

# Cover Page

Hospital Management System

Name: Anisah Chowdhury

ID: 921677676

Github: AnisahC

## Table of Contents

<b>Cover Page.....</b>	<b>1</b>
<b>Table of Contents.....</b>	<b>1</b>
<b>Project Description.....</b>	<b>2</b>
<b>Functional Database Requirements.....</b>	<b>3</b>
<b>Non-functional Database Requirements.....</b>	<b>7</b>
<b>Entity Set Diagram.....</b>	<b>9</b>
<b>Entity Set Descriptions.....</b>	<b>10</b>
<b>Enhanced Entity-Relationship (EER) Diagram.....</b>	<b>14</b>
<b>Normalization Techniques Used.....</b>	<b>15</b>

# Project Description

My motivation behind creating a hospital management system stems from a personal desire of privacy and confidentiality. As I grow older, I want to have greater control over my own health records and results. I believe many patients would like that for themselves as well. Instead of having physical mail that can be opened by the wrong person, a management system can have login software that protects the privacy of its patients. Additionally, the system improves efficiency by making it easier for healthcare staff to obtain records and process patient intake seamlessly.

The database will manage the following: patient personal information, relevant providers, past check-ups and appointments, a calendar for keeping track of all appointments, and health records/lab results.

One unique feature I plan to include in my database is a login system that separates the parent from the child once the system recognizes that the child is 16 or older. This way, patient confidentiality is protected so more people are likely to get checked out. The database also allows for multiple patients to be grouped together for example: parents with young children. Or adults who have elderly parents who cannot access the information on their own and have consented to their kids taking control.

Two software tools that would benefit are Epic Systems and Accolade. These companies both specialize in the healthcare industry and aid hospitals in managing information. A database would allow them to accommodate more patients while keeping data concise. Instead of dealing with multiple tabs and links, data will be easier to input and retrieve.

The first use case is if a patient needs to cancel an appointment, they can simply notify the staff via the app. They don't need to call the hospital and be put on hold just for a simple cancellation. The second use case is doctors now have a central place to look at all their patients' records. Let's say a recurring patient calls out of nowhere with an inquiry, the doctor can use the database to look up medical records of the specific patient and answer their questions. The third case is if a patient is switching providers, they can simply transfer over all their personal information/medical records instead of having to re-enter the information.

# Functional Database Requirements

1. Patient
  - 1.1. A patient shall have a unique patient ID
  - 1.2. A patient shall have an account
  - 1.3. A patient shall have a medical record
  - 1.4. A patient shall have a DOB
  - 1.5. A patient shall have an emergency contact ID
  - 1.6. A patient shall have one and only one doctor
  - 1.7. A patient shall have many nurses
2. Parent
  - 2.1. A parent is also a patient
  - 2.2. A parent shall have a unique parent ID
  - 2.3. A parent shall have one or more children
  - 2.4. A parent shall have an occupation
  - 2.5. A parent shall have a relationship to child
3. Child
  - 3.1. A child is also a patient
  - 3.2. A child shall have at least one parent
  - 3.3. A child shall have a patient ID
  - 3.4. A child shall have a parent ID
  - 3.5. A child shall have a grade level
4. Account
  - 4.1. An account shall be owned by one and only one patient
  - 4.2. An account shall be assigned a role (parent, child, or neither)
  - 4.3. An account shall have an account ID
  - 4.4. An account shall have a patient ID
  - 4.5. An account shall have an address on file
  - 4.6. An account shall have a creation date
5. Doctor
  - 5.1. A doctor shall have a doctor ID
  - 5.2. A doctor shall be assigned to many appointments

- 5.3. A doctor shall have a first name
- 5.4. A doctor shall have a last name
- 5.5. A doctor shall have many specializations
- 5.6. A doctor can have a supervisor who is also a doctor
- 5.7. A doctor shall have many patients
- 6. Nurse
  - 6.1. A nurse shall have many patients
  - 6.2. A nurse shall have a nurse\_id
  - 6.3. A nurse shall have a first name
  - 6.4. A nurse shall have a last name
  - 6.5. A nurse shall have multiple specializations
  - 6.6. A nurse shall be assigned many rooms
- 7. Appointment
  - 7.1. An appointment shall have a patient\_id
  - 7.2. An appointment shall have a doctor\_id
  - 7.3. An appointment shall have one and only one patient
  - 7.4. An appointment shall have a date and time
  - 7.5. An appointment should have a reason for visit
- 8. Medical Records
  - 8.1. A medical record shall belong to one and only one patient
  - 8.2. A medical record shall have a patient ID
  - 8.3. A medical record shall have a date
  - 8.4. A medical record shall have medical history
  - 8.5. A medical record shall have a doctor ID
- 9. Lab Test
  - 9.1. A lab test shall belong to one and only one patient
  - 9.2. A lab test shall have a type
  - 9.3. A lab test shall have a patient ID
  - 9.4. A lab test shall have a result
  - 9.5. A lab test shall have a doctor ID
- 10. Emergency Contact

- 10.1. An emergency contact can be associated with many patients
- 10.2. An emergency contact shall have a patient ID
- 10.3. An emergency contact shall have a full name
- 10.4. An emergency contact shall have a phone number
- 10.5. An emergency contact shall have a relationship to patient
- 11. Bill
  - 11.1. A bill belongs to one and only one account
  - 11.2. A bill shall have a bill ID
  - 11.3. A bill shall have a patient ID
  - 11.4. A bill shall have a payment ID
  - 11.5. A bill shall have an appointment ID
  - 11.6. A bill shall have a due date
- 12. Department
  - 12.1. A department shall have a unique ID
  - 12.2. A department shall have a name
  - 12.3. A department shall have a room ID
  - 12.4. A department shall have a phone number
  - 12.5. A department can contain many rooms
- 13. Rooms
  - 13.1. A room shall have a unique ID
  - 13.2. A room is assigned to one department
  - 13.3. A room shall have one and only one location
  - 13.4. A room shall have one and only one number
  - 13.5. A room shall be assigned to one nurse
- 14. Specialization
  - 14.1. A specialization has a unique ID
  - 14.2. A specialization has a name
  - 14.3. A specialization shall be performed by many nurses
  - 14.4. A specialization shall be performed by many doctors
- 15. Diagnosis
  - 15.1. A diagnosis has a unique ID

- 15.2. A diagnosis shall be given to many patients
- 15.3. A diagnosis shall have a name
- 16. Prescription
  - 16.1. A prescription shall have a unique ID
  - 16.2. A prescription shall be for one and only one patient
  - 16.3. A prescription shall have a patient ID
  - 16.4. A prescription shall have a doctor ID
  - 16.5. A prescription shall have a medication ID
  - 16.6. A prescription shall have a dosage
- 17. Medication
  - 17.1. A medication shall have a unique ID
  - 17.2. A medication shall have a name
  - 17.3. A medication shall have instructions
- 18. Payment
  - 18.1. A payment shall have a unique ID
  - 18.2. A payment shall have a credit card number
  - 18.3. A payment shall have a full name
- 19. Allergies
  - 19.1. An allergy shall have a patient ID
  - 19.2. An allergy shall have a severity
  - 19.3. An allergy shall have an allergen
- 20. Certifications
  - 20.1. A certification shall have a unique ID
  - 20.2. A certification shall be obtained by many doctors
  - 20.3. A certification shall be obtained by many nurses
  - 20.4. A certification shall have a name
  - 20.5. A certification shall have a description

# Non-functional Database Requirements

1. Performance
  - 1.1. The database shall support concurrent sessions without noticeable delay
  - 1.2. The database shall be able to retrieve information no matter how miniscule
  - 1.3. The database shall be able to handle large amounts of information without crashing
2. Security
  - 2.1. The database shall enforce 2FA for patients
  - 2.2. The database shall have restricted access depending on the user role
  - 2.3. The database shall require a password for patients and hospital staff
3. Scalability
  - 3.1. The database shall support 100,000 accounts bi-monthly
  - 3.2. The database shall be able to support high traffic volumes without crashing
  - 3.3. The database shall be able to hold copious amounts of data
4. Capability
  - 4.1. The database shall be able to identify certain patients based on allergy, diagnosis, etc.
  - 4.2. The database shall provide real-time description for the most up-to-date information
  - 4.3. The database shall support retrieval of medical records, lab tests, allergies, etc. of patients
5. Environmental
  - 5.1. The database shall run on all operating systems to ensure maximum compatibility
  - 5.2. The database shall be backed up into a cloud to save important information in case of a black out
  - 5.3. The database shall be transferable via a device driver such as a USB
6. Coding Standard
  - 6.1. The database shall be properly commented to describe purpose
  - 6.2. The database shall follow standard coding rules such as cleanliness, camel case, and proper naming conventions
  - 6.3. The database shall be in SQL queries

7. Storage

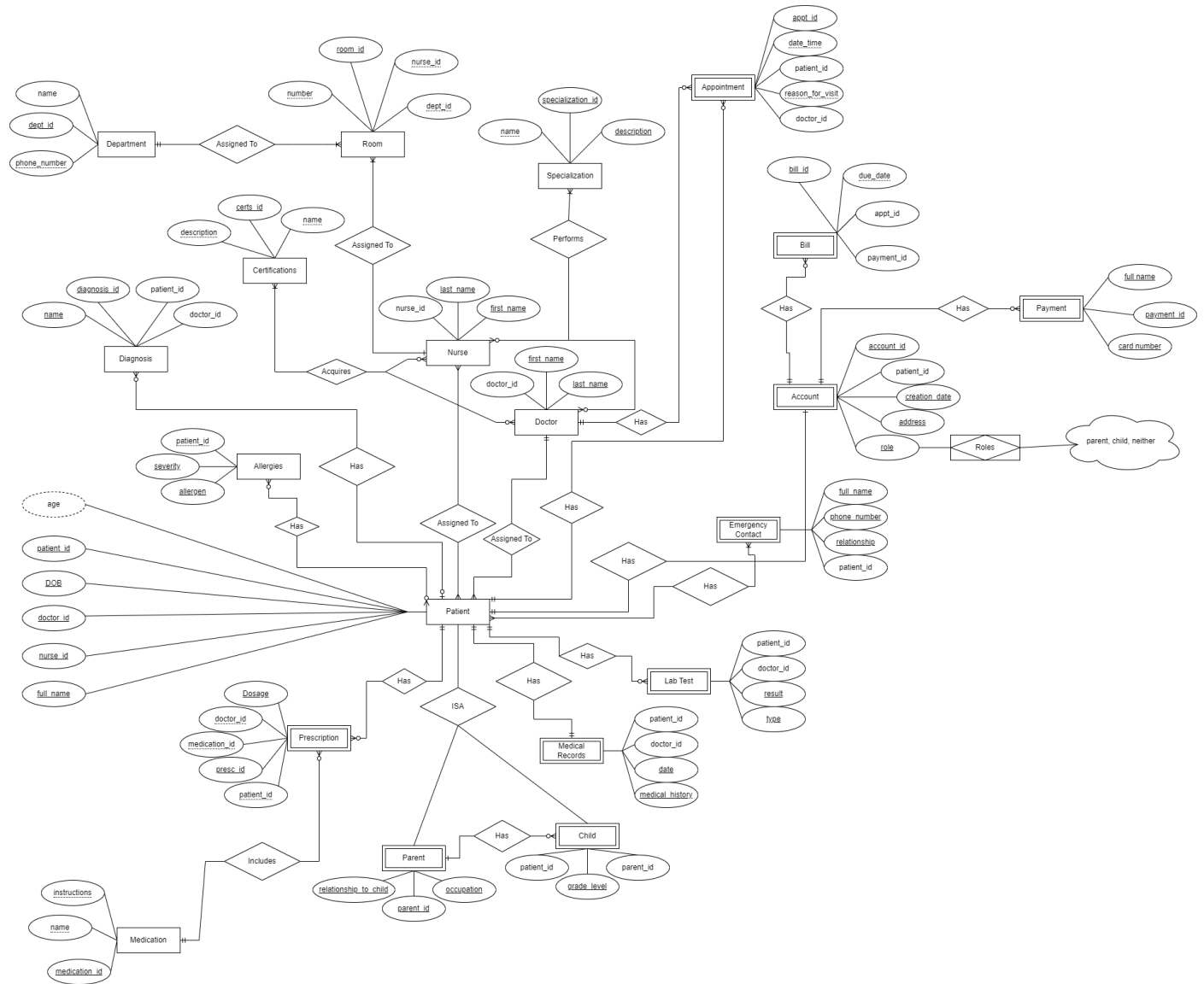
- 7.1. The database should be able to hold at least 10MB of data per patient
- 7.2. The database needs a buffer to avoid memory leaks
- 7.3. The database should archive data older than 7 years but still be able to query it

8. Privacy

- 8.1. The database shall provide access to children's account if a parent is logging in
- 8.2. The database shall comply with HIPAA standards to ensure patient privacy
- 8.3. The database should keep a log of anyone accessing an account



# Entity Set Diagram



# Entity Set Descriptions

## 1. Patient (strong)

- a. patient\_id: numeric, primary key
- b. account\_id: numeric, foreign key (references Account entity)
- c. DOB: date, composite
- d. emergency\_contact\_id: numeric, foreign key (references Emergency Contact entity)
- e. medical\_record\_id: numeric, foreign key (references Medical Record entity)
- f. doctor\_id: numeric, foreign key (references Doctor entity)
- g. nurse\_id: numeric, foreign key (references Nurse entity)
- h. age: numeric, derived
- i. full\_name: alphanumeric

## 2. Parent (weak)

- a. patient\_id: numeric, foreign key (references Patient entity)
- b. parent\_id: numeric, primary key
- c. occupation: alphanumeric
- d. relationship\_to\_child: alphanumeric

## 3. Child (weak)

- a. patient\_id: numeric, foreign key (references Patient entity)
- b. grade\_level: numeric
- c. parent\_id: numeric, foreign key (references Parent entity)

## 4. Account (weak)

- a. account\_id: numeric, primary key
- b. patient\_id: numeric, foreign key (references Patient entity)
- c. creation\_date: date
- d. role: alphanumeric

## 5. Doctor (strong)

- a. doctor\_id: numeric, primary key
- b. last\_name: alphanumeric
- c. first\_name: alphanumeric

**6. Nurse (strong)**

- a. nurse\_id: numeric
- b. first\_name: alphanumeric
- c. last\_name: alphanumeric

**7. Appointment (weak)**

- a. appt\_id: numeric, primary key
- b. patient\_id: numeric, foreign key (references Patient entity)
- c. doctor\_id: numeric, foreign key (references Doctor entity)
- d. reason\_for\_visit: alphanumeric
- e. date\_time: datetime

**8. Medical Records (weak)**

- a. patient\_id: numeric, foreign key (references Patient entity)
- b. doctor\_id: numeric, foreign key (references Doctor entity)
- c. date: numeric
- d. medical\_history: alphanumeric

**9. Lab Test (weak)**

- a. patient\_id: numeric, foreign key (references Patient entity)
- b. doctor\_id: numeric, foreign key (references Doctor entity)
- c. result: alphanumeric
- d. type: alphanumeric

**10. Emergency Contact (weak)**

- a. patient\_id: numeric, foreign key (references Patient entity)
- b. full\_name: alphanumeric
- c. phone\_number: numeric
- d. relationship: alphanumeric

**11. Bill (weak)**

- a. bill\_id: numeric, primary key
- b. appt\_id: numeric, foreign key (references appointment entity)
- c. payment\_id: numeric, foreign key (references payment entity)
- d. due\_date: date

**12. Department (strong)**

- a. dept\_id: numeric, primary key
- b. phone\_number: numeric
- c. name: alphanumeric

### **13. Rooms (strong)**

- a. room\_id: numeric, primary key
- b. nurse\_id: numeric, foreign key (references nurse entity)
- c. number: numeric
- d. dept\_id: numeric, foreign key (references department entity)

### **14. Specialization (strong)**

- a. specialization\_id: numeric, primary key
- b. name: alphanumeric
- c. description: alphanumeric

### **15. Diagnosis (strong)**

- a. diagnosis\_id: numeric, primary key
- b. patient\_id: numeric, foreign key (references patient entity)
- c. doctor\_id: numeric, foreign key (references doctor entity)
- d. name: alphanumeric

### **16. Prescription (weak)**

- a. presc\_id: numeric, primary key
- b. doctor\_id: numeric, foreign key (references doctor entity)
- c. medication\_id: numeric, foreign key (references medication entity)
- d. patient\_id: numeric, foreign key (references patient entity)
- e. dosage: alphanumeric

### **17. Medication (strong)**

- a. medication\_id: numeric, primary key
- b. name: alphanumeric
- c. instructions: alphanumeric

### **18. Payment (weak)**

- a. payment\_id: numeric, primary key
- b. full\_name: alphanumeric
- c. card\_number: numeric

- d. account\_id: numeric, foreign key (references Account entity)

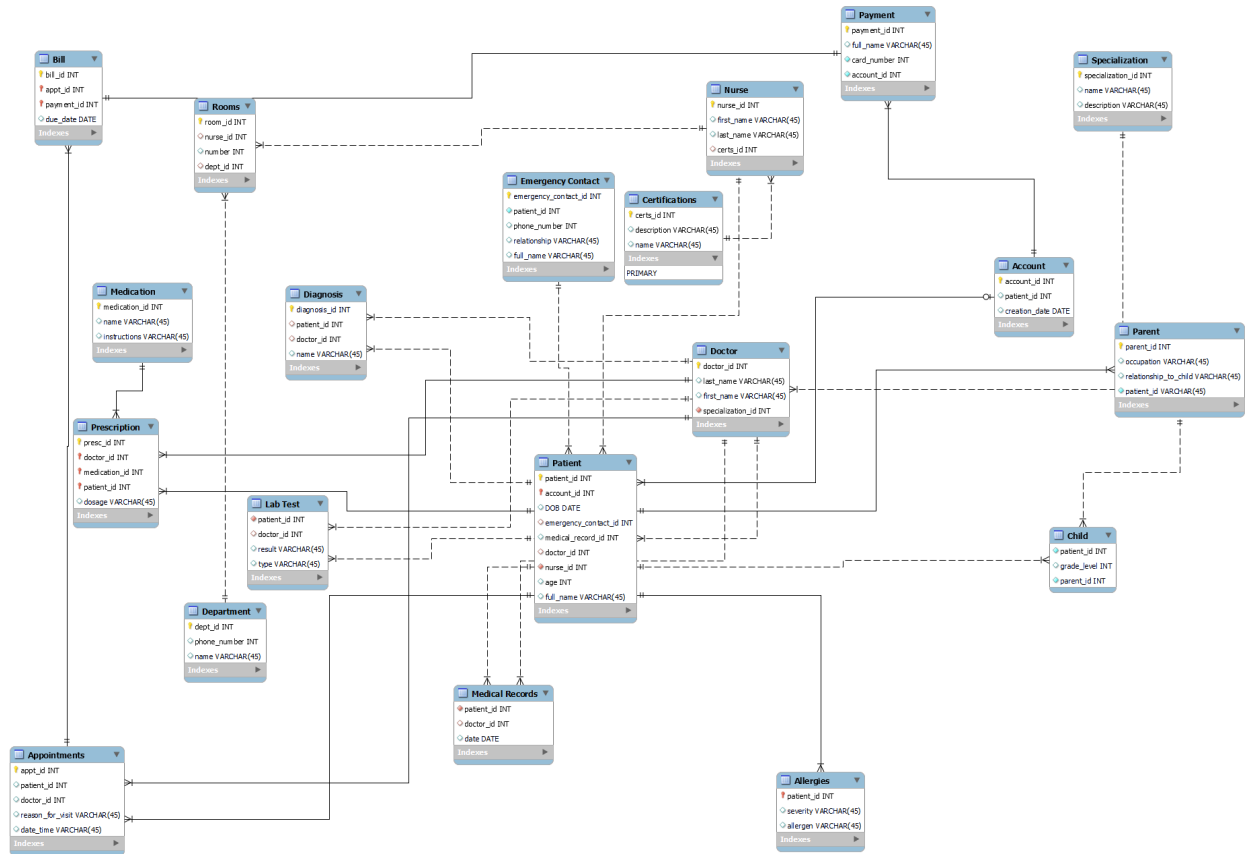
**19. Allergies (strong)**

- a. patient\_id: numeric, foreign key (references patient entity)
- b. severity: alphanumeric
- c. allergen: alphanumeric

**20. Certifications (strong)**

- a. certs\_id: numeric, primary key
- b. description: alphanumeric
- c. name: alphanumeric

# Enhanced Entity-Relationship (EER) Diagram



# Normalization Techniques Used

I applied normalization in my database to increase efficiency and make sure the data wasn't redundant. Below are the different levels and examples of how I applied them.

## 1NF

- Goal: Eliminate non-atomic values and have unique columns.
- Example: In the Patient table, attributes such as patient\_id, full\_name, and DOB were defined as atomic values. Instead of making an attribute called emergency\_contacts, I made a new table that holds emergency\_contact\_id which holds all the necessary information. This ensures each data is stored in a unique cell, making it easier to query and manipulate.

## 2NF

- Goal: Must follow 1NF, and all data must depend on the primary key.
- Example: In the Account table, I originally put payment\_id along with card\_number. This violates normalization because the card\_number needs payment\_id AND account\_id which doesn't make it fully dependent on the primary key. To solve this, I created a new Payment table that holds card\_number. In the Account table I made payment\_id a foreign key. This specifies the data more and makes querying straight forward.

## 3NF

- Goal: Must follow 2NF, and primary keys must define all non-key columns
- Example: In the Doctor table, I didn't leave specialization to be its own attribute because it does not directly apply to doctor\_id. In some cases, others can argue it does depend on the primary key but in my opinion, I can make it more specific. I created a Specialization table with a specialization\_id primary key to store all the information like name and description. This way, the information can be retrieved through fewer inquiries.