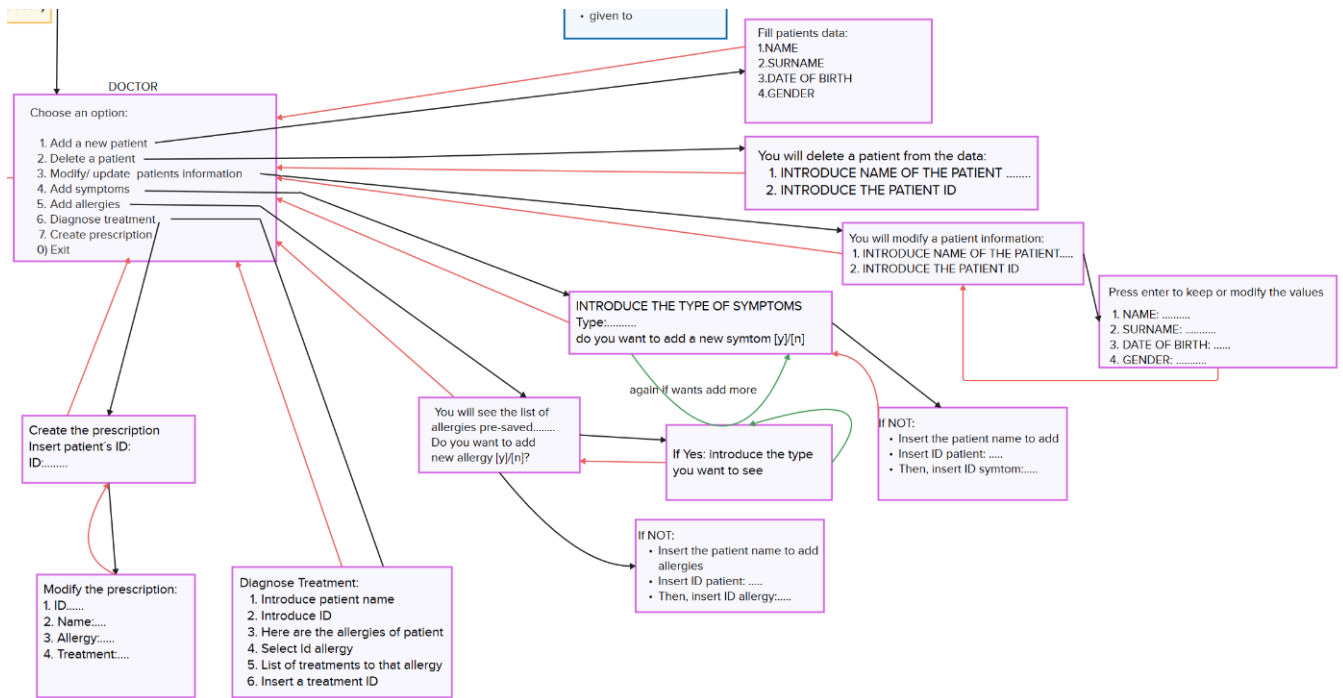


- **Mock-up skeleton:**

Link to the original in the web:

<https://app.mural.co/t/mockupbatabaseproyect9546/m/mockupbatabaseproyect9546/1710780784922/805a8ac5800fd8b751891e80d0aa2d8b4657a41f?sender=u83d35c18692736f8de918685>





• 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.

• HTML:

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.