DATABASE PROJECT- Part1

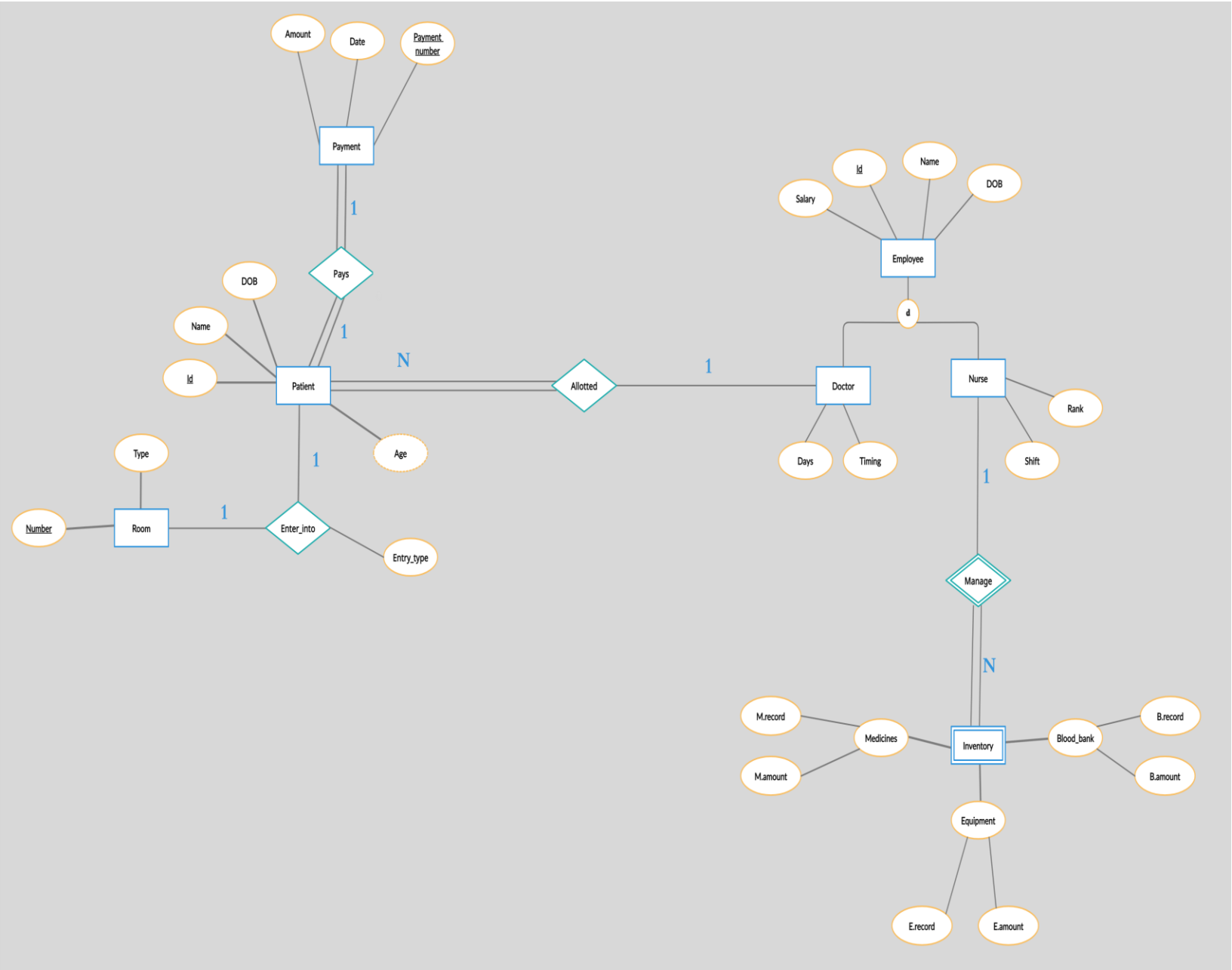
Done By:

-Qusai Alzeer 20180033

-Hala Farraj 20180664

CONTENTS

Diagram


Amount
Date
Payment number
Payment
1
Pays
1
DOB
Name
Id
Patient
N
Allotted
1
Id
Name
DOB
Salary
Employee
d
Doctor
Nurse
Rank
Days
Timing
Shift
Type
1
Age
1
Number
Room
1
Enter_into
Entry_type
1
Manage
N
M.record
Medicines
M.amount
Inventory
Blood_bank
B.record
B.amount
Equipment
E.record
E.amount

Schema

Patient

P.id	Name	DOB	E.id

Payment

P.num	Date	Amount	P.id

Room

Number	Type	Entry_type	P.id

Employee

E.id	Name	DOB	Salary	Days	Timing	Shift	Rank	Job_type

Inventory

M.record	M.amount	E.record	E.amount	B.record	B.amount	E.id

Description

Not every hospital is aware of everything going inside of it, nor is every hospital on top of things in terms of managing the hospital and how well the doctors, the patients, the employees, and the payments blend together. From here, our goal, based on the customer's request, was to design a database model for hospitals. This database is primarily designed for the purpose of managing any hospital that wishes to implement our system inside of it.

The customer had asked for a fully functional database system that can keep up with the hospital's records, follow up with the patients, assign respective doctors to them, schedule their reservations as well as to handle their payments. Additionally, the customer had placed requirements and restrictions for the desired database model on us.

In order to meet the customer's needs and to achieve our goal; our group split the tasks into smaller subtasks. Initially, each of us was assigned a specific task to perform. But it was only a matter of time until this technique proved its ineffectiveness; the tasks were atomic; it was nearly impossible to divide them. For instance, we were required to design a database diagram, this diagram elements are strictly connected to one another.

We couldn't divide the design of the database diagram on both of us. We had to do it together so that we both can comprehend everything perfectly. From that point on, we started to conduct Zoom meetings in where we shared our screens and discussed the problems together.
We started working on the project and used an application called "Creately" to design the database diagram. For starters, we defined our main entities and set up their attributes accordingly. Our entities were as follows:

- **Patient - which has the attributes (id, name, DOB)**
- **Payments - which has the attributes (Amount, Date, PaymentNumber)**
- **Room - which has the attributes (type, number)**
- **Employee - which has the attributes (id, name, Salary, DOB)**
- **Doctor - which has the attributes (Days, Timing)**
- **Nurse - which has the attributes (Rank, Shift)**
- **Inventory - which has the attributes (M.record, M.amount, E.record, E.amount, B.record, B.amount)**

Marked in green are the primary keys.

After constructing the base of our diagram (the entities), we had to connect these entities with each other in a systematic, innovative way. We discussed the payment system and we both agreed that one of the key characteristics of a well-organized hospital is the payments system.

Patients loathe waiting for hours in a long endless queue, just to pay their bills and get told that they need to wait another hour or two for the doctor to be free, we all have experienced this and know exactly how frustrating it can leave people. And therefore, A quick money transaction system, without wasting time or waiting for a queue to move, was needed. As such, our database has a complete billing system that should make it ten times easier for customers to reserve, pay, and get treated in a significantly shorter time.
Moreover, all the payments were put in one payment table, which helps us dramatically when it comes to maintaining the transactional integrity constraints.

Upon entering the hospital, patients will be assigned rooms if they're admitted. We implemented this on our diagram by creating a relationship, named "Enter_into", that connects the patients with the rooms.
This relationship will have an attribute named "Entry_type", which refers to the emergency type of the patient. It is important to keep in mind that some patients may not be assigned a room (depending on the availability of the rooms) and therefore, the completeness constraints on this relationship will be Partial and 1-to-1 on both sides of the relationship. (if the "Type" attribute was set to 'null'; that signifies that there are no rooms available and hence, the patient is not admitted to enter and the reservation is not successful).

If a patient was admitted into the hospital, they will be assigned to rooms based on their emergency types. Additionally, A doctor will be assigned to take care of said patient. This can be represented in the diagram by creating a relationship between the patient and the doctor.
We can also notice that the patient must have exactly one doctor, meanwhile, the doctor can have many patients or none. Therefore, the relationship named "Allotted" that assigns the doctor to the patient(s) will be total and many on the patient's end, and partial and one on the doctor's end.

Now, moving to the hospital's staff; A hospital may have doctors or nurses as employees, but it could also have other employees as well (i.e, database administrators who can maintain and manage the hospital electronically, or other workers who might be on charge of keeping the hospital's corridors clean and sanitized). So, all in all, an employee could be a doctor, a nurse, or other. From this requirement, we defined an entity called "Employee" which represents a superclass connected in partial and disjoint relationship with its subclasses "Doctor" and "Nurse". (Doctor/ Nurse is an Employee).
And the reason that we decided make it partial is because employees are not restricted to the two types of Doctor and Nurse only, but could extend to other employees as aforementioned. Also, it was made to be disjoint since doctors cannot be nurses at the same time and vice versa.

For the last part of the diagram, we were told that nurses can have different ranks, ranging from 1 to 5, and only those among them who have a rank of 1 can manage the hospitals inventory.

Which leads us to our final relationship, namely, "Manage". The inventory has no primary key so it had to be a weak entity. Not all nurses need to manage the inventory, but rather, only those who have a rank of 1. So, we assumed that the nurse, the one with rank of 1, can manage many inventories. On the other hand, each inventory must be managed by at one nurse of rank 1.

From there, we can define the relationship between the nurse and the inventory to be 1-to-m, from the nurse's side to the inventory's one. And the completeness constraints are total on the inventory's end (since each inventory must have at least one manager), and partial on the nurses' end (since some nurses of other ranks do not manage the inventory).

Moreover, we tried to make an EER diagram from the inventory because we thought that blood bank, for example, is an inventory; but lastly, we found that there is no attributes for the subclasses (no entity without an attribute) so we cancelled this idea. Finally, we found that we can solve this problem by creating composite attributes (giving a record and amount for each one). Logically, we found that it is the best way to know the amount of the record. To be more obvious, here is a table of what we mean by the given attributes, as instances:

M.record	M.amount	E.record	E.amount	B.record	B.amount	E.id
Galvus	**750**	**Otoscope**	**50**	**O-**	**165**	**201326**
Enteresto	**60**	**Scalpel**	**20**	**A+**	**30**	**201326**
Relaxon	**537**	**Oxygen tank**	**80**	**AB+**	**68**	**231867**

- **M.record: The medicine that we want to know its amount**
- **M.amount: The amount of the recorded medicine (as pills)**
- **E.record: The equipment what we want to know its amount**
- **E.amount: The amount of the recorded equipment**
- **B.record: The blood type that we want to know its amount**
- **B.amount: The amount of the recorded blood type (as units)**
- **E.id: The ID of the nurse who is managing this inventory**

So, the nurse that is with 231867 id manages Relaxon, Oxygen tank and AB+ and their amounts.
As we mentioned, a nurse can manage many inventories.
We found that this is the best way to save the amount of every record in the database and it can be easily entered in the GUI, when it's implemented.