VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

REPORT

HOSPITAL DATA MANAGEMENT

Course: Principle of Database Management
Lecturer: Assoc. Prof. Nguyen Thi Thuy Loan
Group: 07
Topic: Hospital Data Management

Semester 01, 2023 - 2024

TABLE OF CONTENTS

GitHub repository: Group 07

LIST OF FIGURES

GitHub repository: Group 07

I. GROUP'S INFORMATION

Group 07 is modified by 05 members with their assigned responsibilities. Here is the group member list:

No.	Full name	ID	Responsibility
1.	PHẠM LÊ THANH NHÀN	ITDSIU20073	Team leader
2.	NGUYỄN TRẦN QUỐC THỊNH	ITITIU20312	Team member
3.	HUỲNH NGUYỄN QUỐC HƯNG	ITCSIU21183	Team member
4.	NGUYỄN QUỐC HƯNG	ITITIU21211	Team member
5.	BÀNH VĨNH THUẬN	ITITIU21323	Team member

II. GENERAL INFORMATION OF PROJECT

1. Topic

Topic 05: Hospital Data Management

Hospitals have unique data requirements. Not only do they have to maintain the medical records of their patients, but they also must manage their staff and its multiple departments. You can solve the data-related problems of hospitals by creating a DBMS solution.

First, you should assign unique IDs to the patients and store the relevant information under the same. You'll have to add the patient's name, personal details, contact number, disease name, and the treatment for the 9 patients are going through. You'll also have to mention under which hospital department the patient is (such as cardiac, gastro, etc.).

After that, you should add information about the hospital's doctors. A doctor can treat multiple patients, and he/she would have a unique ID as well. Doctors would also be classified into different departments. Patients would get admitted into rooms, so you'll need to add that information to your database too. Apart from that, there would be distinct rooms (ICUs and Operation Theaters) in the hospital.

Then, you'd have to add the information of ward boys and nurses working in the hospital and assigned to different rooms. You can start with a small hospital and expand it as you move on. Make sure that the data is easily understandable and accessible.

2. Abstract

The project is expected to help each member of our group understand the main goal of the Principle of Database Management course. In this project, a database system and a basic app are developed to manage data for patients and doctors within a hospital and allow them to access and communicate together.

The database system must store information clearly about patients, doctors (users), and facilities. Besides, the app is built with a simple interface in which users have their own accounts and are able to log in and interact. Through the app, while patients can book appointments and view their medical records, doctors can also receive requests, arrange them, and review their patient' s medical records.

Hence, this project may try to help communication between patients and doctors based on the database system and the app.

GitHub repository: Group 07

3. Technologies used

Frontend: React.js

Backend: Node.js, Express

Database: MySQL

4. Core Data Classes

No.	Class	Attributes
1.	Department	**dep_id** INT PRIMARY KEY, **dep_name** VARCHAR(100);
2.	Account	**userid** BIGINT PRIMARY KEY, **username** VARCHAR(100) UNIQUE, **password** VARCHAR(250), **usertype** VARCHAR(50);
3.	Doctor	**doc_id** INT PRIMARY KEY, **doc_lastname** VARCHAR(100), **doc_firstname** VARCHAR(100), **doc_phone** VARCHAR(10), **doc_email** VARCHAR(200), **dep_id** INT, **userid** BIGINT;
4.	Nurse	**n_id** INT PRIMARY KEY, **n_lastname** VARCHAR(100), **n_firstname** VARCHAR(100), **n_phone** VARCHAR(10), **n_email** VARCHAR(100), **dep_id** INT, **userid** BIGINT;
5.	Patient	**p_id** INT PRIMARY KEY, **p_lastname** VARCHAR(100), **p_firstname** VARCHAR(100) **p_phone** VARCHAR(10), **p_email** VARCHAR(100), **p_province** VARCHAR(250), **p_dob** DATE, **p_biogender** VARCHAR(1), **userid** BIGINT;
6.	Appointment	**a_id** INT PRIMARY KEY, **a_date** DATE, **a_time** TIME, **p_id** INT, **doc_id** INT;
7.	MedicalRecord	**mr_id** INT PRIMARY KEY, **mr_date** DATE, **symptom** VARCHAR(50), **disease** VARCHAR(50), **treatment** VARCHAR(50), **p_id** INT, **doc_id** INT;
8.	PatientRoom	**room_no** VARCHAR(10) PRIMARY KEY, **bed_num** INT,

GitHub repository: Group 07

		r_type VARCHAR(1);
9.	PatientRoomArrangement	**id** BIGINT PRIMARY KEY, **room_no** VARCHAR(10), **n_id** INT, **p_id** INT;

5. Relationship Rules

No.	Rule
1.	A department has many doctors and nurses. Every doctor or nurse works in only a department.
2.	A patient can book many appointments. Each appointment corresponds to a patient only.
3.	A doctor treats many patients. Each patient is treated by a doctor only.
4.	A nurse serves many patient rooms. Each patient room is served by a nurse only.
5.	There can be many patients in a room based on its number of beds. A patient is only in a room.
6.	A patient has only a medical record. Doctors are able to update or view all their patients' medical records.
7.	Every doctor or nurse or patient has only an app account. An account corresponds to a user only.

III. TIMEFRAME

1. Contribution

No.	Full name	Responsibilities	Contribution
1	PHẠM LÊ THANH NHÀN	Building database Developing user interface	100%
2	NGUYỄN TRẦN QUỐC THỊNH	Building database Developing user interface	100%
3	HUỲNH NGUYỄN QUỐC HƯNG	Developing user interface	100%
4	NGUYỄN QUỐC HƯNG	Developing user interface	100%
5	BÀNH VĨNH THUẬN	Building database	100%

2. Responsibilities

Stage	Task	Responsible member	Deadline
Planning & Proposal	Define the main goal of this course	All	02/10
	Choose topic		
	Topic detail understanding		
	Write & submit the proposal	Thanh Nhan	06/10
Processing & Midterm Report	Design & explain Entity Relationship Diagram (ERD)	Quoc Thinh Thanh Nhan	15/10
	Design & explain Database Relationship Diagram	Quoc Thinh Vinh Thuan	22/10

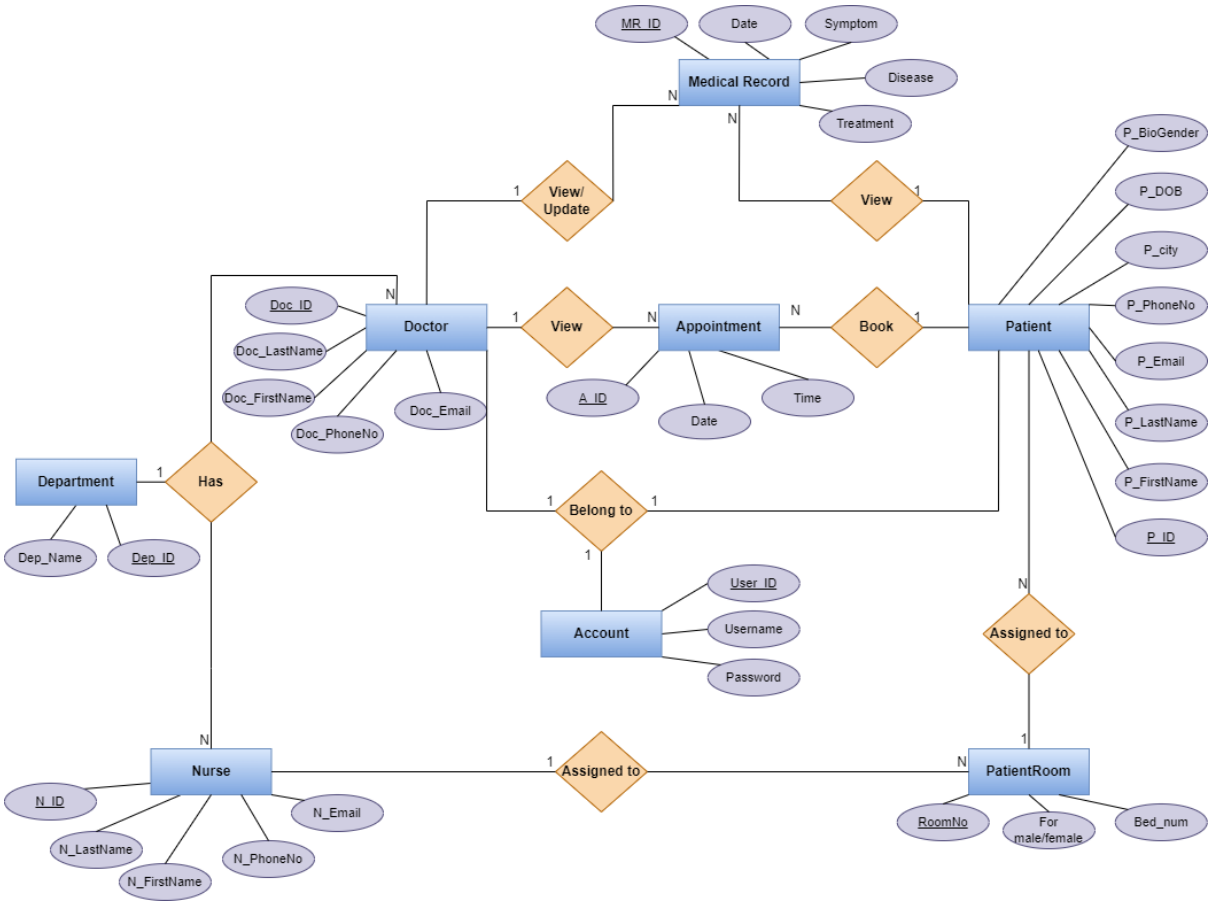
GitHub repository: Group 07

	Complete & submit the midterm report	Vinh Thuan	26/10
	Build & explain app	H.N.Q.Hung N.Q.Hung Quoc Thinh	02/12
Final	Complete & submit the final report	Thanh Nhan Vinh Thuan	09/12
	PowerPoints	H.N.Q.Hung N.Q.Hung	16/12
	Presentation	All	

IV. PROJECT ANALYSIS

1. Database System

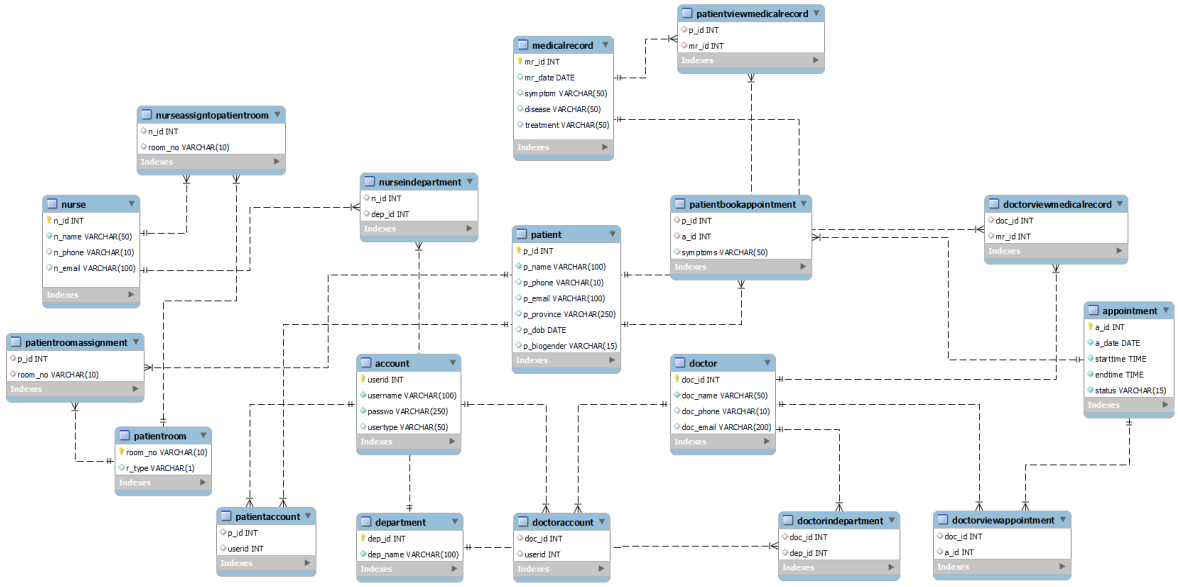
1.1. Entity Relationship Diagram



Figures 1. Entity-Relationship Diagram

GitHub repository: [Group 07]

1.2.Database Diagram



Figures 2. Database Diagram

1.3. Database Design and Data Building in MySQL

a. Department Table:

```
CREATE TABLE Department (
	dep_id INT NOT NULL AUTO_INCREMENT,
	dep_name VARCHAR(100) NOT NULL,
	PRIMARY KEY (dep_id));
```

```
INSERT INTO Department (dep_name)
VALUES
	('Accident and emergency'),
	('Breast Screening'),
	('Chaplaincy'),
	('Diagnostic Imaging'),
	('General Surgery'),
	('Microbiology');
```

b. Account Table:

```
CREATE TABLE Account (
	userid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
	username VARCHAR(100) NOT NULL UNIQUE,
	passwo VARCHAR(250) NOT NULL,
	usertype VARCHAR(50));
```

Each doctor and patient have an account as user to log into the app. Each user has their own user identity. They register to the app by creating their user's name and password. They also must choose their roles as doctor or patient. The 'username' attribute is unique in that no couple of users has the same username (we use "email" for username).

```
INSERT INTO Account (username, passwo, usertype)
VALUES
	('lhanh123@gmail.com', 'lhanh123', 'Doctor'),
	('hqbao123@gmail.com', 'hqbao123', 'Doctor'),
	('tqduc123@gmail.com', 'tqduc123', 'Doctor'),
	('nhgiang123@gmail.com', 'nhgiang123', 'Nurse'),
	('hthuyen123@gmail.com', 'hthuyen123', 'Nurse'),
	('htngan123@gmail.com', 'htngan', 'Nurse'),
	('tmlong123@gmail.com', 'tmlong123', 'Patient'),
```

GitHub repository: [Group 07]

```
        ('lnminh123@gmail.com', 'lnminh123', 'Patient'),
        ('pbngoc123@gmail.com', 'pbngoc123', 'Patient');
```

c. Doctor Table:

```
CREATE TABLE Doctor (
        doc_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        doc_name VARCHAR(50) NOT NULL,
        doc_phone VARCHAR(10),
        doc_email VARCHAR(200) UNIQUE);
```

```
INSERT INTO Doctor (doc_name, doc_phone, doc_email)
VALUES
        ('Le Hien Anh', '0123456789', 'lhanh123@gmail.com'),
        ('Hoang Quoc Bao', '0123456788', 'hqbao123@gmail.com'),
        ('Tran Quoc Duc', '0123456777', 'tqduc123@gmail.com');
```

d. DoctorInDepartment Table:

```
CREATE TABLE DoctorInDepartment (
        doc_id INT,
        dep_id INT,
        FOREIGN KEY (dep_id) REFERENCES Department(dep_id),
        FOREIGN KEY (doc_id) REFERENCES Doctor(doc_id));
```

```
INSERT INTO DoctorInDepartment (dep_id, doc_id)
VALUES (1,1),
               (1,2),
         (1,3),
         (2,5),
         (3,4),
         (4,9),
         (5,10),
         (6,6),
         (4,7),
         (6,8);
```

e. DoctorAccount Table:

```
CREATE TABLE DoctorAccount (
        doc_id INT,
        userid INT,
        FOREIGN KEY (userid) REFERENCES Account(userid),
        FOREIGN KEY (doc_id) REFERENCES Doctor(doc_id));
```

```
INSERT INTO DoctorAccount(doc_id, userid)
VALUES
        (1,1),
        (2,2),
        (3,3);
```

f. Nurse Table:

```
CREATE TABLE Nurse (
        n_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        n_name VARCHAR(50) NOT NULL,
        n_phone VARCHAR(10),
        n_email VARCHAR(100));
```

```
INSERT INTO Nurse (n_name, n_phone, n_email)
```

GitHub repository: [Group 07]

```
VALUES
	('Nguyen Hoang Giang', '0123456001', 'nhgiang123@gmail.com'),
	('Ha Thanh Huyen', '0123456033', 'hthuyen123@gmail.com'),
	('Huynh Tuyet Ngan', '0123456023', 'htngan123@gmail.com'),
	('Do Minh Anh', '056123784', 'dmanh123@gmail.com'),
	('Le Linh Khanh', '056784512', 'llkhanh123@gmail.com'),
	('Hoang Minh Ngan', '05647812', 'hmngan123@gmail.com'),
	('Luu Nhat Linh', '023145678', 'lnlinh123@gmail.com'),
	('Vu Anh Ngoc', '014567823', 'vangoc123@gmail.com');
```

g. NurseInDepartment Table:

```
CREATE TABLE NurseInDepartment (
	n_id INT,
	dep_id INT,
	FOREIGN KEY (dep_id) REFERENCES Department(dep_id),
	FOREIGN KEY (n_id) REFERENCES Nurse(n_id));
```

```
INSERT INTO NurseInDepartment (n_id, dep_id)
VALUES
	(1,4),
    (2,3),
    (3,1),
    (4,4),
    (5,6),
    (6,2),
    (7,5),
    (8,5);
```

h. Patient Table:

```
CREATE TABLE Patient (
	p_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
	p_name VARCHAR(100) NOT NULL,
	p_phone VARCHAR(10),
	p_email VARCHAR(100),
	p_province VARCHAR(250),
	p_dob DATE,
	p_biogender VARCHAR(15));
```

```
INSERT INTO Patient (p_name, p_phone, p_email, p_province, p_dob, p_biogender)
VALUES
('Tran Minh Long', '0123456722', 'tmlong123@gmail.com', 'Ho Chi Minh City',
'1999-02-03', 'M'),
('Luong Nhat Minh', '0123456712', 'lnminh123@gmail.com', 'Vinh Long', '2002-03-
07', 'M'),
('Pham Bao Ngoc', '0123456784', 'pbngoc123@gmail.com', 'Ben Tre', '1997-04-08',
'F');
```

i. PatientAccount Table:

```
CREATE TABLE PatientAccount (
	p_id INT,
	userid INT,
	FOREIGN KEY (userid) REFERENCES Account(userid),
	FOREIGN KEY (p_id) REFERENCES Patient(p_id));
```

```
INSERT INTO PatientAccount(p_id, userid)
VALUES
	(1,7),
```

GitHub repository: [Group 07]

```
        (2,8),
        (3,9);
```

j. Appointment Table:

```
CREATE TABLE Appointment (
        a_id INT PRIMARY KEY,
        a_date DATE NOT NULL,
        starttime TIME NOT NULL,
        endtime TIME NOT NULL,
        status varchar(15) NOT NULL);
```

```
INSERT INTO Appointment (a_id, a_date, starttime, endtime, status)
VALUES
        (1, '2023-12-01','10:00','11:00', 'Done');
```

k. PatientBookAppointment Table:

```
CREATE TABLE PatientBookAppointment (
        p_id INT,
        a_id INT,
        symptoms varchar(50),
        FOREIGN KEY (p_id) REFERENCES Patient(p_id),
        FOREIGN KEY (a_id) REFERENCES Appointment(a_id));
```

```
INSERT INTO PatientBookAppointment (p_id, a_id, symptoms)
VALUES (1,1,'fever');
```

l. DoctorViewAppointment Table:

```
CREATE TABLE DoctorViewAppointment (
        doc_id INT,
        a_id INT,
        FOREIGN KEY (doc_id) REFERENCES Doctor(doc_id),
        FOREIGN KEY (a_id) REFERENCES Appointment(a_id));
```

m. MedicalRecord Table:

```
CREATE TABLE MedicalRecord (
        mr_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        mr_date DATE,
        symptom VARCHAR(50),
        disease VARCHAR(50),
        treatment VARCHAR(50));
```

```
INSERT INTO MedicalRecord (mr_date, symptom, disease, treatment)
VALUES ('2023-11-02','fever','dengue', 'medication'),
               ('2023-12-01','caugh','pneumonia', 'medication');
```

n. PatientViewMedicalRecord Table:

```
CREATE TABLE PatientViewMedicalRecord (
        p_id INT,
        mr_id INT,
        FOREIGN KEY (p_id) REFERENCES Patient(p_id),
        FOREIGN KEY (mr_id) REFERENCES MedicalRecord(mr_id));
```

```
INSERT INTO PatientViewMedicalRecord (p_id, mr_id)
VALUES (1,1),
        (2,2);
```

GitHub repository: [Group 07]

o. DoctorViewMedicalRecord Table:

```
CREATE TABLE DoctorViewMedicalRecord (
	doc_id INT,
	mr_id INT,
	FOREIGN KEY (doc_id) REFERENCES Doctor(doc_id),
	FOREIGN KEY (mr_id) REFERENCES MedicalRecord(mr_id));
```

```
INSERT INTO DoctorViewMedicalRecord (doc_id, mr_id)
VALUES (1,1),
	(1,2);
```

p. PatientRoom Table:

```
CREATE TABLE PatientRoom (
	room_no VARCHAR(10) NOT NULL PRIMARY KEY,
	r_type VARCHAR(1));
```

```
INSERT INTO PatientRoom (room_no, r_type)
VALUES ('R001', 'M'),
	('R002', 'M'),
	 ('R003', 'F'),
	 ('R004', 'F'),
	 ('R005', 'F'),
	 ('R006', 'M'),
	('R007', 'M'),
	 ('R008', 'M'),
	 ('R009', 'F'),
	 ('R010', 'F'),
	 ('R011', 'F');
```

q. NurseAssignToPatientRoom Table:

```
CREATE TABLE NurseAssignToPatientRoom (
	n_id INT,
	room_no VARCHAR(10),
	FOREIGN KEY (n_id) REFERENCES Nurse(n_id),
	FOREIGN KEY (room_no) REFERENCES PatientRoom(room_no))
```

```
INSERT INTO NurseAssignToPatientRoom (room_no, n_id)
VALUES ('R001', 1),
	('R002', 2),
	 ('R003', 3),
	 ('R004', 3),
	 ('R005', 4),
	 ('R006', 5),
	 ('R007', 6),
	 ('R008', 7),
	 ('R009', 7),
	 ('R010', 8),
	 ('R011', 8);
```

r. PatientRoomAssignment Table:

```
CREATE TABLE PatientRoomAssignment (
	p_id INT,
	room_no VARCHAR(10),
	FOREIGN KEY (p_id) REFERENCES Patient(p_id),
	FOREIGN KEY (room_no) REFERENCES PatientRoom(room_no));
```

GitHub repository: [Group 07]

```
INSERT INTO PatientRoomAssignment (room_no, p_id)
VALUES       ('R001', 1),
             ('R001', 2),
       ('R002', 4),
       ('R002', 5),
       ('R007', 7),
       ('R008', 10),
       ('R003', 6),
       ('R003', 3),
       ('R004', 8),
       ('R005', 9),
       ('R005', 11);
```

1.4. Normal Form Evaluation

This database is in the Second Normal Form, explained by the following table:

Normal Form	Description
1NF	There is no composite or multi-valued attribute.
2NF	All non-primary-key attribute is fully functionally dependent on the primary key.
3NF	There is no transitive dependence in all tables.

2. Database Queries

2.1. Query 1

All departments with their doctors, ordering by names of departments.

```
SELECT dep.dep_name as department,
       d.doc_name as doctor
FROM Department dep
LEFT JOIN DoctorInDepartment did
ON dep.dep_id = did.dep_id
LEFT JOIN Doctor d
ON did.doc_id = d.doc_id
ORDER BY dep.dep_name ASC;
```

department	doctor
Accident and emergency	Le Hien Anh
Accident and emergency	Hoang Quoc Bao
Accident and emergency	Tran Quoc Duc
Breast Screening	Karyn Ramirez
Chaplaincy	nhan pham
Diagnostic Imaging	Nguyen Quoc Hung Huynh
Diagnostic Imaging	Tran Quoc Thinh Nguyen
General Surgery	Vinh Thuan Banh
Microbiology	Le Thanh Nhan Pham
Microbiology	Quoc Hung Nguyen

2.2. Query 2

All departments with their nurses, ordering by names of departments.

```
SELECT dep.dep_name as department,
n.n_name as nurse
FROM Department dep
LEFT JOIN NurseInDepartment nid
ON dep.dep_id = nid.dep_id
LEFT JOIN Nurse n
ON nid.n_id = n.n_id
ORDER BY dep.dep_name ASC;
```

department	nurse
Accident and emergency	Huynh Tuyet Ngan
Breast Screening	Hoang Minh Ngan
Chaplaincy	Ha Thanh Huyen
Diagnostic Imaging	Nguyen Hoang Giang
Diagnostic Imaging	Do Minh Anh
General Surgery	Luu Nhat Linh
General Surgery	Vu Anh Ngoc
Microbiology	Le Linh Khanh

2.3. Query 3

All patients who are between 20 and 30 years old with their name, gender, date of birth and age calculated from date of birth.

GitHub repository: [Group 07]

```sql
SELECT
p_name AS Patient,
p_biogender AS Gender,
p_dob AS 'Date of Birth',
TIMESTAMPDIFF(YEAR, p_dob, CURDATE())
AS Age
FROM Patient
WHERE p_dob BETWEEN
DATE_SUB(CURDATE(), INTERVAL 30 YEAR)
AND DATE_SUB(CURDATE(), INTERVAL 20
YEAR);
```

Patient	Gender	Date of Birth	Age
Tran Minh Long	Male	1999-02-03	24
Luong Nhat Minh	Male	2002-03-07	21
Pham Bao Ngoc	Female	1997-04-08	26
Kareem Trevino	Male	2002-02-03	21
Carly Valentine	Female	1998-12-08	25
Jordan Poole	Male	1995-02-03	28
Denton Langley	Female	2000-04-02	23
Christopher Riddle	Male	1999-11-11	24
Serena Dejesus	Female	1997-03-08	26

2.4. Query 4

All rooms that has patient with room_no, patient name and nurse name

```sql
SELECT pr.room_no as Room,
       p.p_name as Patient,
       n.n_name as Nurse
FROM PatientRoom pr
RIGHT JOIN PatientRoomAssignment pra
ON pr.room_no = pra.room_no
INNER JOIN Patient p
ON pra.p_id = p.p_id
LEFT JOIN NurseAssignToPatientRoom npr
ON pr.room_no = npr.room_no
INNER JOIN Nurse n
ON npr.n_id = n.n_id;
```

Room	Patient	Nurse
R001	Tran Minh Long	Nguyen Hoang Giang
R001	Luong Nhat Minh	Nguyen Hoang Giang
R002	Brenda Summers	Ha Thanh Huyen
R002	Kareem Trevino	Ha Thanh Huyen
R003	Carly Valentine	Huynh Tuyet Ngan
R003	Pham Bao Ngoc	Huynh Tuyet Ngan
R004	Clio Galloway	Huynh Tuyet Ngan
R005	Denton Langley	Do Minh Anh
R005	Serena Dejesus	Do Minh Anh
R007	Jordan Poole	Hoang Minh Ngan
R008	Christopher Riddle	Luu Nhat Linh

3. User Interface Development

3.1. General Information

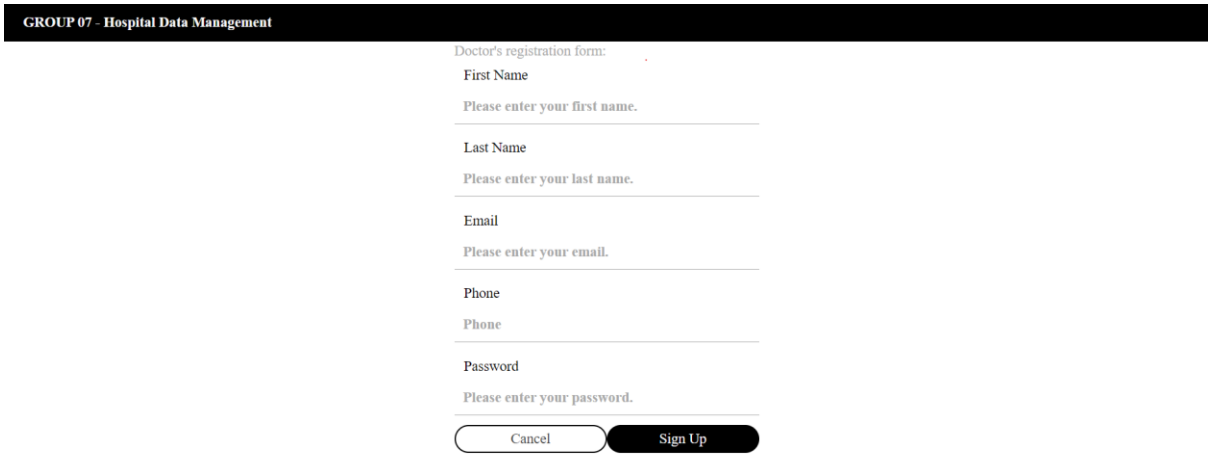
Our group tries to develop an application that connects to the database for not only get data from the database to display on the application, but also interacting with user interface, then insert the result of interactions (data) into the database.

The application provides two roles of users: Doctor and Patient, and two features for users to interact with.

3.2. Feature 1 – Register and Login

Both doctors and patients are allowed to sign up and sign in the application.

a. Doctor registration:



Figures 3. Doctor Registration

Step 1: After clicking on 'Sign Up' button, the system will check whether the email that is used for registration has exist or not by querying the database. If not, the next step will be implemented.

```
//Checks whether the email that is used for registration has exist or not
app.get('/checkIfDocExists', (req, res) => {
  let params = req.query;
  let email = params.email;
  let statement = `SELECT * FROM Doctor WHERE doc_email = "${email}"`;
  console.log(statement);
  con.query(statement, function (error, results, fields) {
    if (error) throw error;
    else {
      return res.json({
        data: results
      })
    };
  });
});
```

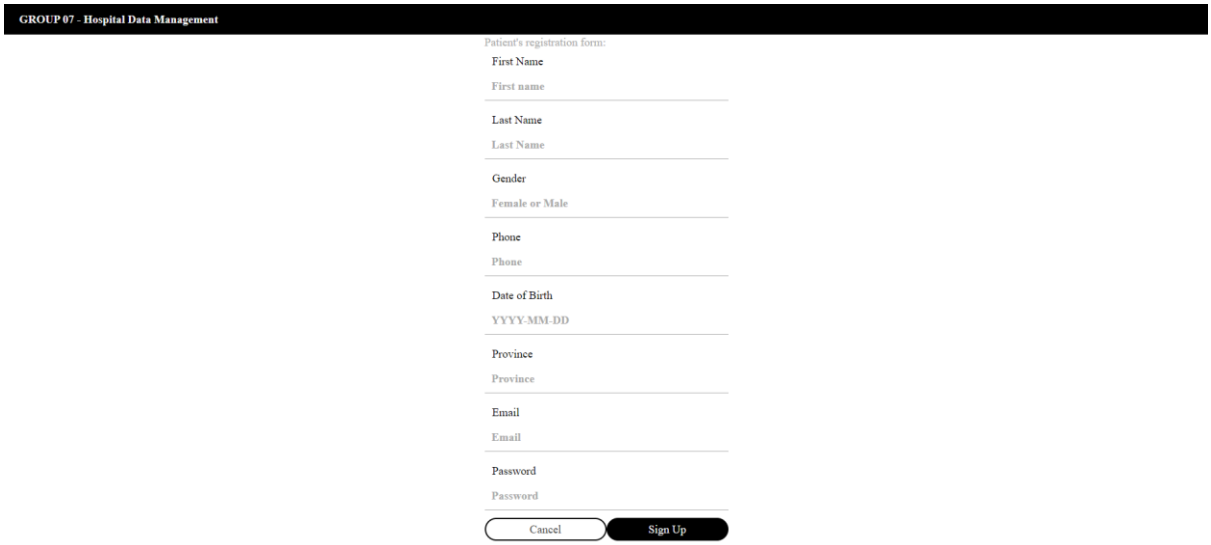
Step 2: The system will take the information that doctor has filled in the application to insert into the database. Email, phone number and full name of doctor will be inserted into the 'Doctor' table. Email, password and usertype (doctor) will be inserted into the 'Account' table (email will be the username of the account).

```
//Create Doctor Account
app.get('/makeDocAccount', (req, res) => {
  let params = req.query;
  let name = params.name + " " + params.lastname;
  let email = params.email;
  let password = params.password;
  let phone = params.phone;
  let sql_statement = `INSERT INTO Doctor (doc_email, doc_phone, doc_name)
                      VALUES ` + `("${email}", "${phone}", "${name}")`;
  console.log(sql_statement);
  con.query(sql_statement, function (error, results, fields) {
    if (error) throw error;
    else {
      let sql_statement = `INSERT INTO Account (username, passwo, usertype)
```

GitHub repository: [Group 07]

```
                      VALUES ` + `("${email}", "${password}", "Doctor")`;
      console.log(sql_statement);
      con.query(sql_statement, function(error){
        if (error) throw error;
      })
      email_in_use = email;
      password_in_use = password;
      who = 'doc';
      return res.json({
        data: results
      })
    };
  });
});
```

b. Patient registration:



Figures 4. Patient Registration

Step 1: Similarly to the Doctor registration, system will check for existed email.
Step 2: Data insertion
The filled information will be inserted into 'Account' and 'Patient' table

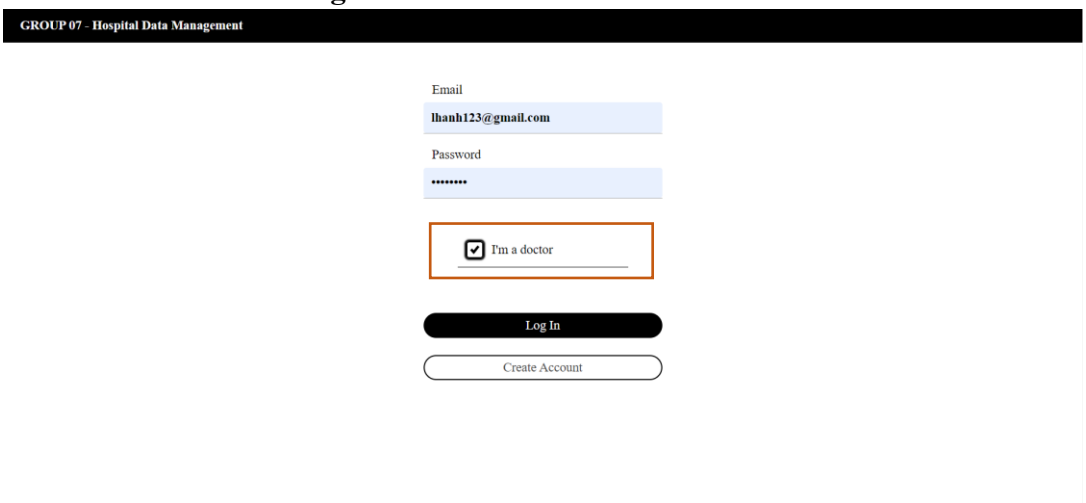
```
//Creates Patient Account
app.get('/makeAccount', (req, res) => {
  let query = req.query;
  let name = query.name + " " + query.lastname;
  let dob = query.dob;
  let email = query.email;
  let phone = query.phone;
  let password = query.password;
  let province = query.province;
  let gender = query.gender;

  let sql_statement1 = `INSERT INTO Account (username, passwo, usertype)
                      VALUES ` + `("${email}", "${password}", "Patient")`;
  console.log(sql_statement1);
  con.query(sql_statement1, function (error, results, fields) {
    if (error) throw error;
    else {
```

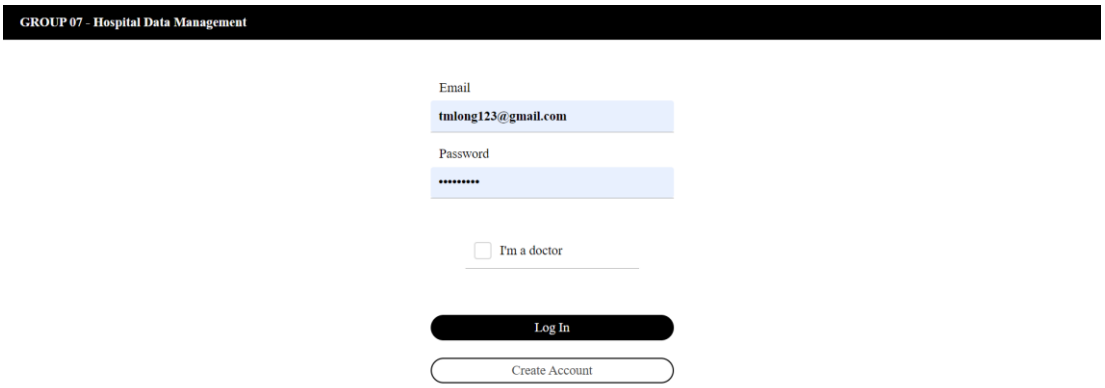
GitHub repository: [Group 07]

```
        let sql_statement1 = `INSERT INTO Patient (p_email, p_dob, p_name,
p_province, p_biogender, p_phone)
                              VALUES ` + `("${email}", "${dob}", "${name}",
"${province}", "${gender}", "${phone}")`;
        console.log(sql_statement1);
        con.query(sql_statement1, function (error, results, fields) {
            if (error) throw error;
        })
        email_in_use = email;
        password_in_use = password;
        who="pat";
        return res.json({
            data: results
      })};
  });
});
```

c. Doctor and Patient login:



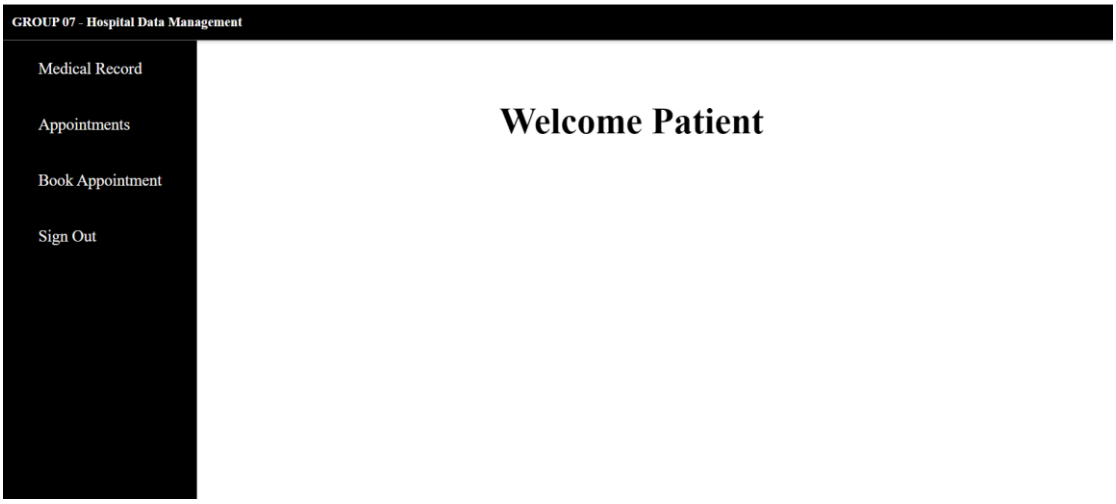
Figures 5. Doctor Login



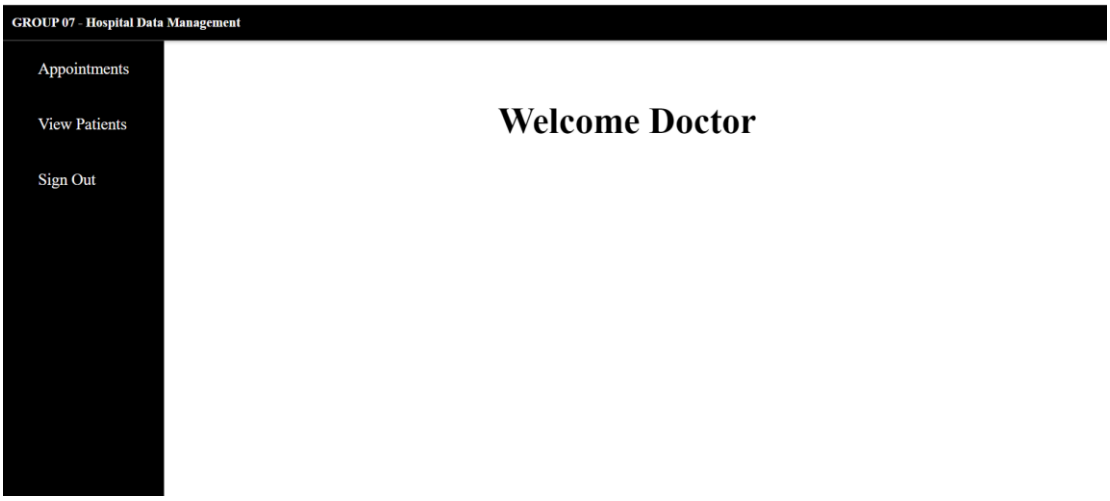
Figures 6. Patient Login

Login functions for doctors and patients are similar that requires to fill in the blanks of username (email) and password. If users are doctors, they must click on the box of 'I'm a doctor'.

GitHub repository: [Group 07]



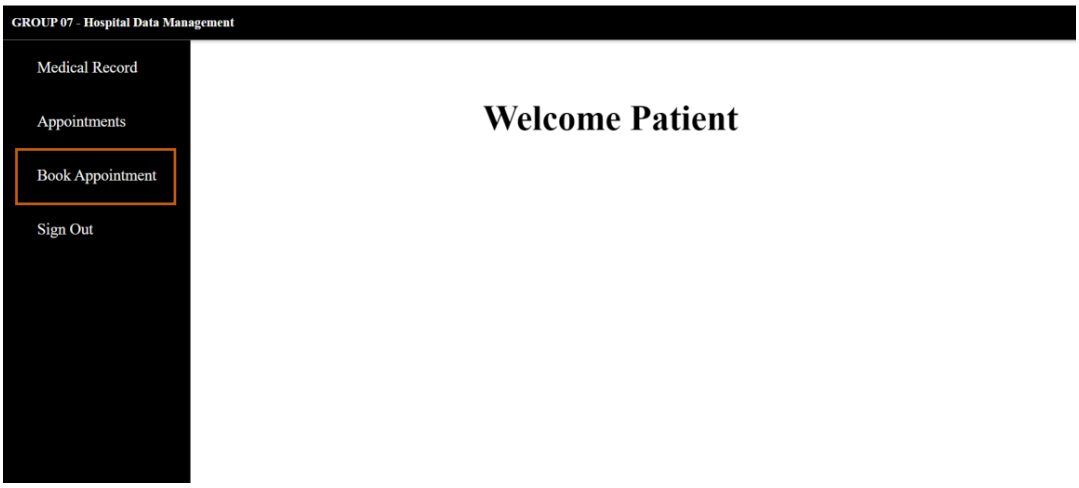
Figures 7. Successful Patient Login

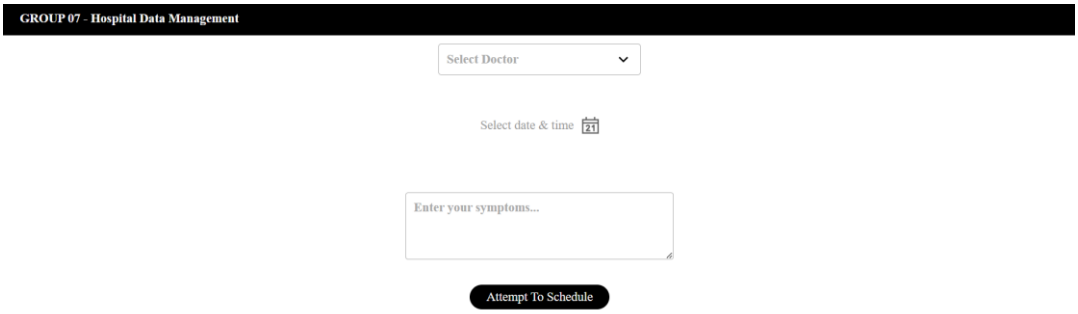


Figures 8. Successful Doctor Login

3.3. Feature 2 – Schedule and View Appointment

a. Patient Book Appointment:





Figures 9. Patient Book Appointment

Patients can choose a doctor, date, time for appointment and list some symptoms of their health conditions.

Whenever clicking on 'Apptempt To Schedule' button, the system will generate an ID number for the new appointment by firstly querying ID of the latest appointment in the database:

```
//Generates ID for appointment
app.get('/genApptUID', (req, res) => {
  let statement = 'SELECT a_id as id FROM Appointment ORDER BY id DESC LIMIT 1;'
  con.query(statement, function (error, results, fields) {
    if (error) throw error;
    else {
      let generated_id = results[0].id + 1;
      return res.json({ id: `${generated_id}` });
    };
  });
});
```

Then, inserting new appointment's information into 'Appointment' table

```
app.get('/schedule', (req, res) => {
  let params = req.query;
  let time = params.time;
  let date = params.date;
  let id = params.id;
  let endtime = params.endTime;
  let ndate = new Date(date).toLocaleDateString().substring(0, 10)
  let sql_date = `STR_TO_DATE('${ndate}', '%m/%d/%Y')`;
  //sql to turn string to sql time obj
  let sql_start = `CONVERT('${time}', TIME)`;
  //sql to turn string to sql time obj
  let sql_end = `CONVERT('${endtime}', TIME)`;
  let doctor = params.doc;
  let symptom = params.symptoms;
  let sql_try = `INSERT INTO Appointment (a_id, a_date, starttime, endtime,
status)
                 VALUES (${id}, ${sql_date}, ${sql_start}, ${sql_end},
"NotDone")`;
  console.log(sql_try);
  con.query(sql_try, function (error, result, fields) {
    if (error) throw error;
    else {
      return res.json({
```

GitHub repository: [Group 07]

```
        data: result
      })}});
```

After that, do insertion into 'PatientBookAppointment', 'DoctorViewAppointment', and 'MedicalRecord' table.

```
let sql_try1 = `INSERT INTO PatientBookAppointment (p_id, a_id, symptoms)
                VALUES (${p_id2}, ${id}, "${symptoms}")`;
let sql =` `INSERT INTO DoctorViewAppointment (doc_id, a_id)
            VALUES (${doc_id}, ${id})`;
let sql = `INSERT INTO MedicalRecord (mr_date, symptom)
                VALUES (${sql_date}, "${symptom}")`;
```

b. Patient View their Scheduled Appointments

GROUP 07 - Hospital Data Management

Date of Appointment	Start Time	End Time	Symptoms	Status		
12/1/2023	10:00	11:00		Done	See Diagnosis	Delete
12/14/2023	09:00	10:00	t	NotDone	See Diagnosis	Cancel
12/14/2023	09:00	10:00	t	NotDone	See Diagnosis	Cancel
12/18/2023	15:00	16:00	rr	NotDone	See Diagnosis	Cancel
12/19/2023	09:00	10:00	nn	NotDone	See Diagnosis	Cancel
12/26/2023	08:00	09:00	mm	NotDone	See Diagnosis	Cancel
12/21/2023	08:00	09:00	xx	NotDone	See Diagnosis	Cancel
12/21/2023	08:00	09:00	xx	NotDone	See Diagnosis	Cancel
12/26/2023	10:00	11:00	zz	NotDone	See Diagnosis	Cancel

Figures 10. Patient View Scheduled Appointment

Query data from the database to display on the application.

```
let statement = `SELECT b.a_id as ID,
                  p.p_name as user,
                  b.symptoms as theSymptoms,
                  a.a_date as theDate,
                  a.starttime as theStart,
                  a.endtime as theEnd,
                  a.status as status
                  FROM Appointment a
                  JOIN PatientBookAppointment b ON a.a_id = b.a_id
                  JOIN Patient p ON b.p_id = p.p_id
                  WHERE p.p_email = "${email}"`;
```

V. CONCLUSION

1. Achieved Goals

In this project, we successfully achieved several significant goals aimed at enhancing the efficiency and functionality of the hospital data management system. Firstly, we successfully

GitHub repository: [Group 07]

designed and built a robust relational database that adheres to the Third Normal Form (3NF) standards. This accomplishment ensures a structured and efficient organization of data, promoting data integrity and minimizing redundancy within the system.

In tandem with the database development, we created a user-friendly application to serve as the interface for accessing and interacting with the database. This application not only provides a seamless display of the database content but also incorporates two interactive features. These features facilitate a two-way communication between the user and the database, enhancing the overall user experience and enabling efficient data manipulation.

2. Future Works

Looking ahead, our future work encompasses the development of an additional feature within the application. This feature aims to empower patients to access and view their medical records, providing them with a secure and convenient means to monitor their health information. Simultaneously, doctors will gain access to an integrated system where they can view both patient appointments and medical records, streamlining their workflow and improving patient care.

Furthermore, as part of our future initiatives, we plan to re-engineer the database structure to achieve an even higher level of normalization. This strategic move is intended to further optimize data organization, reduce redundancies, and enhance the overall efficiency and performance of the system. By continuously refining and expanding the capabilities of the database and application, our project is poised to contribute significantly to the seamless management of hospital data and the delivery of enhanced healthcare services.

VI. REFERENCES

GeeksforGeeks. (2023, September 20). Retrieved from Introduction of ER Model: https://www.geeksforgeeks.org/introduction-of-er-model/

GitHub repository: Group 07