**Project Direction Overview:**

I would like to develop a hospital management system mobile based application named **“Health Matters”** which can be used by any size hospital to better manage their patients and their assets. I intend to target this application for government hospitals to benefit the entire public and the hospital itself. This application will contain the data for the patients (their personal information, medical history, vaccination, etc.) as well as the medical staff at the hospital. The application is initially targeted for use by patients as they are the number one priority and then extend to doctors, administrators and so on.

Now I will explain with some examples for how the application is supposed to be used. The application will be having different type of logins depending on if you are a patient, doctor, or an administrator. A patient is new to the hospital and may or may not know about the hospital application. When the patient visits the hospital, he/she will go to the reception for his/her query and receptionist may refer the patient to some doctor and bill him for the appointment accordingly. On the bill will be mentioned the patient ID number and the details provided by the patient to the receptionist. The receptionist will also inform the patient about the mobile app which can be installed by scanning the QR code or the link provided on a poster at the reception. The patient can now login to the application using the patient ID and will be asked to setup the password and profile and to link his/her email address with it. If the patient had already installed the application before coming to the hospital, then they will already be having a patient ID. Now all the data related to the patient will be visible on the application and he can track it.

People who are visiting government hospitals are usually those who are not very affording. This application is built to provide them with ease of access to their data, make payments, appointments, remind them if something is due like a vaccination or an appointment etc. There will be a lot of programming involved with the front-end GUI and everything, but I will only be focusing on the database component of this application.

**Use Cases and Fields:**

One important use case will be the patient interacting with the app for the first time and creating an account. Steps are as follows:

1. The patient visits the hospital website or app store (Google, Apple, etc.) to install the application for the first time.
2. The application asks them if they are a new user and do not have an associated patient ID and if so, then prompts them to create an account.
3. The patient creates an account and enters his/her personal information which is entered into the database at the back end.
4. The patient now has the access to all his/her information.

I am modelling this data base using a hierarchical model, so I have created a **Person** entity which can be a **Doctor** entity or **Patient** entity or both. Some important fields for the **Person** entity are as follows:

|  |  |  |
| --- | --- | --- |
| **Field** | **What it stores** | **Why it’s needed** |
| Person ID (P\_ID) | A unique number to identify the person. | This is the unique identifier of the person; the person details can be accessed using this ID. |
| First Name (P\_FNAME) | This is the first name of the person. | This will be shown in the account and will be used to address the person. |
| Last Name (P\_LNAME) | This is the last name of the person. | This will be shown in the account and will be used to address the person. |
| Mobile Number (P\_MOBNUM) | This is the mobile number of the person. | This will be used to contact the person. |
| Permanent Address (P\_PADDRESS) | This is the permanent address of the person. | This is where the person lives and can be used to send medications etc. |
| Gender (P\_GENDER) | This is the gender of the person. | This is required to identify the gender of the person. |
| Emergency Mobile Number (P\_EMOBNUM) | This is the emergency mobile number. | This will be used to contact the person appointed to contact in case of an emergency with the person. |
| Email address (P\_EADDRESS) | This is the personal email address of the person. | This will be used to send all the reports, bills, etc. to the person for record. |

**Patient** entity and **Doctor** entity will inherit P\_ID from this table. The **Patient** only has the P\_ID field so far.

Another important use case is the patient using the application to make an appointment.

1. The patient logs into the application using his/her patient ID.
2. The patient goes to the appointment tab to make an appointment with a specialized doctor.
3. The tab shows him the list of doctors working at the hospital, their related information such as name, designation, office hours, etc. which is stored in the database at the back end.
4. The patient selects the doctor to make an appointment and is asked to pay for the appointment either through the application or at the reception.
5. The patient makes the payment through application using his/her credit card and the appointment is confirmed and patient provided with an appointment number.
6. The application will remind the patient of the appointment depending on the reminder setting in the application set by the user.

Some important fields for doctor information are as follows:

|  |  |  |
| --- | --- | --- |
| **Field** | **What it stores** | **Why it’s needed** |
| Office Hours (D\_OFFICEHOURS) | This is the timing of the office. | This will be used to display when the doctor will be available for a checkup. |
| Qualification (D\_QUALIFICATION) | This is the qualification of the doctor. | These are the degrees attained by the doctor. |
| Designation (D\_DESIGNATION) | This is the designation of the doctor. | This is the designation of the doctor such as professor, etc. |
| Account Number (D\_ACCOUNT NUMBER) | This is the account number of the doctor. | This is the account to which the salary of the doctor will be transferred. |

I have modelled the relationship of the **Billing** entity with the **Person** entity, so anyone can owe bills to the hospital. The P\_ID will be the foreign key from **Person** entity and a part of the primary key of the **Billing** entity. Some important fields for payment information are as follows:

|  |  |  |
| --- | --- | --- |
| **Field** | **What it stores** | **Why it’s needed** |
| Billing ID (B\_ID) | A unique number to identify each billing. | This is the unique identifier of the bill; the billing details can be accessed using this ID. |
| Payment Type (B\_PAYMENTTYPE) | This is the type of payment cash / card. | This is required for billing purposes. |
| Billing Address (B\_BADDRESS) | This is the billing address. | This will be used for billing purposes etc. |
| Credit/Debit Card Number (B\_CARDNUMBER) | This is the credit/debit card number. | This will be used for billing purposes. |
| Security Code (CVC) (B\_CVC) | This is the CVC code of the credit/debit card. | This will be used for billing purposes. |
| Credit Card Expiry (B\_CARDEXPIRY) | This is the expiration date of the credit/debit card. | This will be used for billing purposes. |
| Credit Card Holder Name (B\_CARDNAME) | This is the name on the credit/debit card. | This will be used for billing purposes. |
| Payment Status (B\_PAYMENT STATUS) | This store status of payment such as paid, due, etc. | This is needed to check if the payment has been cleared or not. |

The Appointment entity will have the **Patient** & **Doctor** ID as foreign keys. Some important fields for appointment information are as follows:

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| Appointment Number (A\_ID) | The unique number of the appointment. | To check if the appointment was made and the details. |
| Patient Complain (A\_PATIENT COMPLAIN) | The issue with the patient. | To keep record of the patient’s, complain. |

Another use case could be if the patient wants to look up his/her medical history to check his/her history of treatment at the hospital and for his/her medication:

1. The patient logs into the application using his/her patient ID.
2. The patient goes to the patient history tab in the application.
3. The patient checks his medical history stored in the database.
4. The history will show each appointment ID, the doctor with whom the appointment was made, the patient medical condition, diagnosis, the medication prescribed by the doctor.
5. The patient can then log out once they have the required information.

The **Diagnosis** entity will have appointment ID as a foreign key from the **Appointment** entity. Some important fields for the diagnosis are:

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| Doctor Diagnosis (DI\_DOCDIAG) | Diagnosis of the patient complain. | To keep record of the diagnosis. |
| Doctor Medication (DI\_DOCMED) | Medication given by the doctor. | To keep record of the medication. |
| Doctor Referral (DI\_DOCREF) | If doctor refers to some other doctor. | To keep record of doctor referral. |
| Doctor next appointment date (DI\_NEXTAPP) | Next appointment date with the doctor. | To keep record of next appointment date. |
| Patient admission type (IN/OUT) (DI\_ADMTYPE) | Doctor decision if patient should be admitted. | To keep record of the admission type. |

**Health Matters Structural Rules:**

**First use case:**

Patient interacting with the app for the first time and creating an account:

1. The patient visits the hospital website or app store (Google, Apple, etc.) to install the application for the first time.
2. The application asks them if they are a new user and do not have an associated patient ID and if so, then prompts them to create an account.
3. The patient creates an account and enters his/her personal information which is entered into the database at the back end.
4. The patient now has the access to all his/her information.

As I am only focused on the database part so only the information that needs to be stored in the database is of concern to me. From step 2 of this use case, I can see one **entity Patient**. In looking at the other steps I cannot see any entity or relationship. As I only have one entity for now and no relationships therefore, I cannot infer a business rule from just this first use case. So, now I will move on to the next use case.

**Second use case:**

Patient using the application to make an appointment:

1. The patient logs into the application using his/her patient ID.
2. The patient goes to the appointment tab to make an appointment with a specialized doctor.
3. The tab shows him the list of doctors working at the hospital, their related information such as name, designation, office hours, etc. which is stored in the database at the back end.
4. The patient selects the doctor to make an appointment and is asked to pay for the appointment either through the application or at the reception.
5. The patient makes the payment through application using his/her credit card and the appointment is confirmed and patient provided with an appointment number.
6. The application will remind the patient of the appointment depending on the reminder setting in the application set by the user.

From step 2 of this use case, I can see that the **Patient entity** is related to a **Doctor entity** through an **Appointment entity** which is an associative entity. From step 4, I can see that the patient must pay his bills at the hospital so, we have the **Billing entity** related to the **Patient entity**. Now that I have identified four entities and their relationships from this second use case, I will now number and create some business rules.

1. Each patient may have many appointments; each appointment has only one patient.

This business rule is inferred from step 2 of this use case. A patient can book multiple appointments at the hospital with different or even the same doctor. A new patient will have no appointments. But an appointment will always have a must one patient.

1. Each doctor may have many appointments; each appointment has only one doctor.

This business rule is also inferred from step 2 of this use case. A doctor can have many appointments at a time. But if the doctor is new to the hospital, then he may also have no appointments at all. However, each appointment will always have a must one doctor.

1. Each patient must have a billing; each billing must have a patient.

This business rule is inferred from step 4 of this use case. Each patient must pay his / her bills at the hospital. The bills could be for labs, surgeries, appointments, etc. So, a patient must have a billing information for use by the hospital. Likewise, each billing information must have a patient related to it. A doctor at the hospital could also be a patient and he could also have a billing.

**Third use case:**

Patient wants to look up his/her medical history to check his/her history of treatment at the hospital and for his/her medication.

1. The patient logs into the application using his/her patient ID.
2. The patient goes to the patient history tab in the application.
3. The patient checks his medical history stored in the database.
4. The history will show each appointment ID, the doctor with whom the appointment was made, the patient medical condition, diagnosis, the medication prescribed by the doctor.
5. The patient can then log out once they have the required information.

From step 3 of this use case, we can see that an **Appointment entity** is related to a **Diagnosis** entity. There are no other entities or relationships that can be inferred from this use case. The business rule considering this new entity is as follows:

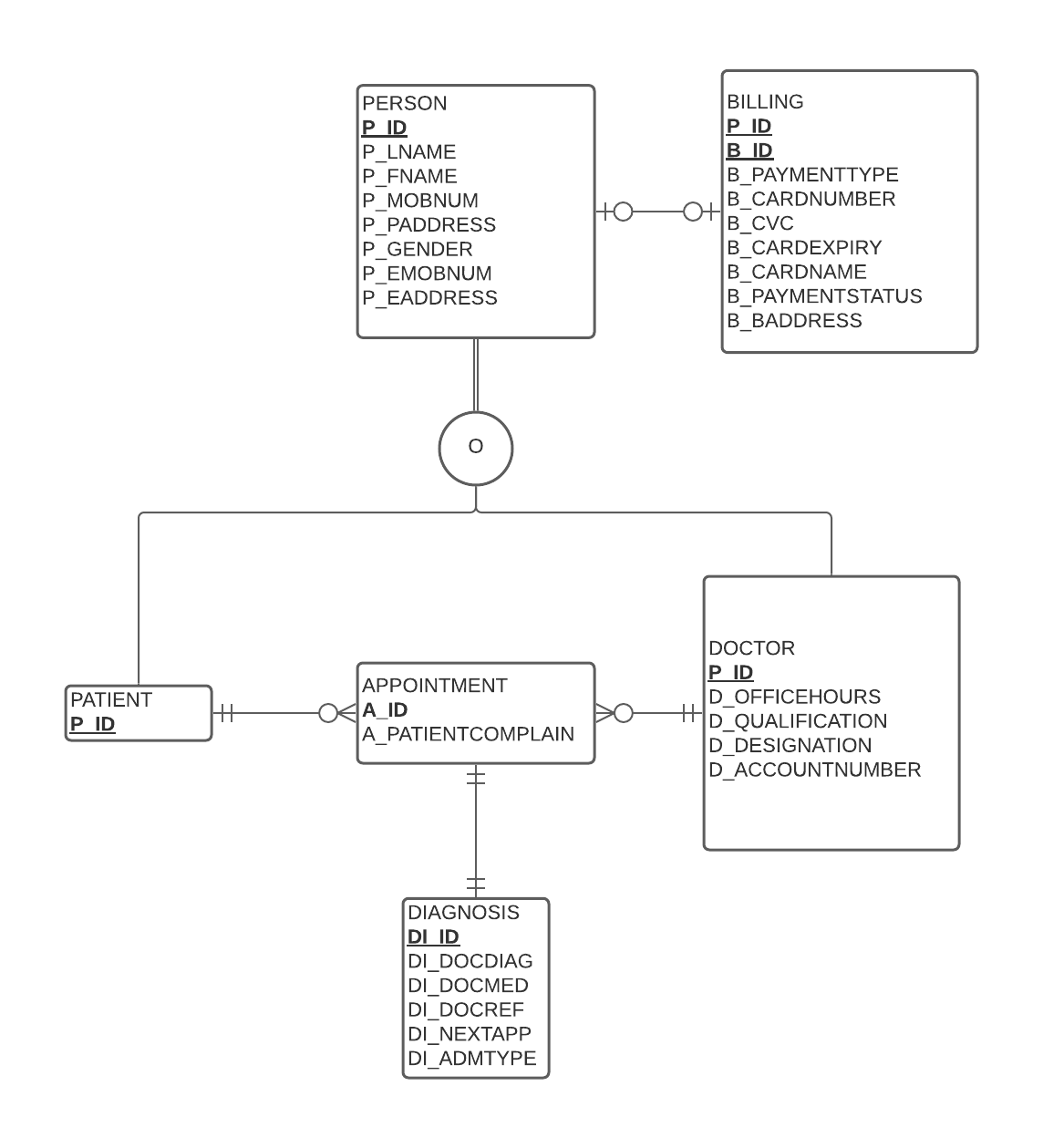
1. An appointment will have only one diagnosis; a diagnosis will have only one appointment.

An appointment between a doctor and a patient will always have a diagnosis provided by the doctor. Likewise, a diagnosis will only happen if an appointment was made with a doctor.

Therefore, now I have 4 structural rules for my database which are as follows:

1. Each patient may have many appointments; each appointment has only one patient.
2. Each doctor may have many appointments; each appointment has only one doctor.
3. Each patient must have a billing; each billing must have a patient.
4. An appointment will have only one diagnosis; a diagnosis will have only one appointment.

**Initial Health Matters ERD**



The ERD shown above implements the above **four** business rules. The third business rule involves a **patient** entity and a **billing** entity which I have implemented in the above ERD between a **person** entity and a **billing** entity. Although the business rule still does hold true, but it is ambiguous as there is no direct relation shown in the ERD between the **patient** and **billing** entity. Therefore, the third business rule after edition is now as follows:

1. A person may have a billing; a billing may have a person.

I have implemented a hierarchical relationship in the database of the hospital management system. All the staff (such as doctors, nurses, etc.) and the patients at the hospitals will have some common attributes. These are implemented in the entity of **Person**. A person could be a doctor or a patient or both at the hospital which is implemented by the overlapping complete constraint in the ERD. A person can have a **Billing** entity based on the fact if he / she has any bills related to the hospital. A patient makes an appointment with a doctor and the doctor provides a diagnosis to the patient related to his / her medical condition.

**Revised Database Rules after Implementation of Specialization Generalization Relationships**

The final **four** business rules are as follows:

1. Each patient may have many appointments; each appointment has only one patient.
2. Each doctor may have many appointments; each appointment has only one doctor.
3. A person may have a billing; a billing may have a person.
4. An appointment will have only one diagnosis; a diagnosis will have only one appointment.

**Summary and Reflection:**

My database is for a hospital management system with priority to the patients for whom this application is being created. This application will also go on to include the hospital staff afterwards. What happens normally is that patients go to the hospitals, and they have a lot of paperwork to deal with such as medical history, medicines, bills, lab reports, etc. The purpose of the application is to keep a track of everything and provide billing and other services as well. The information is also visible to the patient in the application so even if they lose the paperwork, it is all there in the database of the application and accessible to them. Also, all the paperwork will be having the digital signatures of the doctors, lab staff etc. so that the information can be validated easily when used for other purposes.

The structural database rules and ERD for my database design contain the important entities of Person, Billing, Doctor, Patient, Appointment and Diagnosis, as well as relationships between them.

I had started from three very simple use cases but ended up making 4 business rules and then 6 entities based on those rules. The activity has shown me that even a very small and simple use case could require many entities to implement it and could involve complex relationships.

The database will be quite complex, containing all the information of the patient such as their personal, billing, history, and other such information. Handling such a large amount of information could be quite complex. In addition, I also intend to extend the app usage to doctors, administrators, and then other staff at the hospital. The app will also contain their information such as their timing, salaries, and other information of hospital assets.

However, the complexity of the database does not discourage me, and I am excited to get started with the project as soon as possible so that I can learn more about databases and how to implement them in practical situations. Any feedback on making this application better is appreciated.